اصول امنیت شبکه‌های کامپیوتری

کاربردها و استانداردها

ویرایش سوّم (۲۰۰۷ میلادی)

اثر: William Stallings

ترجمهٔ: مسعود موحّد

مدرّس

مرکز آموزش شرکت مخابرات ایران

و

دانشکدهٔ علمی_کاربردی پست و مخابرات

در بارهٔ نویسنده :

William Stallings دارای لیسانس مهندسی برق از دانشگاه Notre Dame و درجهٔ Ph.D. از دانشگاه M.I.T. کشور امریکاست. خلاقیت‌های او اثرات منحصربفردی در فهم شبکه‌های کامپیوتری و معماری کامپیوتر داشته است. وی مؤلف ۱۷ کتاب در زمینه‌های علوم کامپیوتر است که هرکدام بارها تجدید چاپ شده‌اند. در ۲۰ سال گذشته وی مدیر چندین مؤسسهٔ فنی و تحقیقاتی بوده و در حال حاضر یک مشاور مستقل است که مشتریان او کارخانجات سازندهٔ کامپیوتر و تجهیزات شبکه، تولیدکنندگان نرم‌افزار و مؤسسات تحقیقاتی برجسته دولت امریکا می‌باشند. نامبرده هفت بار برندهٔ جایزهٔ بهترین کتاب‌های دانشگاهی علوم مهندسی کامپیوتر بوده است.

Stallings سایت مرجع دانشجویان رشتهٔ کامپیوتر با آدرس WilliamStallings.com/StudentSupport/html را خلق کرده است که اسناد و لینک‌های بسیار متنوعی در زمینه‌های مختلف علوم کامپیوتری برای افراد آماتور و حرفه‌ای فراهم می‌آورد. او عضو هیئت تحریریهٔ مجلهٔ *Cryptologia* است که آخرین دستاوردهای علم رمزشناسی را عرضه می‌نماید. دربین کتاب‌های اوکتاب *Data and Computer Communications*، که اخیراً ویرایش هشتم آن بچاپ رسیده است، بعنوان کتاب استاندارد این رشته در سراسر دنیا شناخته شده است.

در بارهٔ مترجم :

مسعود موّحد دارای فوق لیسانس مهندسی برق و الکترونیک از دانشکدهٔ فنی دانشگاه تهران و درجهٔ D.Eng. در رشتهٔ مخابرات از دانشگاه George Washington کشور امریکاست. وی از سال ۱۳۵۸ به خدمت شرکت مخابرات ایران درآمده و تا کنون ضمن تصدی مشاغل مختلف مدیریت آموزشی، به تدریس در زمینه‌های الکترونیک و مخابرات مشغول بوده است. نامبرده همچنین بمدت چهارسال رئیس مرکز آموزش و مجری طرح‌های آموزشی شرکت مخابرات ایران بوده و بطور همزمان سرپرستی دانشکدهٔ پست و مخابرات وابسته به وزارت ارتباطات و فناوری اطلاعات را بعهده داشته است.

نامبرده در حال حاضر در زمینهٔ مخابره و انتقال داده‌ها، شبکه‌های کامپیوتری و علی‌الخصوص مسائل امنیتی شبکه‌ها و اینترنت به تدریس و تحقیق اشتغال دارد. وی کتاب حاضر را بارها در دوره‌های ضمن خدمت کارشناسان شرکت‌های وابسته به وزارت ارتباطات تدریس نموده است.

بنام آفرینندهٔ اندیشه‌های ژرف

اگرچه ترجمهٔ این کتاب را از مدت‌ها قبل آماده کرده بودم ولی انتشار آن را به دو دلیل به تعویق انداختم. اول اینکه چندین بار آن را تدریس کنم تا ضمن کسب تجربه، با استنباط نویسندهٔ محترم در مورد مطالب کتاب بیشترآشنا شوم و دوم اینکه ویرایش سوم این کتاب که برای سال ۲۰۰۷ میلادی آماده شده است به بازار آید تا آخرین تغییراتی که مؤلف محترم در متن کتاب اعمال نموده‌اند را نیز در متن ترجمه شده وارد نمایم. اکنون که این دو امر حاصل شده است، خداوند را شاکرم که بمن توفیق داد تا گامی بسیار کوچک در وادی بسیار بزرگ امنیت شبکه بردارم. از کلیهٔ دانشجویانی که در کلاس درس با راهنمائی‌های ارزندهٔ خود مشوق من در این امر بوده‌اند سپاسگزارم و از همهٔ خوانندگانی که متن کتاب را ملاحظه می‌فرمایند تقاضا دارم تا بر من منت گذاشته و با ارسال انتقادات و پیشنهادات خود به آدرس پست الکترونیک **movahed@ictfaculty.ir** من را در فعالیت‌های آینده‌ام راهنمائی فرمایند.

در خاتمه لازم است از جناب آقای دکتر غلامعلی حسنی صدر رئیس محترم مرکز آموزش شرکت مخابرات ایران و دانشکدهٔ علمی-کاربردی پست و مخابرات که دستور چاپ این کتاب را صادر فرموده‌اند و همچنین از سرکار خانم معصومهٔ گرامی‌زاده کارشناس محترم مرکز آموزش مخابرات ایران که در تهیهٔ شکل‌ها و جداول کتاب زحمات زیادی را متقبل شده‌اند صمیمانه سپاسگزاری نمایم.

مسعود موحّد

زمستان ۱۳۸۵

فهرست مطالب

فصل ۳ رمزنگاری کلید عمومی و اعتبارسنجی پیام ۷۳

قسمت دوم کاربردهای امنیت شبکه ۱۰۹

فصل ۴ کاربردهای اعتبارسنجی ۱۱۱

فصل ۵ امنیت پست الکترونیک ۱۵۳

پیش گفتار

در این عصر اتصال الکترونیکی جهان، عصر ویروس‌ها و هَکرها، عصر استراق سمع الکترونیک و عصر فریب الکترونیک، واقعاً زمانی قابل تصور نیست که در آن امنیت مطرح نباشد. دو رَوَند متفاوت گرد هم آمده‌اند تا موضوع این کتاب را از اهمیت حیاتی برخوردار سازند. اوّل، رشد انفجارآمیز سیستم‌های کامپیوتری و اتصال آنها از طریق شبکه‌ها، وابستگی هم سازمان‌ها و هم افراد به اطلاعات ذخیره‌شده و مبادله شده با استفاده از این سیستم‌ها را افزایش داده است. این امر بنوبهٔ خود هوشیاری کاربران نسبت به لزوم محافظت دیتا و منابع اطلاعاتی از افشا، تضمین موثق بودن داده‌ها و پیام‌ها، و محافظت سیستم‌ها از حملات مبتنی بر شبکه را ارتقاء بخشیده است. دوّم، مقوله‌های مرتبط با رمزنگاری و امنیت شبکه کمال بیشتری یافته و منجر به ایجاد کاربردهای عملی‌تر وآماده‌تری برای اعمال امنیت شبکه شده‌اند.

اهداف

هدف این کتاب، فراهم‌آوردن یک بررسی عملی از کاربردها و استانداردهای امنیت شبکه است. در مورد کاربردها، تأکید برکاربردهائی خواهد بود که بطور فزاینده‌ای در شبکه‌های سازمان‌ها و اینترنت مورد استفاده قرار گرفته‌اند. در مورد استانداردها نیز تأکید بر روی استانداردهای اینترنتی است که در سطح گسترده‌ای مورد استفاده هستند.

مخاطبین مورد نظر

این کتاب هم برای مخاطبین دانشگاهی و هم برای مخاطبین حرفه‌ای تدارک دیده شده است. بعنوان یک کتاب دانشگاهی، برای یک درس سه واحدی امنیت شبکه دورهٔ لیسانس برای دانشجویان رشتهٔ علوم کامپیوتر، مهندسی کامپیوتر و مهندسی برق مناسب است. این کتاب همچنین می‌تواند یک مرجع قابل مطالعه برای افراد علاقه‌مند باشد.

طراحی کتاب

کتاب در سه قسمت سازمان یافته است:

قسمت اول- رمزنگاری: مرور فشرده‌ای بر الگوریتم‌ها و پروتکل‌های زیرساخت امنیت شبکه دارد که شامل رمزنگاری، توابع درهم‌ساز، امضاءهای دیجیتال و مبادلهٔ کلید است.

قسمت دوم-کاربردهای امنیت شبکه: ابزارهای مهم امنیت شبکه، شامل Kerberos، گواهی‌نامه‌های X.509v3، PGP، S/MIME، امنیت IP، SSL/TLS، SET و SNMPv3 بررسی می‌شوند.

قسمت سوم- امنیت سیستم: به مقوله‌های امنیت در سطح سیستم می‌پردازد که شامل تهدیدها و روش‌های مقابله با آنها در برابر تهاجم و ویروس‌ها، و همچنین استفاده از دیوارهای آتش و سیستم‌های معتمد است.

علاوه بر آن،کتاب شامل یک واژه‌نامهٔ مفصل، یک لیست از علائم اختصاری کثیرالاستعمال و یک فهرست مراجع است. هر فصل شامل تعدادی سؤال، مسأله، لیستی از واژه‌های کلیدی و منابع پیشنهادی برای مطالعهٔ بیشتر است.

خلاصهٔ مفصل‌تری از مطالب طرح‌شده در فصول هر قسمت، در ابتدای آن قسمت ذکر شده است.

مطالب پشتیبان تدریس

برای پشتیبانی مدرسین، مطالب زیر فراهم آمده است:

- **حل المسائل**: پاسخ به تمام سؤالات مرورکنندهٔ بحث و مسائل انتهای هر فصل.
- **اسلایدهای PowerPoint**: مجموعه‌ای از اسلایدها برای تمام فصول، که به امر تدریس کمک می‌کند.
- **فایل‌های PDF**: تمام شکل‌ها و جداول کتاب.
- **لیست پروژه‌ها**: تکالیفی بصورت پروژه، در تمام مقوله‌هائی که در زیر ذکر شده است.

مدرسین می‌توانند برای دسترسی به این مطالب با نمایندهٔ Pearson Education یا Prentice Hall تماس حاصل کنند.

علاوه بر این، پایگاه وِب کتاب مدرسین را با موارد زیر مجهز می‌کند:

- لینک‌هائی به پایگاه‌های وِب درس‌های دیگر، که در تدریس آنها از این کتاب استفاده می‌شود.
- اطلاعات عضویت برای یک لیست پستی اینترنتی برای مدرسین.

سرویس اینترنتی برای اساتید و دانشجویان

برای این کتاب یک صفحهٔ وِب وجود دارد که برای استفادهٔ دانشجویان و اساتید طراحی شده است. این صفحه شامل لینک‌هائی به سایت‌های مرتبط، شکل‌ها و جداول کتاب با فرمت PDF و اطلاعات ثبت نام برای لیست پستی اینترنتی کتاب است. آدرس صفحهٔ وِب **WilliamStallings.com/NetSec2e.html** است. یک لیست پستی اینترنتی هم فراهم شده است تا مدرسینی که این کتاب را تدریس می‌کنند بتوانند اطلاعات، پیشنهادها و سؤالات خود را با یکدیگر و با نویسنده مبادله نمایند. غلط های احتمالی موجود در کتاب نیز در یک لیست غلط‌نامه وارد شده‌اند. علاوه برآن، سایت مرجع دانشجویان علوم کامپیوتر با آدرس **WilliamStallings.com/StudentSupport.html** نیز اسناد، اطلاعات و لینک‌های مفیدی را برای دانشجویان علوم کامپیوتر و افراد حرفه‌ای به همراه دارد.

پروژه‌های مربوط به تدریس امنیت شبکه

برای بسیاری از مدرسین، بخش مهمی از یک درس رمزنگاری و یا امنیت شبکه، یک پروژه یا مجموعه‌ای از پروژه‌هائی است که بتوسط آنها دانشجویان پشتوانهٔ تجربی بهتری را از درس بدست آورند. این کتاب با فراهم آوردن مؤلفهٔ پروژه، نوعی پشتیبانی غیر موازی از درس را بوجود آورده است. جزوهٔ راهنمای استاد نه تنها شامل راهنمائی در مورد تخصیص و ساخت پروژه است، بلکه شامل یک مجموعه از پروژه‌های پیشنهادی است که بازهٔ وسیعی از موضوعات متن را پوشش می‌دهد:

- **پروژه‌های تحقیقاتی**: یک سری از موضوعاتی که به دانشجو آموزش می‌دهد تا چگونه دربارهٔ یک موضوع در اینترنت تحقیق کرده و گزارش تهیه کند.
- **پروژه‌های برنامه‌نویسی**: یک سری از پروژه‌های برنامه‌نویسی که محدودهٔ وسیعی از موضوعات را پوشش داده و می‌توانند به هر زبان مناسبی و روی هر کامپیوتری نوشته شوند.
- **تمرین‌های آزمایشگاهی**: یک سری پروژه‌هائی که شامل برنامه‌نویسی و تجربه آموزی در بارهٔ مفاهیم کتاب است.
- **تکالیف نوشتنی**: یک مجموعه از تکالیف کتبی برای هر فصل.
- **تکالیف مطالعاتی/ گزارش‌دهی**: یک لیست از مقالات موجود در مقوله‌های ذیربط، یکی برای هر فصل، که می‌تواند به دانشجو محول شده تا از روی آن گزارش تهیه نماید.

مطالب جدید در ویرایش سوم

در ظرف سه سالی که از چاپ دوم این کتاب می‌گذرد، مقولهٔ امنیت، نوآوری‌ها و توسعه‌های جدیدی را بخود دیده است. در این چاپ جدید، من کوشیده‌ام تا در ضمن این که ساختار کلی کتاب را تغییر ندهم، تغییرات جدید را نیز درآن منعکس نمایم. برای شروع این بازنگری، چاپ دوم بتوسط اساتید متعددی که این کتاب را تدریس نموده‌اند مرور گردید. علاوه برآن عده‌ای از افراد شاغل در این حرفه نیز فصول این کتاب را مطالعه نمودند. نتیجهٔ کار این شد که در برخی موارد، انشای مطلب برای فهم بهتر تصحیح گردید و شکل‌ها تغییر یافت. همچنین تعدادقابل توجه‌ای مسأله، به فصول اضافه گردید.

علاوه بر این اصلاحات که در جهت فهم بهتر مطلب انجام شد، تغییرات قابل توجهی نیز در برخی سرفصل‌ها ایجاد گردید. عمدهٔ آنها بشرح زیر است:

- **رمزهای دنباله‌ای**: رمزهای دنباله‌ای در تعدادی از پروتکل‌های امنیت شبکه و کاربردها بکار می‌روند. چاپ سوم، این مقوله را پوشش داده و پراستفاده‌ترین الگوریتم این حوزه که RC4 است را توصیف کرده است.
- **زیرساخت کلید عمومی (PKI)**: این مقولهٔ مهم در ویرایش جدید مورد بحث قرار گرفته است.
- **حملات انکار سرویس توزیع شده (DDoS)**: حملات DDoS در سال‌های اخیر توجه زیادی را بخود جلب کرده‌اند.
- **معیارهای مشترک برای ارزیابی امنیت تکنولوژی اطلاعات**: معیارهای مشترک، به یک چهارچوب بین‌المللی برای تشریح نیازهای امنیتی و ارزیابی محصولات و پیاده‌سازی‌ها مبدل شده‌اند.

علاوه براین، بسیاری از سایر مطالب کتاب بازنگری و بروزرسانی شده‌اند.

ارتباط این کتاب با کتاب رمزنگاری و امنیت شبکه، ویرایش چهارم

این کتاب از کتاب *Cryptography and Network Security, Fourth Edition* (CNS4e) اقتباس شده است. CNS4e پوشش جامعی از مبحث رمزنگاری داشته که شامل تحلیل مفصلی از الگوریتم‌ها و پشتوانهٔ ریاضی آنهاست که خود قریب به ۴۰۰ صفحه از کتاب را دربر می‌گیرد.

Network Security Essentials: Applications and Standards, Third Edition (NSE3e) درعوض مرور مختصری بر مباحث فوق در فصول ۲ و ۳ دارد. مابقی NSE3e شامل کلیهٔ مطالب باقیمانده CNS4e است. NSE3e همچنین امنیت SNMP را پوشش می‌دهد که در CNS4e پوشش داده نشده است. بنابراین NSE3e برای تدریس در کالج‌ها و برای خوانندگان حرفه‌ای که علاقهٔ آنها بیشتر به کاربردهای امنیت شبکه بوده و نیاز و توجه کمتری به عمیق شدن در تئوری‌ها و اصول رمزنگاری دارند، طراحی شده است.

فصل ۱

مقدمه

زمه‌های **امنیت اطلاعات** در درون یک سازمان، در طی دهه‌های اخیر دو تغییر عمده یافته است. قبل از استفادهٔ گسترده از تجهیزات پردازش داده‌ها، امنیت اطلاعاتی که از نظر یک سازمان ارزشمند تلقی می‌شد عمدتاً از طـــریق روش‌های فیزیکی و مدیریتی فراهم می‌گردید. مثالی از روش فیزیکی، استفاده از کمدها و فایل‌های مستحکم با قفل های رمزی برای حفاظت از اسناد مهم است. مثالی از روش مدیریتی، جمع‌آوری اطلاعات گزینشی در هنگام استخدام پرسنل است.

با ورود رایانه، نیاز به لوازم خودکار برای حفاظت از فایل‌ها و سایر اطلاعات ذخیره‌شده در رایانه آشکار گردید. این موضوع علی‌الخصوص در مورد یک سیستم به اشتراک گذاشته شده همانند یک سیستم اشتراک زمانی جدی‌تر بوده و حتی در مورد سیستم‌هائی که از طریق شبکهٔ تلفنی، شبکهٔ دیتا، و یا اینترنت قابل دست‌یابی هستند حیاتی‌تر است. نام کلی مجموعهٔ لوازمی که برای حفاظت داده‌ها و خنثی کردن نیات بداندیشان طراحی شده است، **امنیت رایانه** است.

تغییر عمدهٔ دیگر که امنیت را تحت تأثیر قرار داده است، ورود سیستم‌های توزیع شده و استفاده از تسهیلات شبکه‌ای و ارتباطی برای حمل داده‌ها بین پایانه و رایانه، و بین رایانه و رایانه است. معیارهای امنیت شبکه برای حفاظت از داده‌ها در هنگام انتقال آن‌ها ضروری است. در واقع اصطلاح **امنیت شبکه** تا حدودی گمراه‌کننده است زیرا تمام مشاغل، دولت و سازمان‌های آموزشی، تجهیزات پردازش دیتای خود را با مجموعه‌ای از شبکه‌های متعامل بهم وصل کرده‌اند. چنین مجموعه‌ای را اغلب اینترنت نامند و در رابطه با آن، اصطلاح **امنیت اینترنت** و یا امنیت بین شبکه‌ای بکار می‌رود.

بین این دو نوع امنیت، مرزبندی روشنی وجود ندارد. مثلاً یکی از معروف‌ترین انواع حملات بر روی سیستم‌های اطلاعاتی، ویروس رایانه‌ای است. یک ویروس ممکن است از طریق یک دیسکت و یا یک دیسک نوری بصورت فیزیکی وارد شده و متعاقباً روی رایانه‌ای بارگذاری شود. ویروس‌ها همچنین ممکن است از طریق اینترنت وارد شوند. در هر یک از دو مورد، همین‌که ویروس روی یک سیستم رایانه مستقر گردید، لوازم امنیت داخلی رایانه برای تشخیص آن و نجات رایانه از شرّ آن، مورد نیاز خواهد بود.

این کتاب بر روی امنیت اینترنت متمرکز شده است که شامل معیارهائی برای شناسائی، جلوگیری، تشخیص و اصلاح تخلفات امنیتی که اطلاعات را تحت تأثیر قرار می‌دهند، می‌باشد. این مقولهٔ گسترده‌ای است که موارد وسیعی را در بردارد. برای این‌که احساسی از مسائلی که دراین کتاب مورد بحث قرار می‌گیرد داشته باشید، به مثال‌های زیر از مخاطرات امنیتی توجه کنید:

۱- کاربر A یک فایل را برای کاربر B می‌فرستد. فایل شامل اطلاعات حساسی (مثل اطلاعات مالی) است که بایستی از دسترس بیگانه دور باشد. کاربر C که مجاز به خواندن فایل نمی‌باشد، قادر به پائیدن انتقال اطلاعات بوده و یک نسخه از فایل را در هنگام ارسال به دست می‌آورد.

۲- یک مدیر شبکه D، پیامی را برای رایانهٔ E که تحت مدیریت اوست می‌فرستد. پیام به رایانهٔ E فرمان می‌دهد که فایل افراد مجاز را به‌روزکرده و نام چندین کاربر جدیدکه می‌توانند به سیستم دست یابند را در آن وارد کند. کاربر F پیام را دزدیده، محتویات آن را با اضافه کردن و یا حذف کردن نام‌های دلخواه خود تغییر داده و سپس آن را برای E می‌فرستد. E پیام را با تصور این‌که از سوی مدیر D ارسال شده پذیرفته و فایل افراد مجاز را بر اساس آن به‌روز درمی‌آورد.

۳- بجای دزدیدن یک پیام، کاربر F، پیام مورد نظر خویش با ورودی‌های دلخواه خود را ساخته و آن را طوری برای E می‌فرستد که E خیال می‌کند از جانب مدیرD صادر شده است و بنابراین فایل افراد مجاز را بر اساس آن به‌روز در می‌آورد.

۴- کارمندی بدون اخطار قبلی اخراج می‌شود. مدیر امور اداری پیامی به سیستم سِرور می‌فرستد تا حساب او را از اعتبار خارج نماید. وقتی این عمل انجام می‌شود، سِرور بایستی تذکاریه‌ای را برای فایل کارمند ارسال کرده و انجام عمل را تأئید کند. کارمند قادر به استراق سمع پیام بوده و ارسال آن را آنقدر به تأخیر می‌اندازد تا خود بتواند آخرین دست‌یابی به سِرور را پیدا کرده و اطلاعات حساس را استخراج کند. پس از آن پیام ارسال شده، عمل صورت پذیرفته و تأئید آن داده می‌شود. عمل این کارمند ممکن است برای مدت قابل ملاحظه‌ای کشف نشود.

۵- یک پیام ممکن است از طرف یک مشتری برای خرید سهام به کارگزار او فرستاده شود. متعاقباً ممکن است قیمت سهام پائین آمده و مشتری ارسال چنین پیامی را انکار کند.

اگرچه این لیست بهیچ وجه تمام تهدیدهای امنیتی را پوشش نمی‌دهد، ولی نمایش‌گر محدودهٔ وسیع امنیت شبکه است.

امنیت بین شبکه‌ای، هم جذاب و هم پیچیده است. برخی دلایل آن به قرار زیراند:

۱- امنیتی که ارتباطات و شبکه‌ها درگیرآنند آنچنان که در ابتدا برای یک تازه‌کار جلوه می‌کند، ساده نیست. اهداف این امنیت ممکن است خیلی ساده بیان شوند و در واقع نیازهای اساسی سیستم‌های امنیتی اغلب با یک کلمه بیان می‌شوند: محرمانگی، اعتبارسنجی، عدم انکار، صحت. اما مکانیسم‌هائی که بایستی برای حصول این نیازها استفاده شوند اغلب بسیار پیچیده بوده و فهم آنها ممکن است استدلالات زیرکانه‌ای را ایجاب کند.

۲- در طراحی یک مکانیسم و یا الگوریتم امنیتی بخصوص، همیشه بایستی حملات مؤثر بر علیه آن ویژگی امنیتی را در نظر داشت. در بسیاری موارد، حملات موفقیت‌آمیز با نگاهی کاملاً متفاوت به مسأله طراحی می‌شوند و از نقاط ضعف غیرقابل انتظار مکانیسم استفاده می‌کنند.

۳- بعلت نکتهٔ بالا، روش‌های مورد استفاده برای فراهم نمودن سرویس‌های بخصوص، اغلب ساده به ذهن نمی‌آیند. اغلب بیان یک نیاز امنیتی نمی‌تواند زنجیرهٔ پیچیده‌ای از عملیاتی که طراحی شده‌اند را به راحتی توجیه کند. تنها زمانی این معیارها معنی پیدا می‌کنند که شگردهای ضدامنیتی آنها مطالعه شوند.

۴- وقتی مکانیسم‌های متفاوت امنیتی طراحی شدند، لازم است تصمیم گرفته شود که در کجا باید از آنها استفاده کرد. این امر هم در مورد محل فیزیکی آنها (این که مکانیسم‌های امنیتی در کدام نقطهٔ شبکه مورد نیازند) و هم در مفهوم منطقی آنها (این که مکانیسم‌های امنیتی درکدام لایه و یا لایه‌های یک معماری، مثل TCP/IP، باید گنجانده شوند)، صحیح می‌باشد.

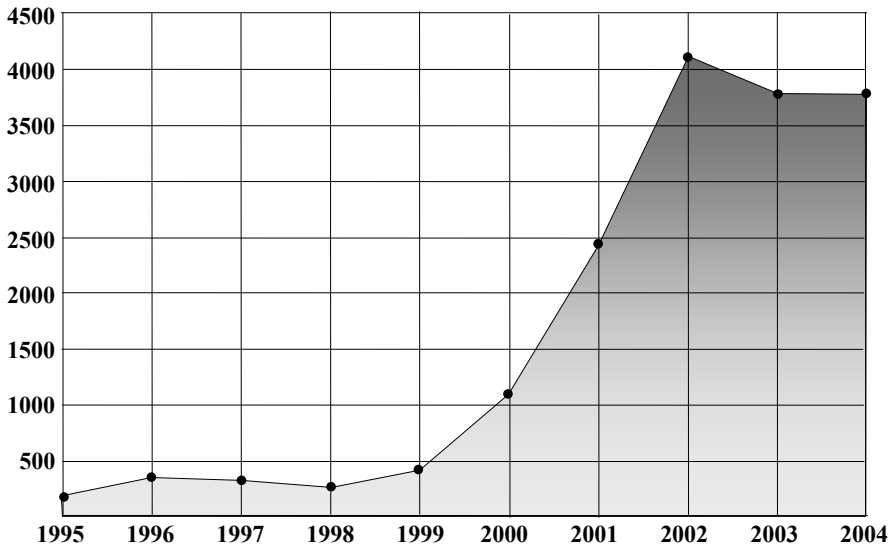
۵- مکانیسم‌های امنیتی معمولاً شامل بیش از یک الگوریتم و یا یک پروتکل هستند. آنها معمولاً اشخاص را مقید می‌کنند تا برخی اطلاعات سرّی را در اختیار داشته باشند (مثلاً یک کلید رمزنگاری) که این خود سؤالاتی در زمینهٔ تولید، توزیع و حفاظت از این اطلاعات سرّی را مطرح می‌سازد. همچنین اتکاء به رفتار برخی پروتکل‌های ارتباطی ممکن است وظیفهٔ طراحی مکانیسم‌های امنیتی را با مشکل مواجه سازد. بعنوان مثال، اگر عملکرد صحیح یک مکانیسم امنیتی نیاز به محدود کردن زمان انتقال یک پیام بین فرستنده و گیرنده را داشته باشد، آنگاه هر پروتکل یا شبکه‌ای که تأخیر زمانی متغیر و یا غیرقابل انتظاری در ارسال پیام را ایجاد نماید، ممکن است این معیار امنیتی را بی‌معنی سازد.

بنابراین، نکات بسیاری را بایستی در نظر داشت. این فصل یک نگاه کلّی به مسأله داشته و ساختار مطالب بقیهٔ کتاب را سازمان می‌دهد. موضوع را با یک بحث کلی در مورد سرویس‌های امنیت شبکه و انواع حملات امنیتی که این سرویس‌ها بایستی به آنها پاسخ دهند، شروع می‌کنیم. آنگاه یک مدل عمومی که سرویس‌ها و مکانیسم‌های امنیتی بایستی از منظر آن دیده شوند را ارائه می‌دهیم.

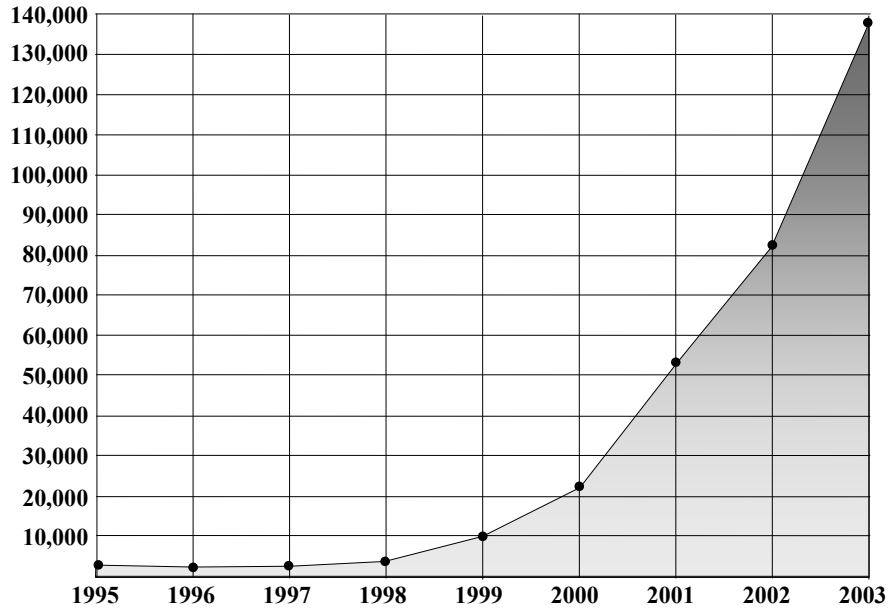
۱-۱ رَوَند امنیت

در سال ۱۹۹۴ میلادی، گروه معماری اینترنت (IAB) گزارشی با عنوان «امنیت در معماری اینترنت» را منتشر نمود (RFC 1636). این گزارش بیان‌کنندهٔ توافق جمعی بر این مطلب بود که اینترنت به امنیت بیشتر و بهتری نیاز دارد. در ضمن، زمینه‌های کلیدی مکانیسم‌های امنیتی نیز در این گزارش مشخص شده بود. در این گزارش به مقوله‌هائی چون نیاز به مصون کردن زیرساخت شبکه از پایش‌های غیرمجاز، کنترل ترافیک شبکه و امن کردن ترافیک کاربرانتهائی- به-کاربر انتهائی با استفاده از مکانیسم‌های رمزنگاری و اعتبارسنجی اشاره شده بود.

این نگرانی‌ها کاملاً بجا هستند. برای تأئید این امر به گزارش رَوَند امنیت مرکز هماهنگی تیم پاسخگوئی به فوریت‌های رایانه‌ای(CERT/CC) توجه کنید. شکل ۱-۱الف رَوَند رشد آسیب‌پذیری‌های مرتبط- با- اینترنت گزارش شده به CERT در طی یک دورهٔ ده‌ساله را نشان می‌دهد. اینها شامل ضعف‌های امنیتی موجود در سیستم‌های عامل رایانه‌ها (مثل Windows و Linux) و همچنین آسیب‌پذیری‌های موجود در مسیریاب‌های اینترنت و سایر تجهیزات شبکه‌اند. شکل ۱-۱ب تعداد مشکلات مرتبط- با- امنیت گزارش شده به CERT را نشان می‌دهد. اینها شامل حملات انکار سرویس، جعل IP که در آن مهاجمین بسته‌هائی با آدرس IP جعلی خلق کرده و کاربردهائی را که اعتبارسنجی مبتنی بر IP دارند را فریب می‌دهند، و فرم‌های متنوعی از استراق‌سمع و بوکشیدن بسته‌ها که در آن حمله‌کنندگان اطلاعات انتقال‌یافته از قبیل اطلاعات مربوط به اتصال به سیستم و محتوای پایگاه‌های داده را می‌خوانند، هستند.



شکل ۱-۱الف آسیب‌پذیری‌های گزارش شده بتوسط CERT



شکل ۱-۱ ب حوادث امنیتی گزارش شده بتوسط CERT

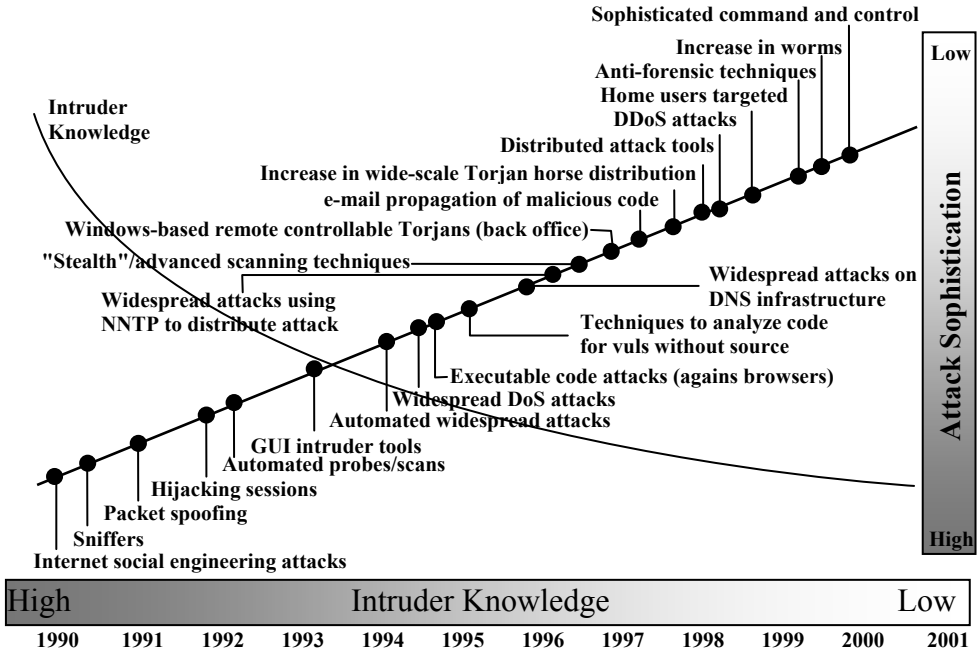
در طول زمان، حملات به اینترنت و سیستم‌های متصل به اینترنت پیچیده‌تر شده در حالی‌که مهارت و معلومات لازم برای انجام حمله کاهش یافته است (شکل ۲-۱). در ضمن، حملات فرم خودکارتری به خود گرفته و می‌توانند صدمات بیشتری را وارد نمایند.

این افزایش در تعداد حملات، با استفاده بیشتر از اینترنت و افزایش پیچیدگی پروتکل‌ها، کاربردها و خود اینترنت مقارن بوده‌اند. زیرساخت‌های حیاتی، بطور روزافزونی برای عملیات خود به اینترنت متکی‌اند. کاربران منفرد نیز به امنیت اینترنت، پست الکترونیک، وِب و کاربردهای مبتنی بر وِب بیش از پیش اتکا دارند. در نتیجه یک محدودهٔ وسیع از تکنولوژی‌ها و ابزارها، برای مقابله با این تهدیدهای فزاینده مورد نیازند. در سطح ابتدائی، الگوریتم‌های رمزنگاری با هدف محرمانگی و اعتبارسنجی اهمیت زیادتری دارند. همچنین طراحان نیازمند تمرکز بر پروتکل‌های مبتنی- بر- اینترنت و آسیب‌پذیری‌های سیستم‌های عامل و کاربردها می‌باشند. این کتاب تمام این زمینه‌های تکنیکی را بررسی می‌نماید.

۱-۲ معماری امنیت OSI

برای تعیین نیازهای امنیتی یک سازمان، و برای ارزیابی و انتخاب خط‌مشی‌ها و محصولات امنیتی مختلف، مدیر مسئول امنیت نیازمند یک روش سیستماتیک برای تشخیص نیازهای امنیتی و مشخص کردن روش‌های تأمین این نیازهاست. این امرخود بقدر کافی در یک محیط متمرکز پردازش داده‌ها پیچیده بوده و در صورت استفاده از شبکه‌های LAN و WAN پیچیدگی آن چندین برابر می‌شود.

توصیه‌نامهٔ X.800 سازمان ITU-T با نام، *معماری امنیت برای OSI*، چنین روش سیستماتیکی را تعریف می‌کند. معماری امنیت OSI برای سازماندهی وظیفهٔ ایجاد امنیت برای مدیران، مفید است. علاوه برآن چون این معماری بصورت یک استاندارد بین‌المللی طراحی شده است، سازندگان رایانه‌ها و تجهیزات ارتباطی، خصوصیات امنیتی محصولات خود را بر اساس تعاریف، سرویس‌ها و مکانیسم‌های این معماری فراهم نموده‌اند.



شکل ۲-۱ رَوَند پیچیدگی حملات و آگاهی‌های مهاجم

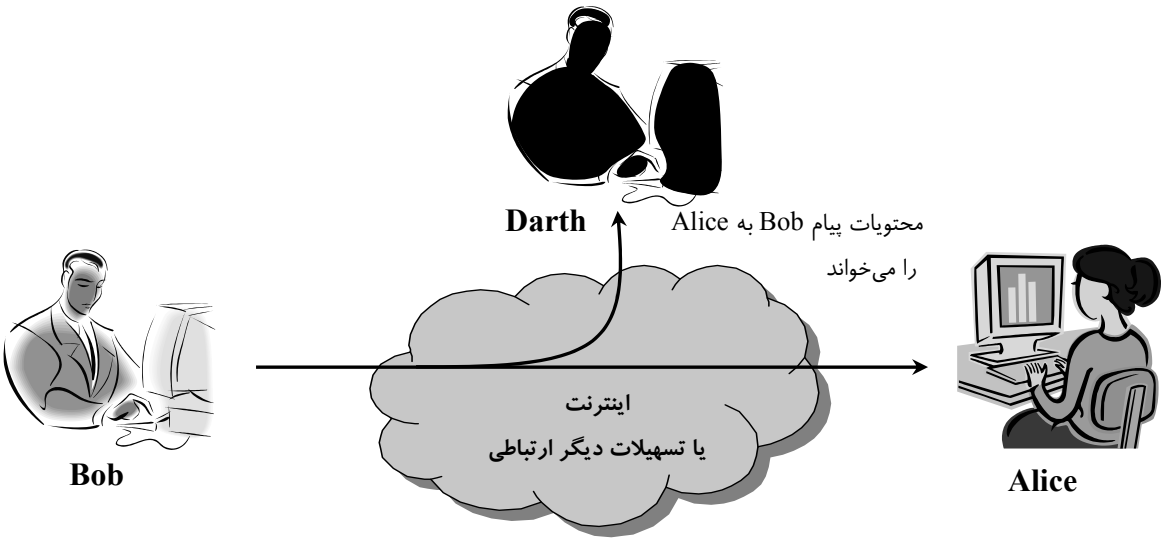
برای مقاصد ما، معماری امنیت OSI یک دید کلی، اگرچه مبهم، از مباحثی که در این کتاب به آنها پرداخته شده است را فراهم می‌سازد. معماری امنیت OSI بر حملات امنیتی، مکانیسم‌ها و سرویس‌ها تمرکز دارد. این عناوین را بطور خلاصه چنین می‌توان تعریف کرد:

- **حملهٔ امنیتی**: هر عملی که امنیت اطلاعات متعلق به یک سازمان را به مخاطره اندازد.
- **مکانیسم امنیتی**: سازوکاری که برای تشخیص، جلوگیری و یا بخودآمدن از یک حملهٔ امنیتی بکار رود.
- **سرویس امنیتی**: سرویسی که امنیت سیستم‌های پردازش اطلاعات و انتقال اطلاعات در یک سازمان را ارتقاء بخشد. هدف این سرویس‌ها مقابله با حملات امنیتی بوده و از یک یا چند مکانیسم امنیتی برای فراهم آوردن سرویس استفاده می‌کنند.

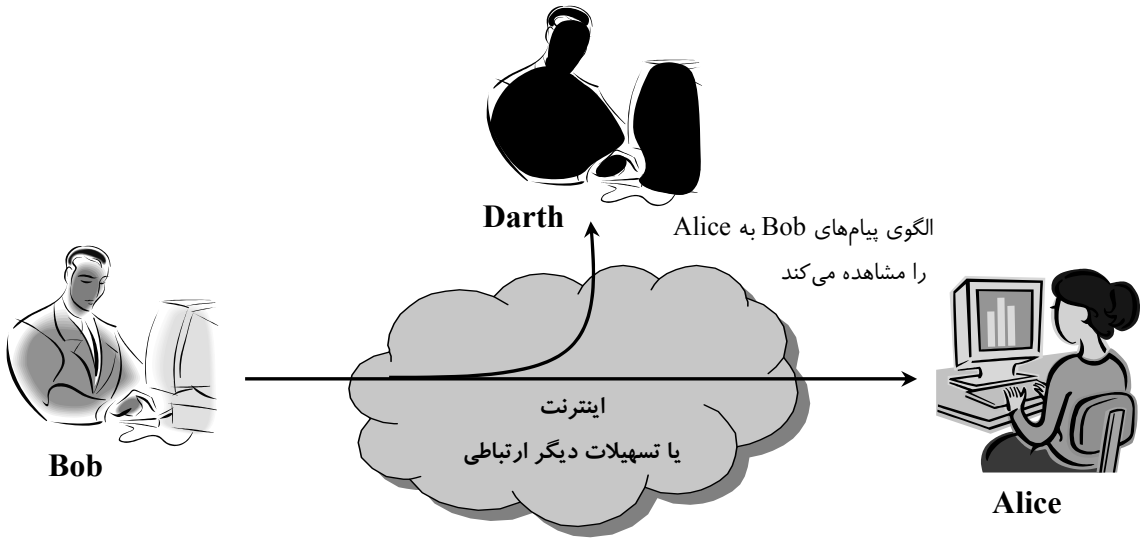
در ادبیات امنیتی، واژه‌های تهدید و حمله مکرراً بکار رفته و تقریباً دارای یک معنی هستند. جدول ۱-۱ تعاریفی که از *واژه‌نامهٔ امنیت اینترنت* RFC 2828 اقتباس شده است را نشان می‌دهد.

جدول ۱-۱ تهدیدها و حمله‌ها (RFC 2828)

تهدید
استعداد بالقوهٔ نقض امنیت در صورت وجود شرایط، قابلیت، عمل، یا اتفاقی که امنیت را مورد مخاطره قرار دهد. یعنی یک تهدید یک خطر احتمالی است که ممکن است از یک نقطهٔ آسیب‌پذیر امنیتی سوءاستفاده نماید.
حمله
هجومی بر امنیت سیستم است که از یک تهدید هوشمند سرچشمه می‌گیرد. یعنی عملی هوشمندانه است که تلاشی زیرکانه برای حمله به سرویس‌های امنیتی و نقض سیاست‌های امنیتی سیستم دارد.



(الف) افشای محتویات پیام



(ب) تحلیل ترافیک

شکل ۳-۱ حملات غیرفعال

۳-۱ حملات امنیتی

یک روش مناسب برای دسته‌بندی حملات امنیتی که هم در X.800 و هم در RFC 2828 استفاده شده است، تقسیم این حملات به دو دستهٔ *حملات غیرفعال* و *حملات فعال* می‌باشد. یک حملهٔ غیرفعال تلاش دارد تا اطلاعات سیستم را به دست آورده و یا از آن استفاده کند ولی روی منابع سیستم تأثیر نمی‌گذارد. یک حملهٔ فعال سعی دارد تا منابع سیستم را تغییر داده و یا بر عملیات آن تأثیر بگذارد.

حملات غیرفعال

حملات غیرفعال دارای ماهیت استراق سمع و یا شنود اطلاعات انتقال یافته است. هدف دشمن در این نوع حمله، دست‌یابی به اطلاعات است. دو نوع حملهٔ غیرفعال، یکی افشای محتویات پیام و دیگری تحلیل ترافیک است.

افشای محتویات پیام را می‌توان بسهولت درک کرد (شکل ۳-۱الف). یک مکالمهٔ تلفنی، یک پیام پست الکترونیک، و یک فایل انتقال یافته ممکن است شامل اطلاعات حساس و یا محرمانه باشند. علاقه‌مندیم که از دست‌یابی دشمن به این اطلاعات جلوگیری نمائیم.

نوع دیگر حملهٔ غیرفعال، **تحلیل ترافیک** است (شکل ۳-۱ب). فرض کنید با توسل به روشی محتویات پیام و یا سایر اطلاعات ترافیکی را طوری تغییر داده‌ایم که دشمنان، حتی اگر پیام را سرقت کنند، نتوانند اطلاعات آن را استخراج نمایند. تکنیک معمول برای این کار رمزنگاری است. ولی حتی اگر حفاظت رمزنگاری را نیز در جای خود داشته باشیم، یک دشمن بازهم ممکن است بتواند الگوی این پیام‌ها را کشف کند. دشمن می‌تواند محل و هویت طرفین ارتباط را تعیین کرده و از تعداد و طول پیام‌هائی که بین آنها ردوبدل می‌شود، آگاه شود. این اطلاعات ممکن است در حدس ماهیت ارتباطی که در حال انجام است مفید باشد.

تشخیص حملات غیرفعال بسیار مشکل است زیرا تأثیری روی خود داده‌ها نمی‌گذارند. معمولاً ترافیک پیام با روند عادی ارسال و دریافت شده و نه فرستنده و نه گیرنده از اینکه طرف سومی پیام را خوانده و یا الگوی ترافیک را ملاحظه کرده است مطلع نمی‌شوند. با وجود این معقول است که از موفقیت چنین حملاتی، معمولاً با رمزنگاری، جلوگیری کرد. بنابراین برای مقابله با حملات غیرفعال تأکید بر پیش‌گیری، بجای تشخیص، است.

حملات فعال

حملات فعال شامل ایجاد تغییرات در جریان دیتا و یا خلق جریان جدیدی از داده‌هاست و می‌توان آنها را به چهار دسته تقسیم کرد: نقاب‌دار، بازخوانی، تغییر پیام و انکار سرویس.

یک حملهٔ **نقاب‌دار** وقتی صورت می‌پذیرد که شخصی یا واحدی وانمود کند که شخص یا واحد دیگری است (شکل۴-۱الف). یک حملهٔ نقاب‌دار معمولاً با حملهٔ فعال دیگری همراه است. بعنوان مثال، دنباله‌های اعتبارسنجی می‌توانند دزدیده شده و پس از این‌که یک عمل اعتبارسنجی معتبر به پایان رسید، بازخوانی شوند و بدین ترتیب به یک واحد مجاز که دارای سطح دست‌یابی پائین‌تری است اجازه دهد تا با جعل هویت واحد دیگری که دارای سطح دست‌یابی بالاتری است، امتیازات بیشتری کسب کند.

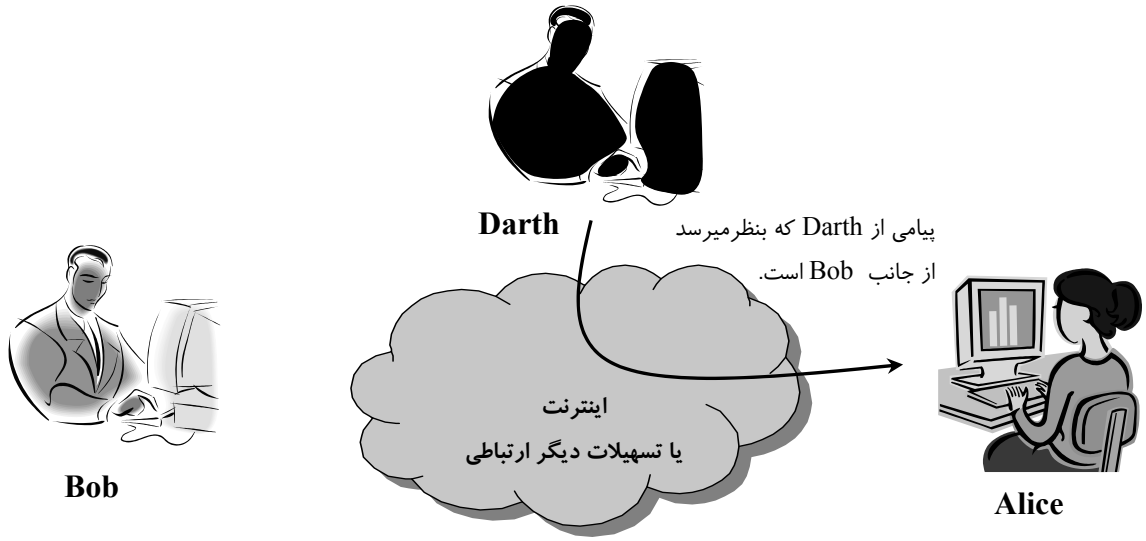
حملهٔ **بازخوانی** شامل دزدیدن غیرفعال واحدهای دیتا و ارسال مجدد آنها با تأخیر، برای ایجاد یک اثر مخرب است (شکل ۴-۱ب).

تغییر پیام بسادگی دارای این معنی است که بخشی از یک پیام قانونی تغییر داده شود، یا این‌که پیام‌ها تأخیر یافته یا نظم آنها برهم زده شود تا نهایتاً باعث اثری غیرمجاز گردند (شکل۴-۱ج). مثلاً پیام " به آقای حمید حمیدی اجازه دهید تا فایل حساب‌های محرمانه را مشاهده کند" به پیام "به آقای مجید مجیدی اجازه دهید تا فایل حساب‌های محرمانه را مشاهده کند" تغییر می‌یابد.

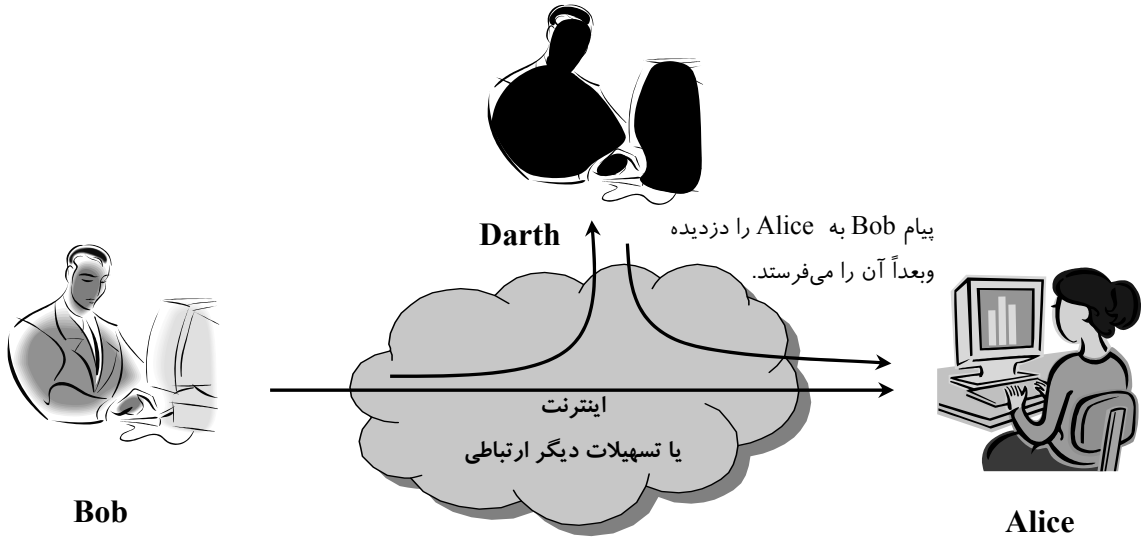
انکار سرویس مانع کارکرد نرمال تجهیزات شده و یا از مدیریت تسهیلات ارتباطی جلوگیری می‌نماید (شکل ۴-۱د). این حمله ممکن است هدف معینی را نشانه بگیرد. مثلاً واحدی ممکن است تمام پیام‌هائی را که برای یک مقصد بخصوص

ارسال می‌شوندحذف کند(مثل بازرسی امنیتی). صورت دیگری از انکار سرویس، ایجاد اختلال در تمام شبکه است که این کار یا با ایجاد خرابی در شبکه و یا با ارسال پیام‌های بسیار زیاد به شبکه بمنظور ایجاد اختلال در عملکرد آن صورت می‌پذیرد.

حملات فعال دارای مشخصاتی خلاف حملات غیرفعال هستند. در حالی‌که تشخیص حملات غیرفعال مشکل است ولی روش‌هائی برای جلوگیری از موفقیت آنها موجود می‌باشد. برعکس، جلوگیری از حملات فعال کاری بس دشوار است زیرا نیاز به محافظت فیزیکی تمام تسهیلات و مسیرهای ارتباطی در تمام زمان‌ها دارد. بجای این کار، هدف تشخیص این حملات و رفع مشکلات و یا تأخیرهائی است که این حملات ممکن است در شبکه ایجاد نمایند. چون تشخیص، خود دارای اثر بازدارندگی است، این کار ممکن است به جلوگیری از حملات نیز کمک کند.

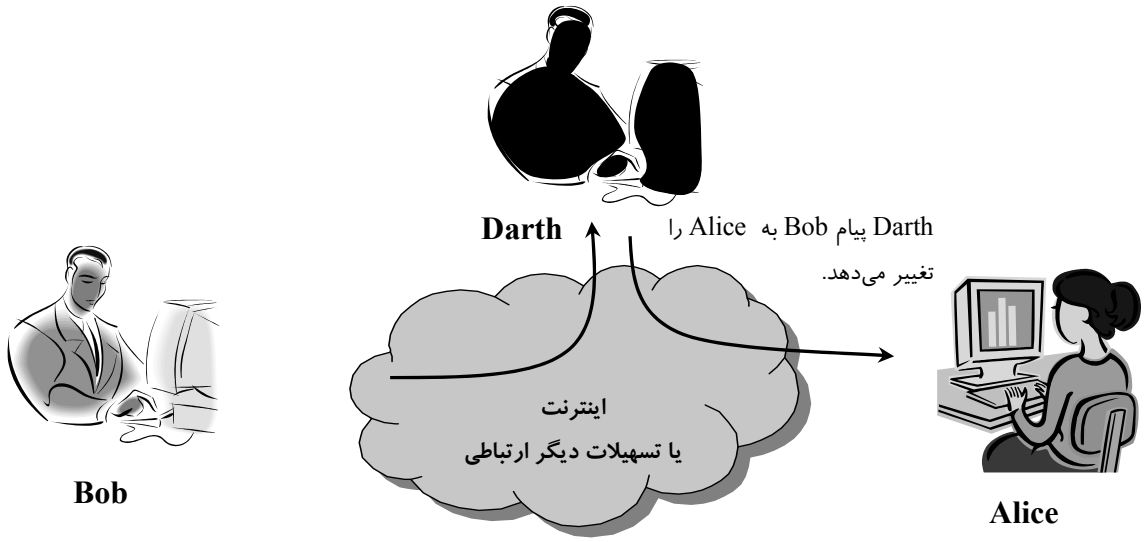


(الف) نقاب‌دار

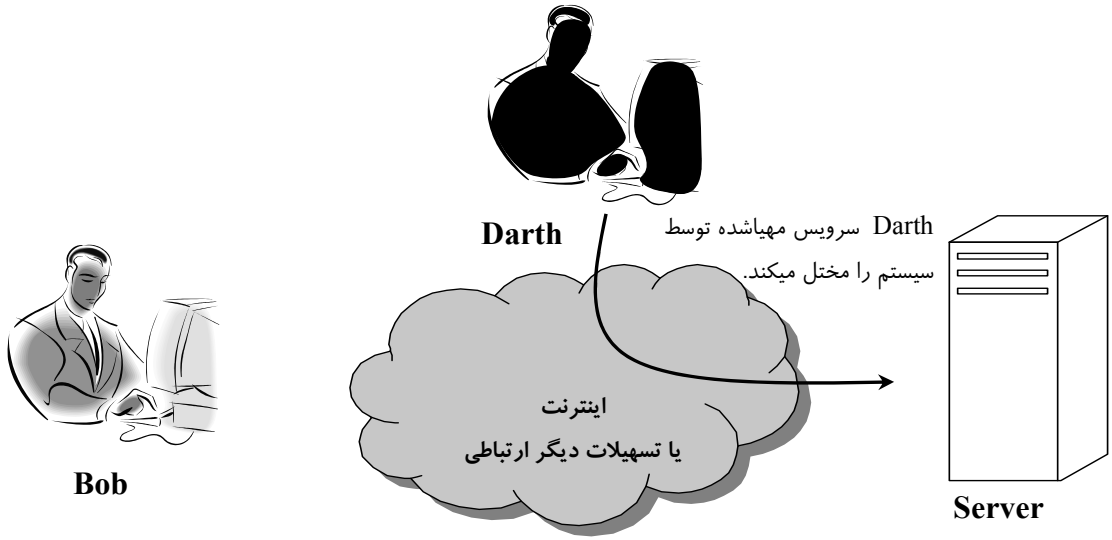


(ب) بازخوانی

شکل ۴-۱ حملات فعال



(ج) تغییر پیام



(د) انکار سرویس

شکل ۴-۱ حملات فعال (ادامه شکل قبل)

۴-۱ سرویس‌های امنیتی

X.800 یک سرویس امنیتی را بعنوان سرویسی تعریف می‌کند که بتوسط یک لایهٔ پروتکلی سیستم‌های باز ارتباطی فراهم شده و امنیت کافی برای سیستم‌ها و یا انتقال داده‌ها را فراهم می‌سازد. شاید RFC 2828 تعریف روشن‌تری را ارائه کند که چنین است: سرویس امنیتی یک سرویس ارتباطی، و یا پردازشی است که بتوسط یک سیستم ایجاد شده تا نوع تعریف شده‌ای از حفاظت را برای منابع سیستم بوجود آورد. سرویس‌های امنیتی، خط‌مشی‌های امنیتی را از طریق مکانیسم‌های امنیتی پیاده‌سازی می‌کنند.

X.800 این سرویس‌ها را به پنج گروه و چهارده سرویس مشخص تقسیم می‌کند (جدول ۲-۱). هریک از این گروه‌ها را بنوبت بررسی می‌کنیم.

اعتبارسنجی

سرویس اعتبارسنجی مسئول اطمینان یافتن از این است که یک ارتباط معتبر است. در مورد یک پیام تنها، مانند یک سیگنال هشداردهنده یا آلارم، وظیفهٔ سرویس اعتبارسنجی این است که به گیرندگان پیام اطمینان دهد که این سیگنال واقعاً از منبعی که ادعا دارد سرچشمه گرفته است. در مورد یک تعامل دائمی، همانند اتصال یک پایانه به یک رایانه، موضوع دو جنبه دارد. اول این که در هنگام برقراری ارتباط، سرویس اعتبارسنجی به طرفین ارتباط اطمینان دهد که طرف مقابل معتبر بوده و هریک از طرفین واقعاً همانی هستند که ادعا می‌کنند. دوم این که سرویس اعتبارسنجی بایستی تضمین کند که اتصال بین دو کاربر در اشغال فرد ثالثی که بتواند خود را بجای هریک از طرفین جازده و ارسال و دریافت غیرمُجازی را ایجاد نماید، درنیامده است.

دو سرویس اعتبارسنجی مشخص در استاندارد تعریف شده‌اند:

- **اعتبارسنجی واحد نظیر**: برای تأئید هویت یک واحد نظیر(peer) در یک مجتمع رایانه‌ای بکار می‌رود. دو واحد را نظیر هم خوانند اگر آنها در دو سیستم مختلف در پروتکل یکسانی پیاده‌سازی شوند. مثلاً مدول‌های TCP در دو سیستم ارتباطی، نظیر هم هستند. استفاده از این سرویس در هنگام برقراری ارتباط و یا در خلال انتقال داده‌هاست. این سرویس تلاش می‌کند تا این اطمینان را فراهم سازد که یک واحد خود را بجای واحد دیگر جانزده و یا یک ارتباط قدیمی را بازخوانی نکرده باشد.
- **اعتبارسنجی منبع دیتا**: برای تأئید هویت منبع یک واحد دیتا بکار می‌رود. حفاظتی در برابر تکرار و یا تغییر داده‌ها ایجاد نمی‌کند. این نوع سرویس از کاربردهائی همانند پست الکترونیک که در آنها هیچ تعاملی بین واحدهای مرتبط وجود ندارد، حمایت می‌کند.

کنترل دست‌یابی

در مقولهٔ امنیت شبکه، کنترل دست‌یابی به مفهوم قابلیت محدودکردن و کنترل دست‌یابی به سیستم‌های میزبان و کاربردها از طریق پیوند ارتباطی است. برای حصول این امر، هر واحد که تمایل به دست‌یابی به سیستم یا کاربردی را دارد بایستی اول شناسائی و یا اعتبارسنجی گردد تا حق دست‌یابی مختص خودش به او داده شود.

محرمانگی داده‌ها

محرمانگی عبارت از حفاظت اطلاعات انتقال‌یافته در برابر حملات غیرفعال است. در رابطه با محتویات یک انتقال دیتا چندین سطح حفاظت را می‌توان تعریف کرد. وسیع‌ترین سرویس، تمام دیتای انتقال‌یافته بین دو کاربر در طول زمان را محافظت می‌کند. مثلاً وقتی یک اتصال TCP بین دو سیستم برقرار می‌شود، این حفاظت وسیع از برملا شدن هرگونه داده کاربر روی اتصال TCP جلوگیری می‌کند. شکل ضعیف‌تر سرویس این است که حفاظت فقط از یک پیام و یا حتی بخش‌های مشخصی از یک پیام صورت پذیرد. این سرویس پالایش شده کمتر از سرویس وسیع مفید بوده و حتی ممکن است به پیچیدگی و هزینهٔ بیشتری منجر شود.

جدول ۲-۱ سرویس‌های امنیتی(X.800)

اعتبارسنجی (AUTHENTICATION)

اطمینان از اینکه واحد ارتباطی همان است که ادعا می‌کند.

اعتبارسنجی واحد نظیر(Peer Entity Authentication)
در رابطه با یک ارتباط منطقی تعریف شده تا نسبت به هویت واحدهای مرتبط ایجاد اطمینان نماید.

اعتبارسنجی منبع دیتا (Data-Origin Authentication)
در یک انتقال غیراتصالی، این اطمینان را ایجاد می‌کند که منبع دیتای دریافت شده همان است که ادعا می‌کند.

کنترل دست‌یابی (ACCESS CONTROL)

ممانعت از استفادهٔ غیرمجاز از یک منبع (یعنی این سرویس کنترل می‌کند که چه کسی می‌تواند به یک منبع دست یافته، تحت چه شرایطی این دست‌یابی می‌تواند انجام شود، و آنهائی که دست‌یابی پیدا کنند مجاز به انجام چه کارهائی هستند).

محرمانگی داده‌ها (DATA CONFIDENTIALITY)

حفاظت از داده‌ها در برابر افشای غیرمجاز

محرمانگی اتصالی (Connection Confidentiality)
حفاظت از تمام دادهٔ کاربر در طول اتصال

محرمانگی غیراتصالی (Connectionless Confidentiality)
حفاظت از تمام دادهٔ کاربر در یک بلوک منفرد

محرمانگی میدان انتخابی (Selective-Field Confidentiality)
محرمانگی میدان‌های انتخاب شده دادهٔ کاربر در تمام یک اتصال و یا در یک بلوک دیتا

محرمانگی جریان ترافیک (Traffic-Flow Confidentiality)
حفاظت از اطلاعاتی که ممکن است از مشاهدهٔ جریان ترافیک داده‌ها بدست آید.

صحت داده‌ها (DATA INTEGRITY)

اطمینان از اینکه دادهٔ دریافت شده دقیقاً همانی است که بتوسط یک واحد معتبر ارسال شده است (یعنی تغییر نیافته، اضافه نشده، حذف نشده و بازخوانی نشده است).

صحت اتصالی با بازیابی (Connection Integrity with Recovery)
صحت کل دادهٔ کاربر روی یک اتصال را تأئید کرده و هرگونه تغییر، حذف، اضافه، و یا بازخوانی داده‌ها در یک دنبالهٔ کامل دیتا را تشخیص می‌دهد. بازیابی داده نیز مورد نظر است.

صحت اتصالی بدون بازیابی(Connection Integrity without Recovery)
همانند مورد بالاست، ولی تنها تشخیص و نه بازیابی مورد توجه است.

صحت اتصالی میدان‌های انتخابی
(Selective-Field Connection Integrity)
صحت میدان‌های انتخاب‌شده‌ای از دادهٔ کاربر در یک بلوک را بعهده داشته و تعیین می‌کند که آیا میدان‌های انتخاب‌شده تغییریافته، حذف یا اضافه شده و یا بازخوانی شده‌اند.

صحت غیراتصالی (Connectionless Integrity)
صحت یک بلوک دیتای منفرد را تعیین نموده و ممکن است قدرت تشخیص تغییر داده را داشته باشد. علاوه برآن توان محدودی از تشخیص بازخوانی را نیز می‌تواند تأمین کند.

صحت غیراتصالی میدان‌های انتخابی
(Selective-Field Connectionless Integrity)
صحت میدان‌های انتخاب شده‌ای در یک بلوک دیتای غیراتصالی را تعیین می‌کند. مشخص می‌کند که آیا میدان‌های مورد نظر تغییر یافته‌اند.

عدم انکار (NONREPUDIATION)

در برابر انکار یکی از طرفین ارتباط نسبت به انکار تمام و یا بخشی از ارتباط، ایجاد حفاظت می‌کند.

عدم انکار، مبدأ (Nonrepudiation, Origin)
اثبات اینکه پیام بتوسط طرف اصلی ارسال شده است.

عدم انکار، مقصد (Nonrepudiation, Destination)
اثبات اینکه پیام بتوسط طرف اصلی دریافت شده است.

جنبهٔ دیگر محرمانگی، حفاظت جریان ترافیک در برابر تجزیه و تحلیل دشمن است. لازمهٔ این کار این است که یک مهاجم نتواند منبع، مقصد، تواتر، طول و یا سایر مشخصه‌های ترافیکی یک تسهیلات ارتباطی را مشاهده نماید.

صحت داده‌ها

همانند محرمانگی، کنترل صحت و یا اصالت داده‌ها می‌تواند به جریان دائمی پیام‌ها، به یک پیام منفرد و یا به میدان‌های انتخاب شده از یک پیام اعمال گردد. بازهم مفیدترین و سرراست‌ترین روش، حفاظت از کل جریان داده‌هاست.

یک سرویس صحت اتصال‌گرا، سرویسی که با جریان پیوستهٔ پیام‌ها سروکار دارد، بایستی این اطمینان را ایجاد کندکه پیام‌ها همانطور که ارسال می‌شوند دریافت گردند، بدون اینکه مجدداً تکرار شده، چیزی به آنها اضافه شده، تغییر یافته، نظم آنها بهم خورده و یا بازخوانی شده باشند. تخریب داده‌ها نیز تحت همین سرویس قرار دارد. بنابراین سرویس صحت با گرایش اتصالی هم تغییر داده‌ها و هم انکار سرویس را مورد خطاب قرار می‌دهد. از سوی دیگر یک سرویس صحت غیراتصالی آن است که تنها با پیام‌های منفرد، بدون توجه به محدودهٔ وسیع آنها، سروکار داشته و فقط در برابر تغییر محتویات پیام حفاظت ایجادکند.

می‌توان بین سرویس‌های با بازیابی و بدون بازیابی تمایز قائل شد. چون سرویس صحت مربوط به حملات فعال است، جنبهٔ تشخیص آنها و نه جنبهٔ جلوگیری از آنها دارای اهمیت است. اگر در صحت دیتا خللی مشاهده گردد، سرویس مربوط ممکن است تنها این خلل را گزارش نماید و لازم باشد تا بخش دیگری از نرم‌افزار و یا نوعی دخالت انسانی مشکل را حل کند. از سوی دیگر مکانیسم‌هائی نیز وجود دارند که علاوه بر تشخیص عدم صحت به حل مشکل نیز کمک می‌کنند. قراردادن مکانیسم‌های بازیابی خودکار معمولاً انتخاب‌های پرجاذبه‌تری هستند.

عدم انکار

عدم انکار، چه فرستنده و چه گیرنده را از انکار یک پیام ارسال شده مانع می‌شود. بنابراین وقتی پیامی ارسال می‌شود، گیرنده پیام می‌تواند اثبات کند که حتماً همان فرستندهٔ ذکرشده، پیام را ارسال کرده است. بطریق مشابه وقتی پیامی دریافت می‌گردد، فرستندهٔ پیام می‌تواند اثبات کند که حتماً همان گیرندهٔ ذکرشده، پیام را دریافت کرده است.

سرویس قابلیت دسترسی

هم X.800 و هم RFC 2828 قابلیت دسترسی (availability) را خاصیت یک سیستم و یا یک منبع می‌دانند که در صورت تقاضا از سوی یک واحد مجاز و بر اساس مشخصه‌های عملکرد، سیستم و منابع آن آمادهٔ سرویس‌دهی باشند (یعنی یک سیستم وقتی قابل دسترس است که هروقت کاربران بخواهند، بتواند بر اساس طراحی خود به آنان ارائهٔ سرویس نماید). حملات مختلفی می‌توانند باعث کم شدن قابلیت دسترسی شوند. از بعضی از این حمله‌ها می‌توان با چاره‌جوئی‌های خودکار مثل اعتبارسنجی و رمزنگاری جلوگیری کرد در حالیکه بازیابی از برخی دیگر در یک سیستم گسترده، نیاز به نوعی دخالت فیزیکی دارد.

X.800 قابلیت دسترسی را بعنوان خاصیتی مرتبط با سرویس‌های مختلف امنیتی می‌شناسد. با وجود این منطقی است که به دنبال یک سرویس مخصوص قابلیت دسترسی نیز باشیم. این سرویس، نگرانی‌های امنیتی مرتبط با حملات انکار سرویس را مورد توجه قرار می‌دهد. این سرویس اتکاء به مدیریت منظم و کنترل منابع سیستم داشته، و بنابراین وابسته به سرویس کنترل دست‌یابی و سایر سرویس‌های امنیتی است.

۱-۵ مکانیسم‌های امنیتی

جدول ۳-۱ لیست مکانیسم‌های امنیتی تعریف شده در X.800 را نشان می‌دهد. همانطور که می‌توان دید، مکانیسم‌ها به دودسته، یکی آنهائی که در یک لایهٔ پروتکلی خاص قرار دارند و دیگری آنهائی که مختص لایهٔ خاص و یا سرویس امنیتی خاصی نیستند تقسیم می‌شوند. این مکانیسم‌ها در محل مناسب خود در این کتاب مورد بحث قرار خواهند گرفت و بنابراین فعلاً وارد جزئیات آنها نمی‌شویم. فقط در مورد تعریف قابلیت رمزنگاری، به نکته‌ای اشاره می‌کنیم. X.800 بین مکانیسم‌های رمزنگاری برگشت‌پذیر و مکانیسم‌های رمزنگاری برگشت‌ناپذیر تفاوت قائل است. یک مکانیسم رمزنگاری برگشت‌پذیر بسادگی یک الگوریتم رمزنگاری است که اجازه می‌دهد تا داده‌ها به رمز درآمده و متعاقباً از رمز خارج شوند. مکانیسم‌های رمزنگاری برگشت‌ناپذیر شامل الگوریتم‌های درهم‌سازی و کُدهای اعتبارسنجی پیام بوده که در کاربردهای امضاء دیجیتال و اعتبارسنجی پیام بکار می‌روند.

جدول ۴-۱ که بر اساس X.800 بنا شده است رابطهٔ بین سرویس‌های امنیتی و مکانیسم‌های امنیتی را نشان می‌دهد.

جدول ۳-۱ مکانیسم‌های امنیتی (X.800)

مکانیسم‌های امنیتی مخصوص	
ممکن است در لایه پروتکلی مناسب قرار داده شود تا بعضی از سرویس‌های امنیتی OSI را فراهم نماید.	**کنترل مسیریابی (Routing Control)** انتخاب مسیرهای فیزیکی امن برای بعضی داده‌ها را امکان‌پذیر کرده و اجازه می‌دهد تا در صورت بروز یک تهدید امنیتی، مسیر تعویض شود.
رمزنگاری (Encipherment) استفاده از الگوریتم‌های ریاضی، تا داده‌ها را به شکل غیرقابل فهمی درآورد. تبدیل دیتا و بازیابی آیندهٔ آن بستگی به الگوریتم و احتمالاً یک یا چند کلید رمز دارد.	**ثبت سند (Notarization)** استفاده از یک طرف ثالث معتمد برای اطمینان یافتن از خصوصیات یک مبادلهٔ دیتا.
امضاء دیجیتال (Digital Signature) دیتای وصل شده به/ یا تبدیل رمزنگاری شدهٔ یک واحد دیتا که به یک دریافت‌کنندهٔ واحد دیتا اجازه می‌دهد تا منبع دیتا و صحت دیتا را اثبات کرده و از تقلب جلوگیری نماید.	**مکانیسم‌های امنیتی فراگیر** مکانیسم‌هائی که به سرویس امنیتی و یا یک لایهٔ پروتکلی خاص OSI وابسته نیستند.
کنترل دست‌یابی (Access Control) مکانیسم‌های متنوعی که حق دست‌یابی به منابع را قانون‌مند می‌سازند.	**عملکرد مطمئن (Trusted Functionality)** اینکه دیتا موافق با شرایط خاصی صحیح باشد (مثلاً بر اساس یک خط‌مشی امنیتی)
صحت دیتا (Data Integrity) مکانیسم‌های متنوعی که از آنها برای اطمینان از صحت یک واحد دیتا و یا دنباله‌ای از واحدهای دیتا استفاده می‌شود.	**برچسب امنیتی (Security Label)** نشانه‌ای که به یک منبع (که ممکن است یک واحد دیتا باشد) وصل می‌گردد تا مشخصه‌های امنیتی آن منبع را نشان دهد.
مبادلهٔ اعتبارسنجی (Authentication Exchange) مکانیسمی با هدف اطمینان یافتن از هویت یک واحد از طریق مبادلهٔ اطلاعات.	**تشخیص وقایع (Event Detection)** تشخیص پیشامدهای مرتبط با امنیت.
لابه‌لا کردن ترافیک (Traffic Padding) واردکردن بیت‌ها در شکاف‌های دیتا به منظور خنثی کردن تلاش‌های تحلیل ترافیک.	**ردپای ممیزی امنیتی (Security Audit Trail)** دیتای جمع‌آوری‌شده که بطرز مؤثری برای تسهیل یک ممیزی امنیتی بکار رود. مروری مستقل بر سوابق سیستم و فعالیت‌های آن است.
	بازیابی امنیتی (Security Recovery) مربوط به درخواست از مکانیسم‌ها، همانند رتق و فتق وقایع و عملیات مدیریتی بوده که به بازیابی منتهی شود.

جدول ۴-۱ رابطهٔ بین سرویس‌های امنیتی و مکانیسم‌های امنیتی

سرویس	مکانیسم							
	رمزنگاری	امضاء دیجیتال	کنترل دست‌یابی	صحت داده‌ها	مبادلهٔ اعتبارسنجی	لایه‌لائی ترافیک	کنترل مسیریابی	ثبت‌سند
اعتبارسنجی واحد نظیر	بلی	بلی			بلی			
اعتبارسنجی منبع دیتا	بلی	بلی						
کنترل دست‌یابی			بلی					
محرمانگی	بلی						بلی	
محرمانگی جریان ترافیک	بلی					بلی	بلی	
صحت داده‌ها	بلی	بلی		بلی				
عدم انکار		بلی		بلی				بلی
قابلیت دسترسی				بلی	بلی			

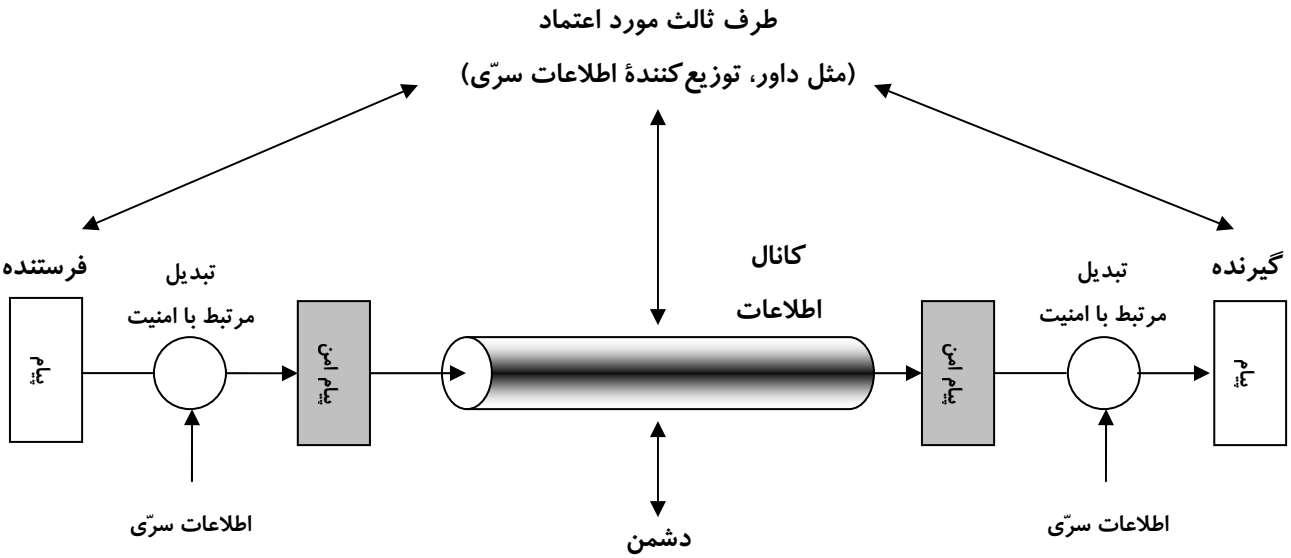
۱-۶ یک مدل برای امنیت شبکه

یک مدل برای بیشتر مطالبی که مورد بحث قرار خواهد گرفت، در حالت بسیار کلی، در شکل ۵-۱ نشان داده شده است. قرار است یک پیام، از یک طرف مکالمه به طرف دیگر، در عرض نوعی اینترنت ارسال گردد. دو طرف مکالمه که *رؤسای* (*principals*) این ارتباط هستند بایستی با یکدیگر همکاری نموده تا این انتقال صورت پذیرد. با تعریف یک مسیر ارتباطی از درون شبکه‌ها، که مبدأ را به مقصد متصل می‌کند، و همکاری در استفاده از پروتکل‌های ارتباطی (مثل TCP/IP)، یک کانال اطلاعاتی منطقی بین دو رئیس ارتباط برقرار می‌شود.

جنبه‌های امنیتی وقتی وارد قضیه می‌گردند که لازم باشد تا انتقال اطلاعات را از یک دشمن که ممکن است تهدیدی برای محرمانگی، اعتبار و غیره ایجاد کند، محافظت کرد. تمام تکنیک‌هائی که امنیت را فراهم می‌سازند دارای دو مؤلفه هستند:

- یک تبدیل مرتبط با امنیت، روی پیامی که قرار است ارسال شود صورت می‌پذیرد. مثال‌های این مورد شامل رمزنگاری پیام است که طوری پیام را درهم می‌ریزد تا قابل خواندن بتوسط دشمن نباشد و همچنین اضافه کردن یک کُد مبتنی بر محتویات پیام که می‌تواند برای تأئید هویت ارسال‌کننده بکار رود.
- نوعی اطلاعات سرّی که بین دو رئیس ارتباط مشترک است و امید می‌رود تا برای دشمن ناشناخته باشد. مثالی در این زمینه یک کلید رمزنگاری است که به‌همراه تبدیل برای درهم ریختن پیام قبل از ارسال، و اصلاح پیام پس از دریافت بکار می‌رود.

یک طرف ثالث قابل اعتماد نیز ممکن است برای انجام انتقال امن اطلاعات مورد نیاز باشد. برای مثال طرف ثالث ممکن است مسئولیت توزیع اطلاعات سرّی به دو رئیس و پنهان نگاه داشتن آن از دید دشمن را بعهده داشته باشد. یا طرف ثالث ممکن است برای داوری اختلافاتی که ممکن است بین دو رئیس در مورد اعتبار یک پیام انتقال یافته رخ دهد مورد نیاز باشد.



شکل ۵-۱ مدل امنیت شبکه

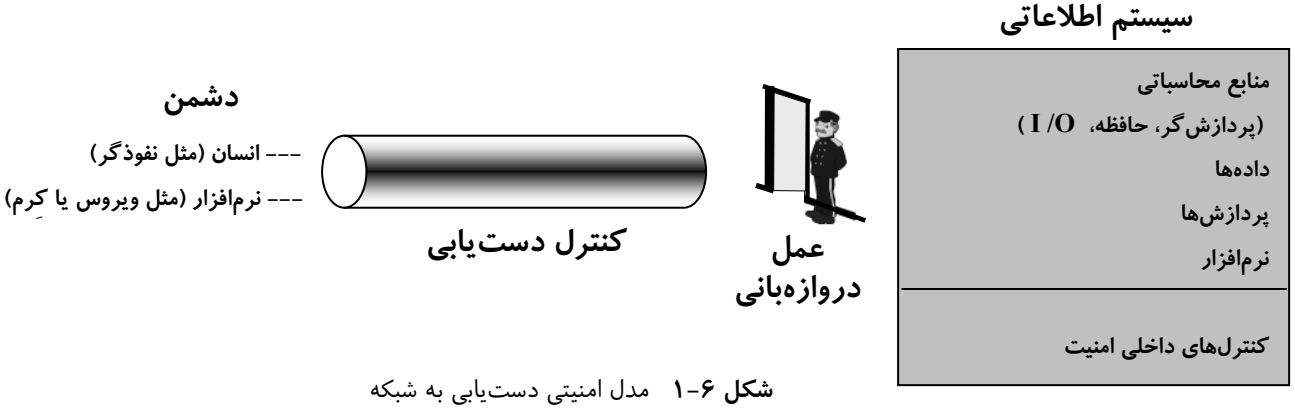
مدل کلّی نشان می‌دهد که در طراحی یک سرویس امنیتی خاص، چهار وظیفهٔ اصلی وجود دارد:

۱- یک الگوریتم برای انجام تبدیل مرتبط با امنیت پیام بایستی طراحی شود. الگوریتم باید چنان باشد که یک دشمن نتواند هدف آن را شکست دهد.
۲- اطلاعات سرّی لازم بتوسط الگوریتم تولید شود.
۳- روش‌هائی برای توزیع و به اشتراک گذاشتن اطلاعات سرّی تعیین گردد.
۴- پروتکلی تعیین شود که دو رئیس از آن استفاده کرده و با بهره‌برداری از الگوریتم امنیتی و اطلاعات سرّی، سرویس امنیتی خاصی را برای طرفین فراهم آورد.

قسمت دوم این کتاب روی مکانیسم‌های امنیتی و سرویس‌هائی که در مدل شکل ۵-۱ قرار دارند متمرکز است. ولی مقوله‌های دیگری نیز در این کتاب مورد بحث قرارگرفته که مرتبط با امنیت بوده ولی دقیقاً در این مدل نمی‌گنجند. یک نمونه از این موارد در شکل ۶-۱ نشان داده شده است که نگرانی‌های مربوط به حفاظت یک سیستم اطلاعاتی از دست‌یابی ناخواسته را نشان می‌دهد. بیشتر خوانندگان با نگرانی‌های ناشی از تلاش هَکرها برای نفوذ به سیستم‌های یک شبکه آشنائی دارند. یک هَکر می‌تواند فردی باشد که بدون نیت سوء، از شکستن دیوارهای امنیتی و ورود غیرمجاز به یک سیستم رایانه‌ای لذت ببرد. یک مهاجم ممکن است یک کارمند ناراضی بوده که بخواهد به سیستم صدمه زده و یا مجرمی باشد که بخواهد از تسهیلات رایانه‌ای برای سوء استفاده مالی استفاده کند (مثل بدست آوردن شمارهٔ کارت‌های اعتباری یا انجام نقل و انتقال غیرقانونی پول).

نوع دیگری از دست‌یابی نامطلوب، قراردادن نوعی منطق در سیستم رایانه‌ای برای بهره‌گیری از نقاط آسیب‌پذیر سیستم بوده که می‌تواند برنامه‌های کاربردی و همچنین برنامه‌های سیستمی مانند ویرایش‌گرها و کامپایلرها را تحت تأثیر قرار دهد. برنامه‌های موذی می‌توانند دو نوع تهدید را بوجود آورند:

- **تهدیدهای دست‌یابی به اطلاعات** که داده‌ها را به نمایندگی از طرف کاربرهائی که نبایستی دست‌یابی به آنها داشته باشند دزدیده و یا تغییر می‌دهند.
- **تهدیدهای سرویس** که از نواقص سرویس‌ها در رایانه‌ها سوء استفاده کرده و مانع استفادهٔ کاربران قانونی از آنها می‌شوند.



شکل ۶-۱ مدل امنیتی دست‌یابی به شبکه

ویروس‌ها و کِرم‌ها دو مثال از حملات نرم‌افزاری می‌باشند. چنین حملاتی می‌تواند از طریق یک دیسکت علیه رایانه صورت پذیرفته و شامل منطقی نامطلوب باشد که در پوشش یک نرم‌افزار مفید پنهان شده است. این حمله‌ها همچنین می‌توانند از طریق شبکه علیه سیستم انجام شوندکه مورد اخیر بیشتر در مقولهٔ امنیت شبکه قرار دارد.

مکانیسم‌های امنیتی که بایستی با دست‌یابی‌های ناخواسته مقابله نمایند به دو دستهٔ بزرگ تقسیم می‌شوند (شکل۶-۱). دستهٔ اول همانند یک دروازه‌بان عمل می‌کنند. اینها شامل روش‌های ورود به سیستم بر اساس استفاده از کلمهٔ عبور و روش‌های بازرسی می‌باشند که برای تشخیص و حذف کِرم‌ها، ویروس‌ها و سایر حملات مشابه بکار می‌روند. ولی اگر یک کاربر ناخواسته و یا نرم‌افزار بداندیش توانست به سیستم دست یابد آنگاه خط دوم دفاعی که شامل موارد متنوع کنترل‌های داخلی اعم از پائیدن فعالیت‌ها و تحلیل اطلاعات ذخیره‌شده می‌باشند وارد عمل شده و تلاش خواهند کرد تا حضور مهاجمین ناخواسته را تشخیص دهند. این مطالب در قسمت سوم این کتاب مورد بحث قرار می‌گیرند

۱-۷ استانداردهای اینترنت و انجمن اینترنت

بسیاری از پروتکل‌هائی که مجموعهٔ پروتکلی TCP/IP را می‌سازند، استاندارد شده و یا در شرف استانداردشدن هستند. با موافقت جهانی، سازمانی بنام انجمن اینترنت (Internet Society) مسئول ایجاد و انتشار این استانداردهاست. انجمن اینترنت یک سازمان حرفه‌ای است که بر نیروهای وسیعی که درگیر کارهای اینترنتی و استانداردسازی هستند، نظارت می‌کند.

این بخش توصیف مختصری از روش‌هائی که برای ایجاد استاندارد مجموعهٔ پروتکلی TCP/IP بکار می‌رود را فراهم می‌سازد.

سازمان‌های اینترنت و انتشارات RFC

انجمن اینترنت کمیتهٔ هماهنگ‌کنندهٔ طراحی، مهندسی، و مدیریت اینترنت است. محدودهٔ پوششی آن، عملیات خود اینترنت و استانداردکردن پروتکل‌هائی است که بتوسط سیستم‌های انتهائی بکار می‌روند. سه سازمان تحت مدیریت انجمن اینترنت، مسئول واقعی کار استانداردها و انتشارات می‌باشند:

- **گروه معماری اینترنت Internet Architecture Board (IAB)**: مسئول تعریف معماری کلی اینترنت بوده و راهنمائی و جهت فعالیت IETF را فراهم می‌آورد.
- **نیروی مهندسی اینترنت Internet Engineering Task Force (IETF)**: بازوی توسعه و مهندسی اینترنت.
- **گروه راهبری مهندسی اینترنت Internet Engineering Steering Group (IESG)**: مسئول مدیریت فنی فعالیت‌های IETF و پیاده‌سازی استانداردهای اینترنت.

گروه‌های کاری که بتوسط IETF بسیج می‌شوند، توسعه واقعی استانداردهای جدید و پروتکل‌های اینترنت را بعهده دارند. عضویت در یک گروه، کاری داوطلبانه است و هر گروه علاقه‌مند می‌تواند در آن شرکت نماید. در جریان تهیهٔ یک مشخصه، یک گروه کاری یک پیش‌نویس از اسناد موجود بعنوان یک پیش‌نویس اینترنت تهیه کرده و آن را در فهرست "Internet Draft" بطور مستقیم روی خط قرار می‌دهد. این سند ممکن است تا شش ماه در محل ذکرشده قرار داشته تا گروه‌های علاقه‌مند بتوانند آن را مطالعه کرده و نظرات خود را ابراز دارند. در خلال این مدت، IESG ممکن است انتشار این سند را بعنوان یکRFC (Request for Comment) تصویب کند. اگر این پیش‌نویس ظرف یک دورهٔ شش ماهه، فرم

RFC بخود نگرفت از فهرست خارج خواهد شد. گروه کاری در پی آن ممکن است یک نسخهٔ اصلاح و دستکاری شدهٔ آن را انتشار دهد.

IETF مسئول انتشار RFCها با تصویب IESG است. RFCها یادداشت‌های کاری کمیتهٔ توسعه و مهندسی اینترنت است. یک سند در این سری ممکن است هر موضوعی در رابطه با ارتباطات رایانه‌ای بوده و می‌تواند هرچیزی از گزارش یک ملاقات تا مشخصات یک استاندارد باشد.

کار IETF به هشت شعبه تقسیم شده که هرشعبه دارای یک مدیر بوده و خود شامل گروه‌های کاری بسیار است. جدول ۵-۱ شعبات IETF و وظایف آنها را نشان می‌دهد.

رَوَند استانداردسازی

تصمیم‌گیری راجع به اینکه کدام RFC یک استاندارد اینترنت شود، بتوسط IESG و بر اساس توصیهٔ IETF صورت می‌پذیرد. برای اینکه یک مشخصه بصورت استاندارد درآید، بایستی دارای شرایط زیر باشد:

- پایدار بوده و خوب درک شده باشد.
- از نظر تکنیکی، رقیب تکنیک‌های دیگر باشد.
- دارای صور پیاده‌سازی متعدد، مستقل و متعامل با تجارب عملیاتی قابل توجه باشد.
- از حمایت عمومی چشم‌گیری برخوردار باشد.
- بطور قابل ملاحظه‌ای در بخشی و یا تمام بخش‌های اینترنت مفید باشد.

جدول ۵-۱ عرصه‌های IETF

عرصهٔ IETF	موضوع	نمونهٔ گروه‌های کاری
عمومی	روش‌های کاری IETF	Policy Framework Process for Organization of Internet Standards
کاربردها	کاربردهای اینترنت	Web- related protocols(HTTP) EDI- Internet integration LDAP
اینترنت	زیرساخت اینترنت	IPv6 PPP extensions
عملیات و مدیریت	استانداردها و تعاریف مرتبط با عملیات شبکه	SNMPv3 Remote Network Monitoring
مسیریابی	پروتکل‌ها و مدیریت اطلاعات مسیریابی	Multicast routing OSPF QoS routing
امنیت	پروتکل‌های امنیتی و فن‌آوری‌ها	Kerberos IPsec X.509 S/MIME TLS
حمل و نقل	پروتکل‌های لایه حمل‌ونقل	Differentiated services IP telephony NFS RSVP
سرویس‌های کاربران	روش‌های ارتقاء کیفیت اطلاعات برای کاربران اینترنت	Responsible Use of the Internet User Services FYI documents

اختلاف کلیدی بین این شرایط و آنهائی که از طرف ITU بصورت استاندارد بین‌المللی مطرح می‌شوند این است که در اینجا، تأکید بر تجارب عملیاتی است.

سمت چپ شکل ۷-۱ قدم‌هائی را نشان می‌دهد که *مسیر استاندارد* نامیده می‌شوند و یک مشخصه بایستی آنها را پیموده تا بصورت یک استاندارد درآید. این تحول در RFC 2026 تعریف شده است. این قدم‌ها شامل میزان فزاینده‌ای از بازرسی‌های دقیق و آزمایش‌های متنوع است. در هر قدم، IETF بایستی توصیه‌هائی برای رشد پروتکل را فراهم آورد و IESG بایستی آن را تصویب کند. عمل وقتی آغاز می‌شود که IESG نسخهٔ منتشرشدهٔ پیش‌نویس سند را بعنوان یک RFC پیشنهادشده برای استاندارد بپذیرد.

خانه‌های سفید در شکل ۷-۱ نمایش حالات موقت بوده که بایستی برای حداقل زمان ممکن اشغال شوند. با وجود این یک سند بایستی حداقل شش ماه بصورت یک استاندارد پیشنهادشده، و حداقل چهارماه بصورت یک پیش‌نویس استاندارد، در حالت انتظار قرار داشته باشد تا زمان کافی برای تجدیدنظر و پیشنهادها موجود باشد. خانه‌های خاکستری نمایش‌گر حالات طولانی بوده که ممکن است سال‌ها طول بکشند.

برای اینکه یک مشخصه به حالت پیش‌نویس استاندارد ارتقاء یابد، حداقل بایستی دو پیاده‌سازی مستقل و متعامل از آن با تجربیات عملی کسب شده وجود داشته باشد.

پس از اینکه مشخصه بصورت قابل توجهی پیاده‌سازی شده و تجربیات عملی از آن بدست آمده باشد، آنگاه آن مشخصه ممکن است به سطح یک استاندارد اینترنت ارتقاء یابد. در این مرحله، به مشخصه یک شمارهٔ STD و همچنین یک شمارهٔ RFC داده می‌شود.

بالاخره زمانی که یک پروتکل دیگر به درد نخورد، حالت تاریخی به آن نسبت داده می‌شود.

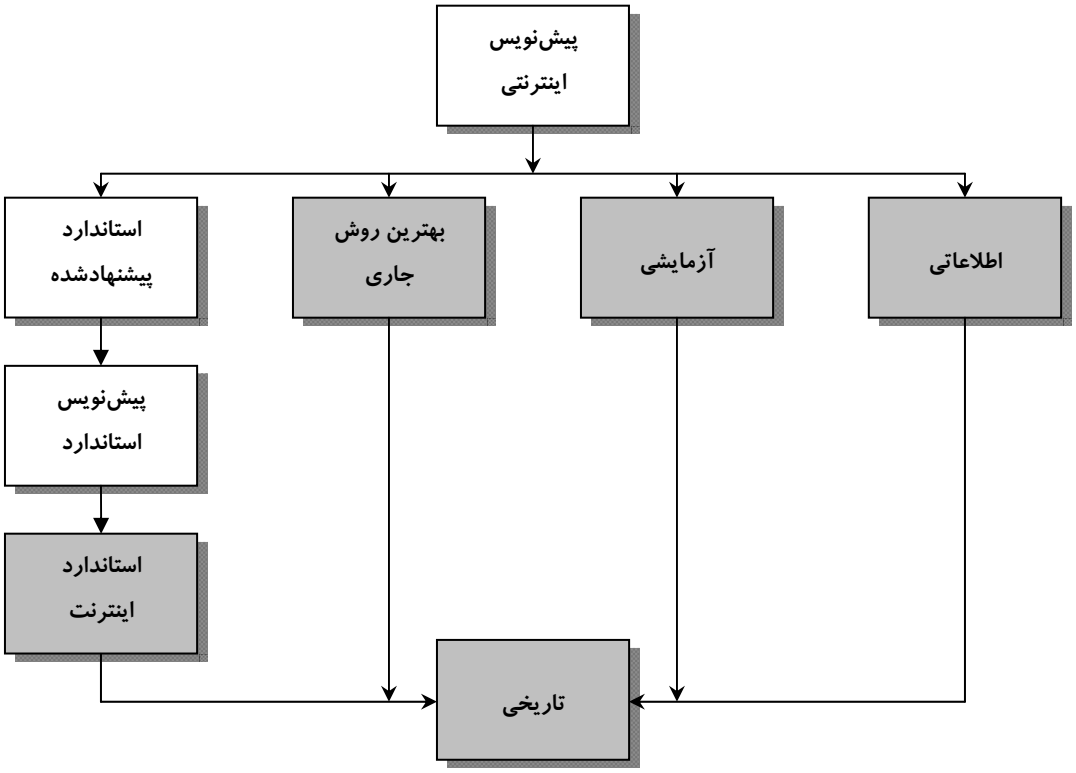
دسته‌بندی استانداردهای اینترنتی

تمام استانداردهای اینترنت در یکی از دو گروه زیر قرار دارند:

- **مشخصات فنی (Technical specification (TS)**: یک TS، یک پروتکل، سرویس، روش، قرارداد و یا فرمت را تعریف می‌کند. بیشتر استانداردها از نوع TS هستند.
- **گزارش قابلیت عملیاتی (Applicability statement (AS)**: یک AS مشخص می‌سازد که چگونه و تحت چه شرایطی یکی و یا بیش از یکی از TSها می‌توانند برای حمایت از یک قابلیت اینترنتی بکار گرفته شوند. یک AS، یک یا چند TS که مرتبط با یک قابلیت بوده را معرفی کرده و ممکن است مقادیر و یا محدودهٔ پارامترهای مخصوصی را که مرتبط با یک TS و یا زیرمجموعهٔ عملیاتی یک TS باشند را مشخص نماید.

سایر انواع RFC

RFCهای بسیاری وجود دارند که جهت آنها این نیست که بصورت یک استاندارد اینترنت درآیند. بعضی RFCها نتیجهٔ فعالیت های موشکافانهٔ انجمن‌ها در مورد بیان یک اصل و یا نتایج این که بهترین راه حل برای انجام برخی عملیات یا وظیفهٔ یک IETF کدام است را استاندارد می‌کنند. این RFCها را بهترین روش جاری Best Current Practice (BCP) می‌نامند. تصویب BCPها تقریباً همان مسیر تصویب استانداردهای پیشنهاد شده را طی می‌کند. برخلاف اسنادی که روی خط استاندارد قرار دارند، برای BCPها یک دورهٔ سه مرحله‌ای وجود ندارد. یک BCP، حالت پیش‌نویس اینترنت تا BCP تصویب شده را در یک قدم طی می‌کند.



شکل ۱-۷ رَوَند انتشار یک RFC اینترنت

یک پروتکل یا مشخصهٔ دیگری که آمادهٔ استاندارد شدن تشخیص داده نمی‌شود، ممکن است بصورت یک RFC آزمایشی منتشر گردد. پس از کار بیشتری، مشخصه ممکن است مجدداً ارائه گردد. معمولاً اگر مشخصه پایدار بوده، اهداف طراحی مشخصی را برآورده کرده، درک خوبی از آن حاصل شده، بازنگری‌های قابل توجهی در آن بوجود آمده و بنظر برسد که ارزش قابل ملاحظه‌ای دارد، آنگاه آن RFC بصورت یک استاندارد پیشنهاد شده درمی‌آید.

نهایتاً یک مشخصهٔ اطلاعاتی (Informational Specification) برای اطلاع انجمن منتشر می‌شود.

۱-۸ ساختار این کتاب

این فصل وظیفهٔ معرفی تمام کتاب را بعهده دارد. بقیهٔ کتاب در سه قسمت سازمان داده شده است:

- **قسمت اول**: مرور مختصری بر الگوریتم‌های رمزنگاری و پروتکل‌های زیرساخت کاربردهای امنیت شبکه دارد که شامل رمزنگاری، توابع درهم‌ساز، امضاءهای دیجیتال و مبادلهٔ کلید هستند.
- **قسمت دوم**: استفاده از الگوریتم‌های رمزنگاری و پروتکل‌های امنیتی، برای فراهم‌آوردن امنیت در شبکه‌ها و اینترنت را بررسی می‌کند. عناوینی همچون اعتبارسنجی کاربر، امنیتe-mail، امنیت IP و امنیت WEB در این فصل گنجانده شده‌اند.

- **قسمت سوم**: مربوط به تسهیلات امنیتی طراحی شده برای حفاظت سیستم‌های رایانه‌ای، در برابر تهدیدهای امنیتی مانند مهاجمین، ویروس‌ها و کِرم‌هاست. دراین قسمت به تکنولوژی دیوارآتش نیز پرداخته می‌شود.

بسیاری از الگوریتم‌های رمزنگاری، پروتکل‌ها و کاربردهای امنیت شبکه که در این کتاب مورد توصیف قرار گرفته‌اند بصورت استاندارد درآمده‌اند. مهم‌ترین آنها، استانداردهای اینترنت‌که در RFCهای اینترنت تعریف شده، و استانداردهای فدرال پردازش اطلاعات (FIPS) که به توسط سازمان ملی استانداردها و تکنولوژی آمریکا(NIST) منتشر می‌شوند، می باشند.

۱-۹ منابع مطالعاتی

[PFLE02] امنیت رایانه و امنیت شبکه را بخوبی معرفی کرده است. دو بررسی فوق‌العادهٔ دیگر را می‌توان در [PIEP03] و [BISH05] جستجو کرد. [BISH03] تقریباً همان مطالب [BISH05] را با جزئیات ریاضی بیشتر و قوی‌تری پوشش داده است. [SCHN00] یک منبع خواندنی ارزنده برای هرکسی است که در زمینهٔ امنیت رایانه و امنیت شبکه فعالیت می‌کند. این کتاب محدودیت‌های تکنولوژی و علی‌الخصوص رمزنگاری در فراهم‌آوردن امنیت را بررسی کرده و نیاز توجه به سخت‌افزار، پیاده‌سازی‌های نرم‌افزاری، شبکه‌ها و مردمی که در ایجاد امنیت و اخلال در امنیت مشارکت دارند را گوشزد می‌کند.

BISH03 Bishop, M. *Computer Security: Art and Science*. Boston : Addison-Wesley, 2003.
BISH05 Bishop, M. *Introduction to Computer Security*. Boston : Addison-Weseley, 2005.
PFLE02 Pfleeger, C. *Security in Computing*. Upper Saddle River, NJ: Prentice Hall, 2002.
PIEP03 Pieprzyk, J.; Hardjono, T.; and Seberry, J. *Fundamentals of Computer Security*. New York: Springer-Verlag, 2003.
SCHN00 Schneier, B**.** *Secrets and Lies: Digital Security in a Networked World*. New York: Wiley 2000.

۱-۱۰ منابع اینترنت و وِب

منابع متعددی در اینترنت پشتیبان این کتاب بوده و به فرد کمک می‌کنند تا خود را با پیشرفت‌های این حوزه هم‌گام سازد.

وِب‌سایت‌های این کتاب

یک صفحهٔ وِب بطور اختصاصی برای این کتاب در آدرس زیر تهیه شده است:

WilliamStallings.com/NetSec/NetSec3e.html

سایت شامل مطالب زیر است:

- **سایت‌های مفید در وِب**: لینک‌هائی به سایت‌های دیگر مرتبط با مطلب، بر اساس فصول کتاب، فراهم شده که شامل سایت‌های این بخش و فصول دیگر است.
- **غلط‌نامه**: یک لیست از غلط‌های کتاب در اینجا نگهداری و مرتباً به روز می‌شود. لطفاً به هر اشتباهی برخورد می‌کنید، آن را برای نگارندهٔ کتاب e-mail کنید.
- **شکل‌ها**: تمام شکل‌های این کتاب با فرمت PDF.
- **جدول‌ها**: تمام جدول‌های این کتاب با فرمت PDF.
- **اسلایدها**: مجموعه‌ای از اسلایدها بصورت Power Point برای هرفصل.
- **لیست پستی اینترنت**: سایت شامل اطلاعات لازم برای ثبت نام در لیست پستی این کتاب است.
- **دوره‌های امنیت شبکه**: لینک‌هائی به دوره‌هائی که بر اساس این کتاب تدریس می‌شود وجود دارد که می‌تواند برای سایر مدرسین جهت معماری درس مفید باشد.

اینجانب همچنین سایت Computer Science Student Resource Site را در آدرس ذیل نگاه داشته‌ام

WilliamStalling.com/StudentSupport.html

هدف این سایت فراهم‌آوردن اسناد، اطلاعات، و لینک‌هائی برای دانشجویان کامپیوتر و افراد حرفه‌ای است. لینک‌ها و اسناد در چهار گروه طبقه‌بندی شده‌اند:

- **ریاضی**: شامل قسمت یادآوری ریاضیات، تئوری مقدماتی صف، بحث مقدماتی سیستم‌های اعداد، و لینک‌های متعددی به سایر سایت‌های ریاضی است.
- **چگونه**: هدایت و راهنمائی برای حل تکالیف، نوشتن گزارش‌های فنی و آماده‌سازی برای ارائهٔ مطلب.
- **منابع تحقیق**: لینک‌هائی به مجموعهٔ مقالات، گزارشات فنی و فهرست‌هاست.
- **متفرقه**: موارد متنوع دیگری از اسناد و لینک‌ها.

سایر وِب‌سایت‌ها

سایت‌های متعددی وجود دارند که در مورد عناوین مطرح شده در این کتاب ارائهٔ اطلاعات می‌نمایند. در فصول آینده، درهرفصل سایت‌های مرتبط با مطلب را در بخش *منابع مطالعاتی* معرفی خواهیم کرد. نظر به این‌که آدرس سایت‌ها مکرراً تغییر می‌کنند، آنها را در این کتاب نیاورده‌ایم. برای تمام سایت‌هائی که در این کتاب لیست شده است، لینک مرتبطی را می‌توان در سایت این کتاب پیدا کرد. سایر لینک‌هائی که در این کتاب ذکر نشده است در طول زمان به سایت اضافه خواهند شد.

سایت‌های زیر در رابطه با رمزنگاری و امنیت شبکه قابل توجه‌اند:

- **COAST**: مجموعه کاملی از لینک‌های مرتبط با رمزنگاری و امنیت شبکه.
- **IETF Security Area**: مطالب مربوط به تلاش‌های استانداردسازی امنیت اینترنت.
- **Computer and Network Security Reference Index**: یک مرجع خوب از سازندگان و محصولات تجاری، سؤالاتی که مکرراً پرسیده می‌شوند(FAQ)، آرشیو گروه‌های خبری، مقاله‌ها و لیست سایر سایت‌ها.
- **The Cryptography FAQ**: سؤالات مفصل و ارزشمند در تمام زمینه‌های رمزنگاری به همراه پاسخ آنها.
- **Tom Dunigan's Security Page**: یک لیست فوق‌العاده از سایت‌های مرجع در مورد رمزنگاری و امنیت شبکه.
- **IEEE Technical Committee on Security and Privacy**: کپی خبرنامه‌ها، و اطلاعات و فعالیتهای مرتبط با IEEE..
- **Computer Security Resource Center**: مربوط به سازمان ملی استانداردها و تکنولوژی آمریکا (NIST) که شامل اطلاعات وسیعی در مورد تهدیدهای امنیتی، فن‌آوری و استانداردهاست.
- **Security Focus**: اطلاعات متنوعی در بارهٔ امنیت، با تأکید خاصی بر محصولات فروشندگان و نیازهای کاربران انتهائی.
- **SANS Institute**: مشابه Security Focus است. مجموعهٔ وسیعی از مقالات را دربر دارد.
- **Data Protection Resource Directory**: مجموعهٔ گوناگونی از لینک‌ها.

گروه‌های خبری USENET

تعدادی از گروه‌های خبری USENET، به بعضی زمینه‌های امنیت شبکه یا رمزنگاری اختصاص دارند. تقریباً همانند تمام گروه‌های خبری USENET، مطالب سازمان یافته نبوده و پراکنده‌اند ولی ممکن است با جستجو در لابلای آنها نکات مفیدی را بدست آورد. مرتبط‌ترین آنها بقرار زیراند:

- **sci.cryp.research**: بهترین گروهی است که میتوان دنبال کرد. عمدتاً عناوین مقالاتی را معرفی می‌کند که به جنبه‌های فنی رمزنگاری ارتباط دارند.
- **sci.crypt**: مباحث عام رمزنگاری و عناوین مرتبط با آن.
- **sci.crypt.random-numbers**: بحث در مورد تصادفی بودن توان رمزنگاری.
- **alt.security**: یک بحث کلی از عناوین رمزنگاری.
- **comp.security.misc**: مباحث عام امنیت رایانه.
- **comp.security.firewalls**: بحث در مورد محصولات دیوارهای آتش و فن‌آوری آنها.
- **comp.security.announce**: اخبار، اطلاعیه‌های CERT.
- **comp.risks**: بحث در مورد خطراتی که از جانب رایانه‌ها و کاربران متوجه جامعه است.
- **comp.virus**: بحث ساده شده‌ای در مورد ویروس‌های رایانه‌ای.

۱-۱۱ واژه‌های کلیدی، سؤالات مرورکنندهٔ بحث و مسائل

واژه‌های کلیدی

مهاجم	intruder	کنترل دست‌یابی	access control
نقاب‌دار	masquerader	تهدید فعال	active threat
عدم انکار	nonrepudiation	اعتبارسنجی	authentication
معماری امنیت OSI	OSI security architecture	اعتبار	authenticity
تهدید غیرفعال	passive threat	قابلیت دسترسی	availability
بازخوانی	replay	محرمانگی داده‌ها	data confidentiality
حملات امنیتی	security attacks	صحت داده‌ها	data integrity
مکانیسم‌های امنیتی	security mechanisms	انکار سرویس	denial of service
سرویس‌های امنیتی	security services	رمزنگاری	encryption
تحلیل ترافیک	traffic analysis	صحت، یکپارچگی	integrity

سؤالات مرورکنندهٔ بحث

۱-۱ معماری امنیت OSI چیست؟

۱-۲ تفاوت بین حملات امنیتی فعال و غیرفعال چیست؟

۱-۳ حملات امنیتی فعال و غیرفعال را دسته‌بندی کرده و بطور مختصر تعریف کنید.

۱-۴ سرویس‌های امنیتی را طبقه‌بندی کرده و بطور مختصر تعریف کنید.

۱-۵ مکانیسم‌های امنیتی را طبقه‌بندی کرده و بطور مختصر تعریف کنید.

مسائل

۱-۱ جدولی مشابه با جدول ۱-۴ ترسیم کنید که رابطهٔ بین سرویس‌های امنیتی و حملات را نشان دهد.

۱-۲ جدولی مشابه با جدول ۱-۴ ترسیم کنید که رابطهٔ بین مکانیسم‌های امنیتی و حملات را نشان دهد.

قسمت اول

رمزنگاری

تا این زمان، مهم‌ترین وسیلهٔ خودکار مورد استفاده در امنیت شبکه و امنیت اطلاعات، رمزنگاری است. دو شکل از رمزنگاری مرسوم است: رمزنگاری رسمی یا متقارن و رمزنگاری کلید- عمومی یا نامتقارن. قسمت اول مروری کلی بر اصول اساسی رمزنگاری متقارن و رمزنگاری کلید- عمومی داشته، نگاهی به الگوریتم‌های پراستفادهٔ آنها انداخته و کاربردهای اساسی این دو برخورد را مورد بحث قرار می‌دهد.

فصل ۲ رمزنگاری متقارن و محرمانگی پیام

فصل۲ روی رمزنگاری متقارن تمرکز کرده و تأکیدی بر پراستفاده ترین تکنیک رمزنگاری، یعنی استاندارد رمزنگاری دیتا (DES)، و الگوریتم‌های متعاقب آن مثل 3DES و استاندارد رمزنگاری پیشرفته (AES) دارد. صرف‌نظر از سؤالات مربوط به ساختار یک الگوریتم رمزنگاری متقارن، تعدادی از مسائل طراحی، مرتبط با استفاده از رمزنگاری متقارن به منظور ایجاد محرمانگی هستند. این فصل شامل بحثی در مورد رمزنگاری پیام‌های طولانی، رمزنگاری سر- به- سر در مقابل رمزنگاری پیوند و تکنیک‌های توزیع کلید است.

فصل ۳ رمزنگاری کلید- عمومی و اعتبارسنجی پیام

یکی از مسائلی که در زمینهٔ معیارهای امنیتی به اندازهٔ محرمانگی اهمیت دارد، اعتبارسنجی است. اعتبارسنجی پیام این اطمینان را ایجاد می‌کند که یک پیام از یک منبع قانونی سرچشمه گرفته است. علاوه برآن اعتبارسنجی می‌تواند شامل حفاظت پیام در برابر دستکاری، تأخیر، بازخوانی و یا تغییر نظم نیز باشد. فصل۳ با تحلیلی در مورد لازمه‌های اعتبارسنجی شروع شده و آنگاه نگاهی به روش‌های اعتبارسنجی می‌اندازد. یک عنصر کلیدی روش اعتبارسنجی، استفاده از اعتبارسنج است که معمولاً یا کُد اعتبارسنجی پیام (MAC) بوده و یا یک تابع درهم‌ساز (hash) است. ملاحظات طراحی برای الگوریتم‌های هر دو نوع مورد بررسی قرار گرفته و چندین مثال مشخص تحلیل شده‌اند.

بعد از رمزنگاری متقارن، نوع مطرح دیگر رمزنگاری، رمزنگاری کلید- عمومی است که انقلابی در دنیای امنیت اطلاعات بوجود آورده است. دنبالهٔ فصل سوّم رمزنگاری کلید- عمومی را معرفی می‌کند. الگوریتم RSA مفصلاً مورد بحث قرار گرفته و مسألهٔ مدیریت کلید مجدداً مورد توجه قرار می‌گیرد. این فصل همچنین تکنیک پرکاربرد توزیع کلید Diffie-Hellman را پوشش می‌دهد. علاوه بر این، این فصل امضاء دیجیتال را تعریف کرده و کاربردهای آن را بررسی می‌نماید.

فصل ۲

رمزنگاری متقارن و محرمانگی پیام

رمزنگاری متقارن که از آن با عناوین رمزنگاری رسمی، رمزنگاری کلید- سرّی، و یا رمزنگاری با یک- کلید نیز یاد می‌شود، تنها نوع رمزنگاری مورد استفاده قبل از معرفی رمزنگاری کلید- عمومی دراواخر دهۀ ۱۹۷۰بود. این رمزنگاری در زمان حال نیز پراستفاده‌ترین روش، از بین دو نوع رمزنگاری معمول می‌باشد.

این فصل را با نگاهی به یک مدل عمومی رمزنگاری متقارن شروع می‌کنیم. این امر ما را قادر میسازد تا با محیطی که الگوریتم‌ها درآن مورد استفاده واقع می‌شوند آشنا شویم. سپس به سه الگوریتم مهم رمزنگاری نظر می‌افکنیم که عبارت از DES، 3DES و AES می‌باشند. بعد از آن رمزنگاری دنباله‌ای متقارن را معرفی خواهیم کرد و با رمز دنباله‌ای RC4 که موارد استفادۀ گسترده‌ای دارد آشنا خواهیم شد. در پایان کاربردهای این الگوریتم‌ها برای ایجاد محرمانگی را بررسی می‌کنیم.

۲-۱ اصول رمزنگاری متقارن

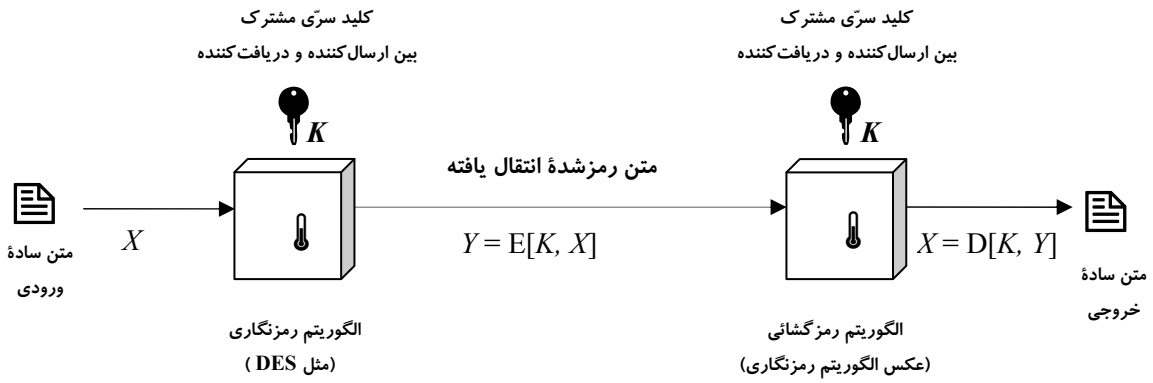
یک طرح رمزنگاری متقارن دارای پنج جزء است (شکل ۱-۲):

- **متن ساده**: این پیام ورودی و یا داده‌هائی است که بعنوان ورودی وارد الگوریتم می‌شود.
- **الگوریتم رمزنگاری**: الگوریتم رمزنگاری، جایگزینی‌ها و تبدیلات مختلفی را روی متن ساده انجام می‌دهد.
- **کلید سرّی**: کلید سرّی نیز یکی از ورودی‌های الگوریتم است. جایگزینی‌ها و تبدیلات انجام‌شده بتوسط الگوریتم، وابسته به این کلید است.
- **متن رمزشده**: این پیام درهم‌ریخته‌شده‌ای است که بعنوان خروجی تولید می‌شود. این متن وابسته به متن ساده و کلید سرّی است. برای یک پیام داده‌شده، دو کلید مختلف دو متن رمزشدۀ مختلف تولید خواهند کرد.
- **الگوریتم رمزگشائی**: این معمولاً همان الگوریتم رمزنگاری است که بطور معکوس اجرا می‌شود. این الگوریتم متن رمزشده و همان کلید سرّی را گرفته و متن ساده اولیه را تولید می‌کند.

برای استفادۀ امن از رمزنگاری متقارن دو چیز مورد نیاز است:

۱- به یک الگوریتم رمزنگاری قوی احتیاج داریم. حداقل مایلیم که الگوریتم چنان باشد که دشمنی که الگوریتم را می‌شناسد و به یک یا چند متن رمزشده دسترسی دارد، قادر نباشد تا متن رمزشده را رمزگشائی کرده و یا کلید رمز را کشف کند. این نیاز معمولاً بصورت محکم‌تری چنین بیان می‌گردد: دشمن بایستی قادر نباشد تا متن رمزشده را رمزگشائی کرده و یا کلید رمز را کشف کند، حتی اگر او چند متن رمزشده بهمراه متون سادۀ نظیر آنها را در اختیار داشته باشد.

۲- فرستنده و گیرنده بایستی کپی‌های کلید سرّی را به روش امنی بدست آورده باشند و آنها را امن نگاه دارند. اگر کسی بتواند کلید را کشف کرده و الگوریتم را نیز بداند، تمام ارتباطاتی که از این کلید استفاده می‌کنند قابل شنود خواهند بود.



شکل ۱-۲ مدل ساده‌شدهٔ رمزنگاری متقارن

مهم است توجه کنیم که امنیت رمزنگاری متقارن بستگی به سرّی بودن کلید، و نه سرّی بودن الگوریتم دارد. یعنی فرض می‌شود که رمزگشائی یک پیام بر مبنای دانستن متن رمزشده *بعلاوهٔ* دانستن الگوریتم رمزنگاری/ رمزگشائی کاری غیرعملی است. بعبارت دیگر، لازم نیست که ما الگوریتم را مخفی نگاه داریم، بلکه فقط کافی است که کلید را مخفی داشته باشیم.

این خصیصهٔ رمزنگاری متقارن همان چیزی است که آن را برای استفادهٔ گسترده مقبول می‌سازد. این واقعیت که الگوریتم لازم نیست تا مخفی بماند، بدین معنی است که سازندگان می‌توانند تراشه‌های ارزان قیمتی که الگوریتم‌های رمزنگاری را عملیاتی می‌سازند تولید نمایند و چنین نیز شده است. این تراشه‌ها در سطح وسیعی در دسترس بوده و در تعدادی محصولات نیز بکار گرفته شده‌اند. در استفاده از رمزنگاری متقارن مسألهٔ اصلی امنیت، حفظ سرّی بودن کلید است.

رمزنگاری

سیستم‌های رمزنگاری معمولاً از سه بُعد مستقل دسته‌بندی می‌شوند:

۱- **نوع عملیات بکار گرفته شده برای تبدیل متن ساده به متن رمزشده**: تمام الگوریتم‌های رمزنگاری بر مبنای دو اصل عمومی قرار دارند: جایگزینی، که در آن هر عنصر متن ساده (بیت، حرف، گروهی از بیت‌ها یا حروف) با عنصر دیگری جایگزین شده، و جابجائی که در آن عناصر متن ساده جای خود را عوض می‌کنند. مهم‌ترین و اصلی‌ترین الزام این است که هیچ اطلاعاتی گم نشود (یعنی تمام عملیات برگشت‌پذیر باشند). بیشتر سیستم‌ها که از آنها با عنوان سیستم‌های ترکیبی یاد می‌شود، شامل چندین مرحلهٔ جایگزینی و جابجائی هستند.

۲- **تعداد کلیدهای استفاده شده**: اگر هم فرستنده و هم گیرنده از یک کلید استفاده کنند، سیستم رمزنگاری را متقارن، تک-کلیدی، کلید- سرّی و یا رسمی گویند. اگر فرستنده و گیرنده هرکدام از یک کلید متفاوت استفاده کنند، سیستم رمزنگاری را نامتقارن، دو- کلیدی و یا کلید- عمومی نامند.

۳- **نحوهٔ پردازش متن سادهٔ پیام**: یک *رمز قالبی*(*block cipher*)، ورودی را بصورت یک بلوک در هر زمان مورد پردازش قرار داده و یک بلوک خروجی برای هر بلوک ورودی تولید می‌کند. یک *رمز دنباله‌ای* (*stream cipher*) عناصر ورودی را بصورت پیوسته پردازش کرده و همینطور که جلو می‌رود، عناصر خروجی نیز بطور پیوسته از آن خارج می‌گردند.

شکستن رمز یا کشف رمز

کوشش برای کشف کردن متن سادهٔ پیام و یا کلید را *شکستن رمز* گویند. استراتژی بکارگرفته‌شده در این مورد، بستگی به طبیعت روش رمزنگاری و اطلاعات قابل دسترسی مسئول کشف رمز دارد.

جدول ۱-۲ انواع مختلف حملات کشف رمز، بر مبنای میزان اطلاعاتی که در اختیار کشف‌کنندهٔ رمز قرار دارد را نشان می‌دهد. مشکل‌ترین مورد زمانی است که *فقط متن رمزشده* در دسترس باشد. در بعضی موارد نه تنها الگوریتم رمزنگاری شناخته شده است، بلکه عموماً می‌توانیم فرض کنیم که دشمن از الگوریتم استفاده شده برای رمزنگاری مطلع است. یکی از حملاتی که تحت این شرایط ممکن است رخ دهد، جستجوی جامع(brute-force) امتحان کردن همهٔ کلیدهای ممکن است. اگر فضای کلید خیلی وسیع باشد، این امر غیرعملی خواهد شد. بنابراین دشمن بایستی به تجزیه و تحلیل خود متن رمزشده متکی بوده و تست‌های آماری مختلفی را روی آن انجام دهد. برای استفاده از این روش، دشمن بایستی ایده‌هائی نسبت به نوع متن ساده پیام که پنهان شده است داشته باشد. یعنی مثلاً بداند که پیام، یک متن انگلیسی، یک متن فرانسه، یک فایل EXE، یک برنامهٔ Java، یک سند مالی و غیره است.

دفاع در برابر حملهٔ فقط- متن رمزشده ساده‌ترین نوع دفاع است زیرا دشمن کمترین اطلاعات را داراست. در بسیاری موارد، شکنندهٔ رمز اطلاعات بیشتری دارد. او ممکن است قادر باشد تا یک یا چند پیام ساده به همراه فرم رمزنگاری‌شدهٔ آنها را بدست آورد. کشف‌کنندهٔ رمز ممکن است بداند که الگوهای مخصوصی در پیام وجود دارد. مثلاً فایلی که با فرمت Postscript کُد می‌شود همیشه با الگوی خاصی شروع می‌شود و یا ممکن است یک عنوان استانداردشده، همیشه همراه یک پیام الکترونیکی انتقال‌دهندهٔ پول وجود داشته باشد. همهٔ اینها مثال‌هائی از *متن سادهٔ معلوم*‌اند. با چنین معلوماتی، تحلیل‌گر رمز ممکن است بتواند کلید را، بر مبنای آگاهی از الگوی متن سادهٔ رمزنشده، بدست آورد.

جدول ۱-۲ انواع حملات به پیام‌های رمزنگاری شده

نوع حمله	آنچه برای کشف کنندهٔ رمز معلوم است
فقط- متن رمزشده	• الگوریتم رمزنگاری • متن رمزشده‌ای که باید باز شود
متن سادهٔ معلوم	• الگوریتم رمزنگاری • متن رمزشده‌ای که باید باز شود • یک یا چندجفت متن ساده- متن رمزشده بتوسط کلید سرّی
متن سادهٔ انتخاب شده	• الگوریتم رمزنگاری • متن رمزشده‌ای که باید باز شود • پیام سادهٔ انتخاب شده بتوسط کشف‌رمزکننده، بهمراه متن رمزشدهٔ نظیر آن بتوسط کلید سرّی
متن رمزشدهٔ انتخاب شده	• الگوریتم رمزنگاری • متن رمزشده‌ای که باید باز شود • متن رمزشدهٔ انتخاب‌شده بتوسط شکنندهٔ رمز، بهمراه متن رمزگشائی شده نظیر آن با استفاده از کلید سرّی
متن انتخاب شده	• الگوریتم رمزنگاری • متن رمزشده‌ای که باید باز شود • پیام سادهٔ انتخاب‌شده بتوسط کشف‌رمزکننده، بهمراه متن رمزشدهٔ نظیرآن با استفاده از کلید سرّی • متن رمزشدهٔ انتخاب‌شده بتوسط کشف‌رمزکننده، بهمراه متن رمزگشائی شده نظیر آن با استفاده از کلید سرّی

موردی که خیلی مرتبط با حملهٔ متن سادهٔ معلوم است، چیزی است که می‌توان از آن با نام حملهٔ کلمهٔ محتمل یاد کرد. اگر دشمن روی کشف رمز یک متن عمومی کار کند، او ممکن است اطلاعات کمی نسبت به محتویات پیام داشته باشد. ولی اگر دشمن بدنبال اطلاعات خیلی تخصصی باشد، آنگاه ممکن است بخشی از پیام را بشناسد. بعنوان مثال اگر یک سند اطلاعات مالی منتقل میشود، دشمن ممکن است از محل برخی کلمات کلیدی در عنوان فایل یا خبر باشد. مثال دیگر اینکه کُد اولیهٔ یک برنامهٔ تهیه‌شده در یک سازمان ممکن است شامل یک جملهٔ مربوط به نام سازمان در یک محل مشخص و استاندارد باشد.

اگر تحلیل‌گر بطریقی قادر باشد تا سیستم منبع را وادارد تا پیام انتخاب‌شده‌ای بتوسط تحلیل‌گر را رمزنگاری کند، آنگاه یک حملهٔ *متن سادهٔ انتخاب‌شده* محتمل است. در حالت کلی، اگر تحلیل‌گر بتواند پیام‌هائی را جهت رمزنگاری انتخاب کند، او ممکن است زیرکانه از پیام‌هائی استفاده کند که انتظار می‌رود تا ساختار کلید را آشکار سازند.

جدول ۱-۲ دو نوع حملهٔ دیگر را نیز ذکرکرده است: متن رمزشدهٔ انتخاب‌شده و متن انتخاب‌شده. این حمله‌ها کمتر بعنوان تکنیک‌های کشف رمز بکار می‌روند، ولی با این وجود راه‌های گشوده‌ای برای حمله‌اند.

تنها الگوریتم‌های نسبتاً ضعیف در برابر حملهٔ فقط- متن رمزشده شکست می‌خورند. معمولاً یک الگوریتم رمزنگاری طوری طراحی می‌شود که در مقابل حملهٔ متن سادهٔ معلوم نیز مقاومت کند.

یک روش رمزنگاری در صورتی از **نظر محاسباتی امن** است که متن رمزشده بتوسط آن روش، یک و یا هردو شرط زیر را داشته باشد:

- هزینهٔ شکستن رمز، از ارزش اطلاعات رمزشده تجاوز کند.
- زمان لازم برای شکستن رمز، از عمر مفید اطلاعات تجاوز کند.

متأسفانه بسیار سخت است تا میزان کوشش لازم برای کشف متن رمزشده را تخمین زد. ولی با فرض اینکه هیچ ضعف ذاتی ریاضی در الگوریتم وجود نداشته باشد، آنگاه با تصور یک حملهٔ همه جانبه میتوان تخمین معقولی نسبت به هزینه‌ها و زمان کشف رمز بدست آورد.

روش حملهٔ همه جانبه، شامل امتحان کردن همهٔ کلیدهای ممکن است تا یک ترجمهٔ قابل فهم از متن رمزنگاری شده به متن ساده به دست آید. بطور متوسط، برای موفقیت بایستی نصف کلیدهای ممکن را امتحان کرد. جدول ۲-۲ زمان صرف شده برای کلیدهائی با اندازه‌های مختلف را نشان می‌دهد. در الگوریتم DES از یک کلید ۵۶- بیتی استفاده می‌شود. برای اندازهٔ هر کلید، نتایج با فرض اینکه یک میکروثانیه برای هر رمزگشائی ساده صرف خواهد شد، نشان داده شده است. یک میکروثانیه برای هر رمزگشائی، اندازهٔ معقولی برای ماشین‌های امروزی است. با استفاده از تعداد زیادی میکروپروسسور با سازماندهی موازی، ممکن است نرخ پردازش را به چندین برابر افزایش داد. ستون آخر در جدول ۲-۲ نتایج را برای سیستمی که بتواند یک میلیون کلید در هر میکروثانیه را آزمایش کند، نشان می‌دهد. همانطور که مشاهده می‌کنید، در چنین سطح عملکردی، DES دیگر نمی‌تواند از نظر محاسباتی امن فرض شود.

ساختار رمز Feistel

بیشتر الگوریتم‌های رمزنگاری متقارن قالبی، از جمله DES، دارای ساختاری هستند که برای اولین بار بتوسط Horst Feistel از شرکت IBM در سال ۱۹۷۳ [FEIS73] توصیف گردید و در شکل ۲-۲ نشان داده شده است. ورودی‌های الگوریتم رمزنگاری، یک بلوک از متن ساده با طول $2w$ بیت و یک کلید K است. بلوک متن ساده به دو نیمهٔ L_0 و R_0 تقسیم میشود. دو نیمهٔ دیتا، n دُور پردازش را پشت سر گذاشته و آنگاه با یکدیگر ترکیب شده تا بلوک متن رمزشده را

جدول ۲-۲ زمان متوسط لازم برای امتحان همهٔ کلیدهای رمز

اندازهٔ کلید (bits)	تعداد کلیدهای ممکن	زمان لازم کشف رمز با یک رمزگشائی در هر میکروثانیه	زمان لازم کشف رمز با یک میلیون رمزگشائی در هر میکروثانیه
۳۲	$۲^{۳۲} = ۴/۳ \times ۱۰^{۹}$	$۲^{۳۱}$ میکروثانیه= ۳۵/۸ دقیقه	۲/۱۵ میلی ثانیه
۵۶	$۲^{۵۶} = ۷/۲ \times ۱۰^{۱۶}$	$۲^{۵۵}$ میکروثانیه= ۱۱۴۲ سال	۱۰/۰۱ ساعت
۱۲۸	$۲^{۱۲۸} = ۳/۴ \times ۱۰^{۳۸}$	$۲^{۱۲۷}$ میکروثانیه= $۵/۴ \times ۱۰^{۲۴}$ سال	$۵/۴ \times ۱۰^{۱۸}$ سال
۱۶۸	$۲^{۱۶۸} = ۳/۷ \times ۱۰^{۵۰}$	$۲^{۱۶۷}$ میکروثانیه= $۵/۹ \times ۱۰^{۳۶}$ سال	$۵/۹ \times ۱۰^{۳۰}$ سال
۲۶ کاراکتر (جایگشت)	$۲۶! = ۴ \times ۱۰^{۲۶}$	۲×۱۰^{۲۶} میکروثانیه= $۶/۴ \times ۱۰^{۱۲}$ سال	$۶/۴ \times ۱۰^{۶}$ سال

ایجاد نمایند. هر دُور i دارای ورودی‌های L_{i-1} و R_{i-1}، که خروجی دور ماقبل بوده، و همچنین یک زیرکلید K_i که از کلید K مشتق شده است می‌باشد. عموماً زیرکلیدهای K_i با یکدیگر و همچنین با K فرق داشته و بتوسط یک الگوریتم تولید زیرکلید خلق می‌شوند.

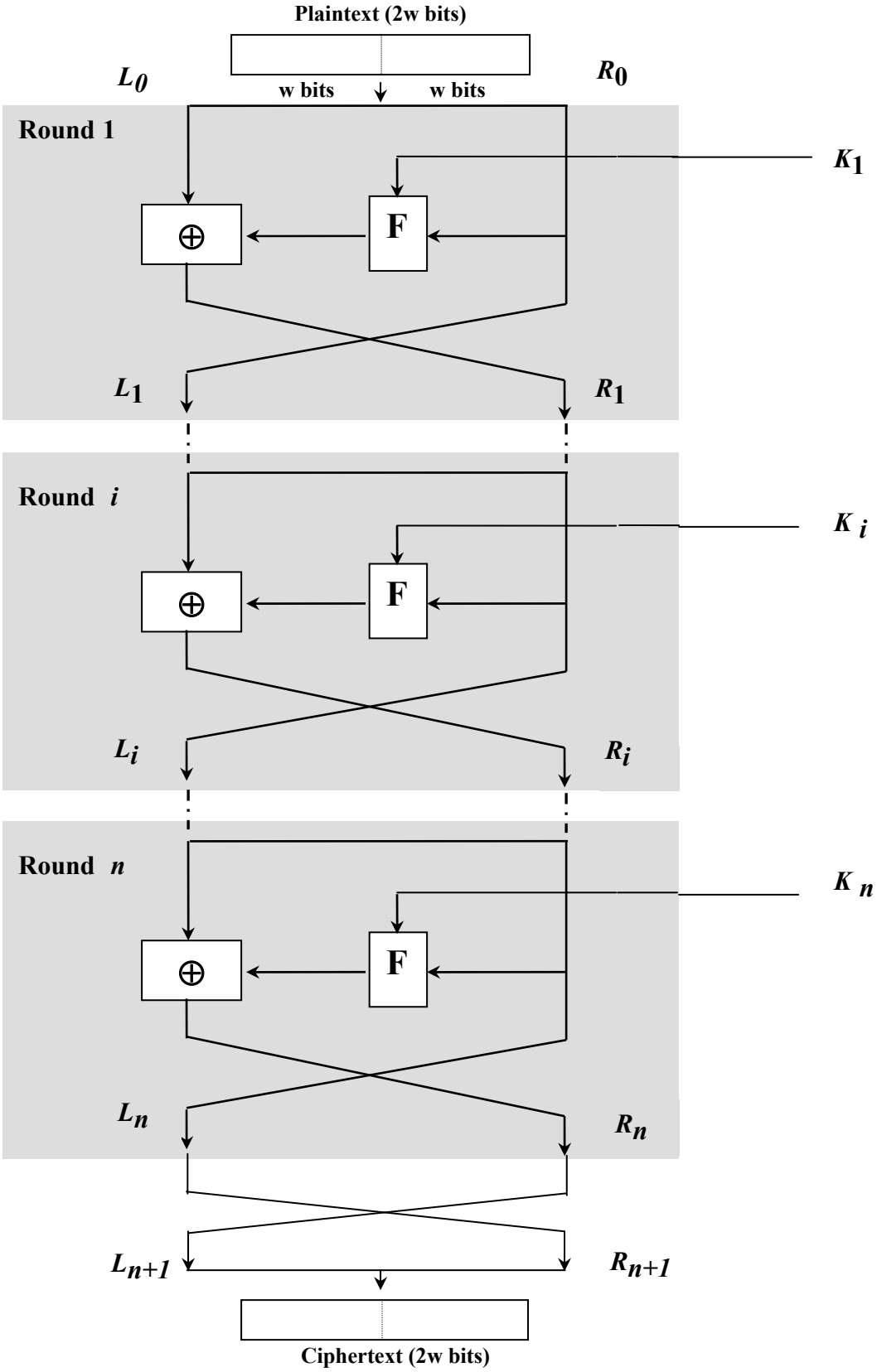
همهٔ دُورهای رمزنگاری دارای ساختار یکسانی هستند. یک جایگزینی روی نیمهٔ چپ دیتا انجام می‌شود. این امر با اعمال یک *تابع دُور* (*round function*) F به نیمهٔ راست دیتا و سپس XOR کردن خروجی این تابع با نیمهٔ چپ دیتا حاصل می‌گردد. در هر دُور، تابع دُور دارای ساختار عمومی یکسانی است که بتوسط زیرکلید دُور K_i پارامترهای آن تغییر می‌یابد. پس از این جایگزینی، یک جایگشت صورت می‌پذیرد که شامل تعویض محل دو نیمهٔ دیتاست.

ساختار Feistel یک مثال خاص از ساختار عمومی‌تری است که بتوسط تمام رمزهای قالبی متقارن مورد استفاده قرار می‌گیرد. در حالت کلی، یک رمز قالبی متقارن شامل تعدادی از دُورهای متوالی است که در هر دُور، عملیات جایگزینی و جابجائی با وابستگی به اندازهٔ کلید سرّی دُور صورت می‌پذیرد. تحقق واقعی یک رمز قالبی متقارن، بستگی به انتخاب پارامترهای زیر و موارد طراحی دارد:

- **اندازهٔ بلوک**: هرچقدر اندازهٔ بلوک‌ها بزرگتر باشد (با فرض ثابت بودن سایر پارامترها)، امنیت بیشتر ولی سرعت رمزنگاری/ رمزگشائی کمتر است. مصالحهٔ مناسب در این مورد، انتخاب بلوکی با طول ۱۲۸ بیت بوده که در طراحی رمز قالبی، تقریباً انتخابی همگانی است.
- **اندازهٔ کلید**: اندازهٔ بزرگتر کلید بمنزلهٔ امنیت بیشتر است، ولی ممکن است سرعت رمزنگاری/ رمزگشائی را کاهش دهد. معمول‌ترین کلیدها در الگوریتم‌های مدرن، دارای طول ۱۲۸ بیت هستند.
- **تعداد دُورها**: جوهرهٔ یک رمز قالبی متقارن در این است که تنها یک دُور رمزنگاری، امنیت مناسبی را ایجاد نمی‌کند و بنابراین دُورهای بیشتری از رمزنگاری برای افزایش امنیت مورد نیاز است. اندازهٔ معمول در این مورد، ۱۶ دُور است.
- **الگوریتم تولید زیرکلید**: پیچیدگی بیشتر در این الگوریتم، بایستی باعث افزایش پیچیدگی در شکستن رمز گردد.
- **تابع دُور**: باز هم پیچیدگی بیشتر، معمولاً بمعنای مقاومت بیشتر در مقابل کشف رمز است.

دو مورد دیگر را نیز در طراحی یک رمز قالبی متقارن بایستی درنظر گرفت:

نرم‌افزار رمزنگاری/ رمزگشائی سریع: در بسیاری موارد، رمزنگاری در دل کاربردها و یا توابع اجرائی طوری قرار گرفته است که خارج از حیطهٔ اجرای سخت‌افزاری الگوریتم است. بنابراین سرعت اجرای الگوریتم یکی از نکته‌های قابل تأمل است.



شکل۲-۲ شبکهٔ کلاسیک Feistel

- **سهولت تحلیل**: اگرچه علاقه‌مندیم که برای جلوگیری از شکستن رمز، هرچه ممکن است الگوریتم را پیچیده‌تر کنیم ولی ایجاد سهولت در تحلیل الگوریتم محسنات زیادی دارد. یعنی اگر الگوریتم بتواند بطور مختصر و روشن بیان شود، تحلیل آن برای آسیب‌پذیری‌های مربوط به شکستن رمز هم ساده‌تر بوده و بنابراین سطح اطمینان به قدرت آن را می‌توان افزایش داد. بعنوان مثال، DES دارای عملکرد تحلیلی ساده‌ای نیست.

رمزگشائی با یک رمز قالبی متقارن نیز ضرورتاً مثل همان رمزنگاری آن است. قاعده چنین است: متن رمزشده را بعنوان ورودی الگوریتم بکار برده ولی زیرکلیدهای K_i را با نظم معکوس استفاده می‌کنیم. یعنی K_n در دور اول، K_{n-1} در دور دوم، و بهمین ترتیب تا K_1 که در دور آخر مورد استفاده قرار می‌گیرد. این خاصیت جذابی است زیرا لازم نیست تا از دو الگوریتم مختلف، یکی برای رمزنگاری و یکی برای رمزگشائی استفاده شود.

۲-۲ الگوریتم‌های رمزنگاری قالبی متقارن

معمول‌ترین الگوریتم‌های بکارگرفته شده برای رمزنگاری متقارن، رمزهای قالبی هستند. یک رمز قالبی، متن سادهٔ ورودی را در قالب بلوک‌هائی با اندازهٔ ثابت پردازش کرده و یک بلوک متن رمزشده با همان اندازه را، برای هر بلوک متن ساده تولید می‌کند. در این بخش سهٔ نوع از مهم‌ترین رمزهای قالبی متقارن را مورد بررسی قرار می‌دهیم: استاندارد رمزنگاری دیتا (DES)، DES سه‌گانه (3DES) و استاندارد رمزنگاری پیشرفته (AES).

استاندارد رمزنگاری دیتا Data Encryption Standard (DES)

پراستفاده‌ترین روش رمزنگاری، بر مبنای استاندارد رمزنگاری دیتا (DES) قرار دارد که درسال ۱۹۷۷ بتوسط دفتر ملی استانداردها در آمریکا که امروز مؤسسهٔ ملی استانداردها و تکنولوژی(NIST) خوانده می‌شود، تحت عنوان استاندارد فدرال پردازش اطلاعات ۴۶ (FIP PUB46) پذیرفته شد. از خود الگوریتم، با نام الگوریتم رمزنگاری دیتا (DEA) یاد می‌شود.

توصیف الگوریتم

متن ساده دارای طول ۶۴ بیت بوده و طول کلید۵۶ بیت است. متون سادهٔ طویل‌تر در بلوک‌های ۶۴- بیتی مورد پردازش قرار می‌گیرند. ساختار DES تقریباً همان ساختار شبکهٔ Feistel با کمی تغییرات است که در شکل ۲-۲ نشان داده شده است. ۱۶ دُور پردازش وجود دارد. از کلید اولیهٔ ۵۴- بیتی، شانزده زیرکلید تولید می‌شود که هرکدام در یک دُور پردازش مورد استفاده قرار می‌گیرند.

نحوهٔ رمزگشائی با DES ضرورتاً شبیه نحوهٔ رمزنگاری با آن است. قاعده چنین است: متن رمزشده را بعنوان ورودی الگوریتم DES بکار برده ولی از زیرکلیدها با نظم معکوس استفاده کنید. یعنی در اولین تکرار کلید K_{16}، در دومین تکرار کلید K_{15}، و بهمین نحو جلورفته و در شانزدهمین و آخرین تکرار کلید K_1 را بکار برید.

توانائی DES

نگرانی نسبت به توانائی DES در دو مقولهٔ جدا قرار دارد: نگرانی در مورد خود الگوریتم و نگرانی در مورد استفاده از یک کلید ۵۴- بیتی. اولین نگرانی، در رابطه با امکان شکستن رمز با استفاده از بکارگیری مشخصه‌های الگوریتم DES است.

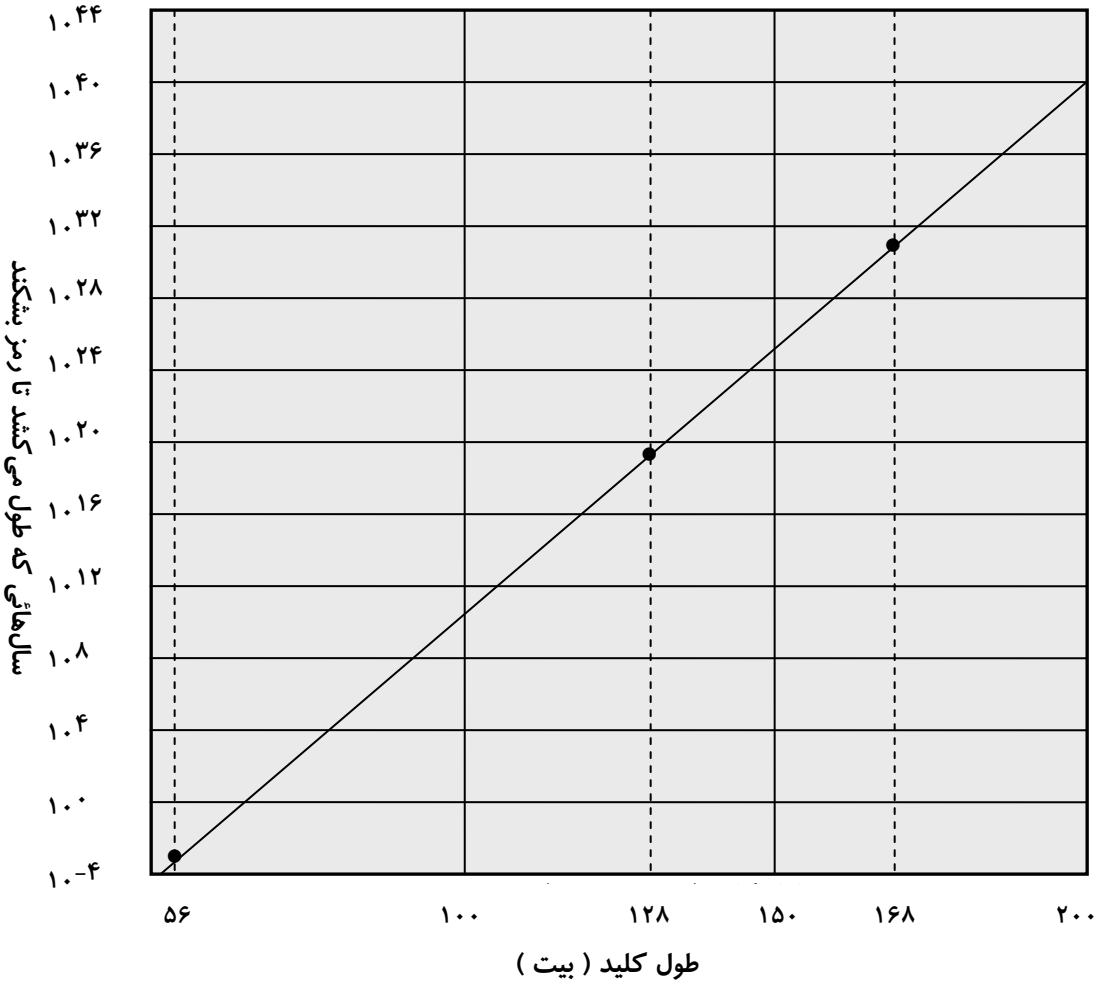
در طول سالیان گذشته، تلاش‌های بسیاری برای کشف و سوءاستفاده از نقاط ضعف الگوریتم DES انجام شده و بهمین مناسبت DES الگوریتمی است که بیش از همه مورد مطالعه قرارگرفته است. با وجود تلاش‌های فراوان، تا کنون کسی نتوانسته است که یک ضعف حیاتی در DES پیدا کند.

نگرانی جدی‌تر مربوط به طول کلید است. با کلیدی با طول ۵۶ بیت، تعداد $۲^{۵۶}$ کلید ممکن وجود دارد که تقریباً $۷/۲ \times ۱۰^{۱۶}$ کلید است. بنابراین در صورت ظاهر، یک حملهٔ همه جانبه غیرعملی خواهد بود. با فرض اینکه بطور متوسط نصف فضای کلید را بایستی برای یافتن آن جستجو کرد، اگر قرار باشد یک ماشین تنها که در هر میکروثانیه یک فقره رمزگشایی DES را انجام میدهد بکارگرفته شود، بیش از هزارسال طول خواهد کشید تا رمز شکسته شود (جدول ۲-۲ را ملاحظه کنید).

اما فرض یک رمزنگاری در هر میکروثانیه بیش از حد، محافظه‌کارانه است. بالاخره و بطور قطعی در ماه جولای سال ۱۹۹۸ اثبات گردید که DES ناامن است. دلیل امر این بود که Electronic Frontier Foundation (EFF) اعلام کرد که رمز یک DES را با استفاده از یک ماشین مخصوص "DES cracker" که با هزینهٔ کمتر از ۲۵۰,۰۰۰ دلار ساخته شده است، شکسته است. این حمله کمتر از سه روز طول کشیده بود. EFF توصیف مفصلی از ماشین مزبور را منتشر نموده است تا دیگران نیز بتوانند رمزشکن خود را بسازند [EFF98] و البته با توجه به اینکه با افزایش سرعت، قیمت سخت‌افزارها پائین می‌آید، DES بطور ضمنی بی‌ارزش خواهد شد.

مهم است توجه شود که در حملهٔ جستجوی کلید، نکات مهم‌تری نیز سوای جستجوی همهٔ کلیدها وجود دارد. بغیر از موردی که واقعاً یک متن ساده در دسترس باشد، یک تحلیل‌گر بایستی یک متن ساده را بعنوان متن ساده تشخیص دهد. اگر پیام صرفاً یک متن ساده به زبان انگلیسی باشد در این صورت نتیجه به سهولت استخراج خواهد شد، اگرچه وظیفهٔ شناخت زبان انگیسی را باید خودکار نمود. اگر متن پیام قبل از رمزنگاری فشرده شده باشد، شناخت آن دشوارتر خواهد شد و اگر پیام نمونهٔ عام‌تری از دیتا، همانند یک فایل عددی بوده و فشرده‌سازی هم شده باشد، مشکل تحلیل خودکار آن بازهم پیچیده‌تر خواهد گردید. بنابراین برای فراهم‌آوردن روش همه جانبه، مقداری دانش در مورد متن ساده مورد نیاز بوده و همچنین لازم است تا وسیله‌ای برای تمیزدادن متن ساده از یک متن بی‌معنی بصورت خودکار وجود داشته باشد. روش EFF این مقوله را مورد توجه قرار داده و همچنین تکنیک‌های خودکاری را که در برخی زمینه‌ها مؤثرند، معرفی می‌کند.

یک نکتهٔ نهائی: اگر تنها فرم ممکن حمله نسبت به یک الگوریتم رمزنگاری، حملهٔ همه جانبه باشد، آنگاه روش مقابله با آن کاملاً روشن بوده و آن استفاده از کلیدهای طولانی‌تراست. برای این که ایده‌ای نسبت به اندازهٔ کلید مورد نیاز پیدا کنیم، اجازه دهید تا از رمزشکن EFF برای تخمین امر استفاده نمائیم. EFF cracker یک نمونهٔ منحصربفرد بوده و ما می‌توانیم فرض کنیم که با تکنولوژی امروز، ساخت یک ماشین سریع‌تر مقرون بصرفه‌تر است. اگر فرض کنیم که یک رمزشکن بتواند در هر میکروثانیه یک میلیون رمزگشائی انجام دهد، که نرخی است که در جدول ۲-۲ از آن استفاده شده است، آنگاه تقریباً ۱۰ ساعت طول خواهد کشید تا یک رمز DES شکسته شود. سرعت این رمزشکنی تقریباً ۷ برابر بیشتر از نتیجهٔ EFF است. با استفاده از این نرخ، شکل ۳-۲ نشان می‌دهد که چقدر طول خواهد کشید تا الگوریتمی با فرم DES را برحسب تابعی از اندازهٔ کلید شکست. بعنوان مثال برای یک کلید ۱۲۸- بیتی، که در الگوریتم‌های فعلی مرسوم است، بیش از $۱۰^{۱۸}$ سال طول خواهد کشید تا بتوان رمزی را، با استفاده از رمزشکن EFF شکست. حتی اگر بتوان سرعت رمزشکن را با فاکتور یک تریلیون ($۱۰^{۱۲}$) افزایش داد، بازهم یک میلیون سال طول خواهد کشید تا رمز شکسته شود. بنابراین یک کلید ۱۲۸- بیتی برای استفاده در الگوریتمی که در مقابل حملهٔ همه جانبه شکست‌ناپذیر باشد، یک انتخاب تضمین شده است.



شکل ۳-۲ زمان شکستن یک رمز (با فرض 10^6 رمزگشائی در هر میکروثانیه)

Triple DES

Triple DES (3DES)، اولین بار در سال ۱۹۸۵ برای استفاده درکاربردهای مالی، با نام X9.17 در استانداردهای ANSI ثبت گردید. 3DES با انتشار FIPS PUB 46-3 در سال ۱۹۹۹ بعنوان بخشی از استاندارد رمزنگاری دیتا (DES) بکار گرفته شد.

3DES از سه کلید و سه بار اجرای الگوریتم DES استفاده می‌کند. تابع از یک دنبالهٔ رمزنگاری- رمزگشائی- رمزنگاری (EDE) تبعیت می‌کند(شکل ۴-۲الف):

$$C = \mathrm{E}(K_3, \mathrm{D}(K_2, \mathrm{E}(K_1, P)))$$

که در آن

C = متن رمزشده (ciphertext)

P = متن ساده (plaintext)

$\mathrm{E}[K, X]$ = رمزنگاری X با استفاده از کلید K

$\mathrm{D}[K, Y]$ = رمزگشائی Y با استفاده از کلید K

رمزگشائی بسادگی همان عملیات قبل است که ترتیب کلیدها در آن عوض شده است(شکل ۴-۲ب):

$$P = \mathrm{D}(K_1, \mathrm{E}(K_2, \mathrm{D}(K_3, C)))$$

ازنظر رمزنگاری، هیچ ویژگی خاصی در استفاده از رمزگشائی مرحلهٔ دوم رمزنگاری 3DES وجود ندارد. تنها حسن آن این است که به کاربران 3DES اجازه می‌دهد تا داده‌هائی را که بتوسط فرم قدیمی DES رمزنگاری شده بودند، رمزگشائی نمایند:

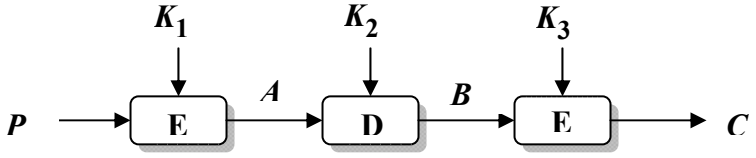
$$C = \mathrm{E}(K_1, \mathrm{D}(K_1, \mathrm{E}(K_1, P))) = \mathrm{E}[K, P]$$

با سه کلید متمایز، 3DES دارای کلیدی با طول مؤثر ۱۶۸ بیت است. FIPS 46-3 همچنین استفاده از دو کلید، $K_1 = K_3$، را اجازه میدهد که در این مورد طول کلید ۱۱۲ بیت خواهد بود. FIPS 46-3 شامل سه دستورالعمل زیر برای 3DES است:

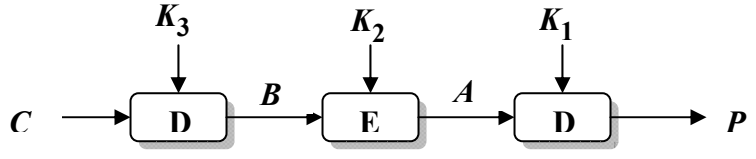
- 3DES الگوریتم رمزنگاری متقارن منتخب و تأئیدشده بتوسط FIPS است.
- DES اولیه که از یک کلید ۵۶- بیتی استفاده می‌کند تنها در استاندارد سیستم‌هائی که وارث آن هستیم مجاز می‌باشد. ساختارهای جدید بایستی از 3DES حمایت نمایند.
- توصیه میشود که سازمان‌های دولتی، سیستم‌های موروثی DES را با 3DES تعویض نمایند.
- پیش‌بینی میشود که 3DES و AES در کنار هم، بعنوان الگوریتم‌های پذیرفته شدهٔ FIPS، همزیستی داشته و در طول زمان بتدریج 3DES حذف و AES استاندارد غالب شود.

بسهولت مـــی‌توان دریافت که 3DES یک الگوریتم نیرومند است. چـــون الگوریتم رمـــزنگاری بستر آن DEA (Data Encryption Algorithm) است، 3DES می‌تواند در مقابل تلاش‌های مربوط به کشف رمز، همان ادعاهای DEA را داشته باشد. علاوه برآن با یک کلید ۱۶۸- بیتی، حملهٔ همه جانبه به آن عملاً غیرممکن است.

بالاخره و در نهایت قرار است AES جایگزین 3DES شود، ولی این تحول سالها طول خواهد کشید. NIST پیش‌بینی میکند که 3DES برای آینده‌ای قابل پیش‌بینی، الگوریتم موفقی باشد.



(الف) رمزنگاری



(ب) رمزگشائی

شکل ۴-۲ Triple DES

استاندارد رمزنگاری پیشرفته Advanced Encryption Standard (AES)

3DES دارای دو جاذبه است که استفادهٔ گسترده از آن در چندسال آینده را تضمین می‌کند. اولاً با طول کلید ۱۶۸- بیتی خود، بر آسیب‌پذیری‌های ناشی از حملهٔ همه جانبه در DEA غلبه می‌کند. ثانیاً الگوریتم رمزنگاری 3DES همان DEA است. این الگوریتم بیش از هر الگوریتم رمزنگاری دیگر در طول زمان مورد رسیدگی دقیق قرار گرفته و هیچ نوع آسیب‌پذیری، بجز مورد حملهٔ همه جانبه، در آن مشاهده نشده است. در نتیجه در مورد مقاومت 3DES در مقابل کشف رمز اعتماد زیادی وجود دارد. از این‌رو اگر فقط مسألهٔ امنیت مورد توجه بود، 3DES الگوریتم انتخابی مناسبی برای رمزنگاری در طول دهه‌های آینده باقی می‌ماند.

مشکل اصلی 3DES این است که این الگوریتم از نظر نرم‌افزاری لَخت است. DEA اولیه برای تجهیزات سخت‌افزاری سال‌های میانهٔ ۱۹۷۰ طراحی شده بود و کُد نرم‌افزاری بهره‌وری را تولید نمی‌کند. 3DES که سه برابر DES عملیات اجرائی دارد، حتماً کندتر خواهد بود. مشکل دوم این است که DEA و 3DES هر دو از یک بلوک دیتا با اندازهٔ ۶۴- بیت استفاده می‌کنند. به دلایلی که هم مربوط به بهره‌وری و هم مربوط به مسائل امنیتی می‌شود، استفاده از بلوکی با اندازهٔ بزرگتر مطلوب‌تر است.

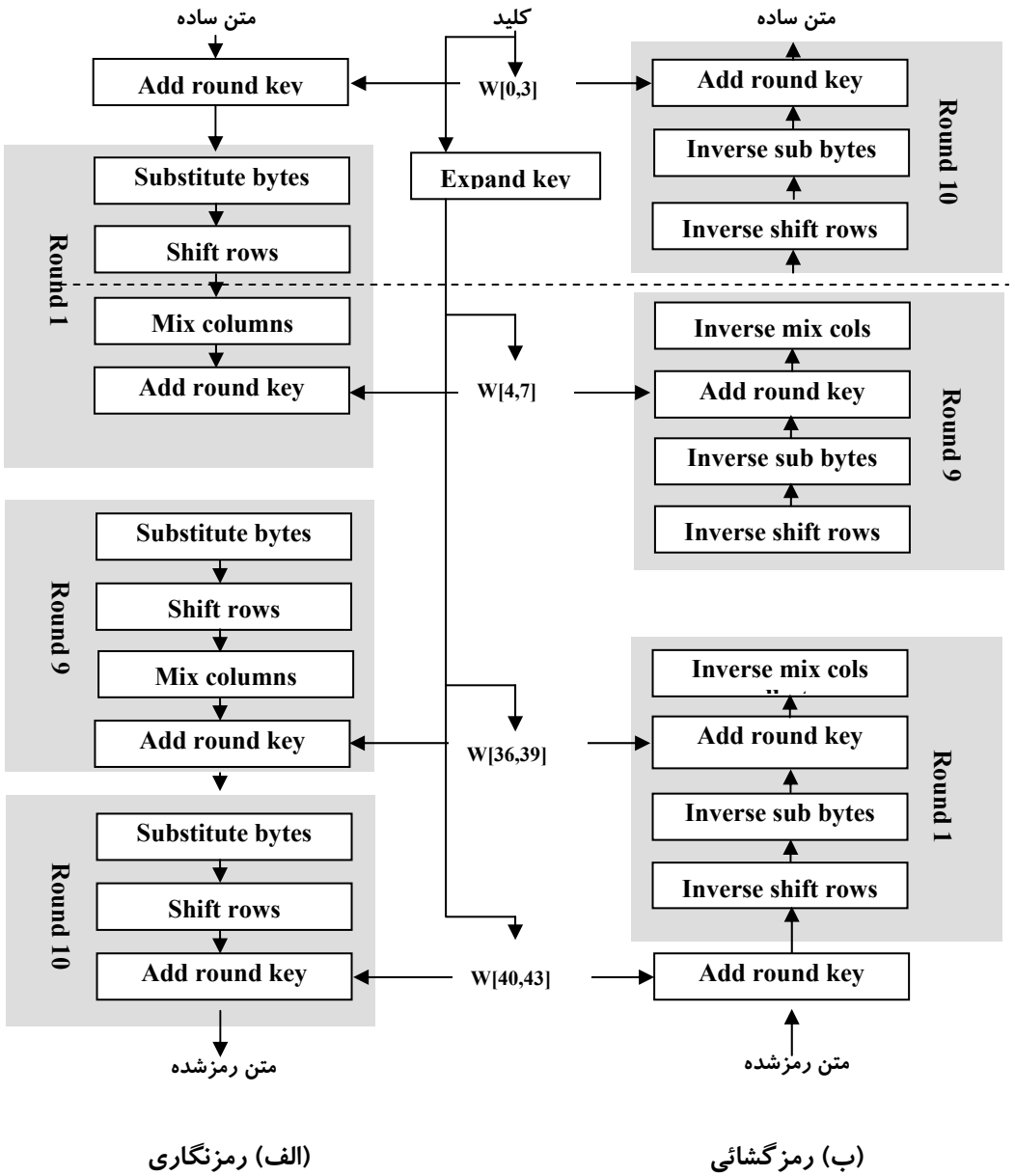
بعلت این مشکلات، 3DES در درازمدت کاندیدای معقولی نیست. برای جانشینی آن با انتخاب بهتری، NIST در سال ۱۹۹۷، فراخوانی برای طراحی یک استاندارد رمزنگاری پیشرفته (AES) منتشر کرد که بایستی دارای توان امنیتی برابر و یا بهتر از 3DES و بهره‌وری قابل ملاحظه‌ای می‌بود. علاوه بر بیان نیازهای کلی، NIST مشخص نمود که AES بایستی یک رمز قالبی متقارن با طول بلوک ۱۲۸- بیت بوده و از کلیدهائی با طول ۱۲۸، ۱۹۲، و ۲۵۶ بیت پشتیبانی نماید. نکات مورد ارزیابی شامل امنیت، بهره‌وری محاسباتی، نیازهای مربوط به حافظه، تناسب سخت‌افزار و نرم‌افزار و قابلیت انعطاف اعلام گردید.

در اولین دور ارزیابی، ۱۵ الگوریتم از بین پیشنهادها انتخاب شدند. در دور بعدی، ۵ الگوریتم پذیرفته شدند. بالاخره NIST ارزیابی خود را به پایان رسانده و یک استاندارد نهائی (FIPS PUB 197) را در نوامبر سال ۲۰۰۱ منتشرنمود. Rijndael بعنوان الگوریتم انتخابی AES پذیرفته شد. دو پژوهشگری که Rijndael را تهیه و ارائه کردند هر دو رمزنگارانی از بلژیک به نام‌های Dr. Joan Daemen و Dr. Vincent Rijmen بودند.

بررسی الگوریتم

AES از یک بلوک دیتا با طول ۱۲۸ بیت و یک کلید که میتواند ۱۲۸، ۱۹۲، و یا ۲۵۶ بیت باشد استفاده می‌کند. در این بررسی طول کلید را ۱۲۸ بیت فرض می‌کنیم که احتمالاً یکی از پراستفاده‌ترین آنها خواهد بود.

شکل ۵-۲ ساختار کلی AES را نشان می‌دهد. ورودی الگوریتم‌های رمزنگاری و رمزگشائی یک بلوک منفرد ۱۲۸- بیتی است. در FIPS PUB 197، این بلوک بصورت یک ماتریس مربعی از بایت‌ها تعریف شده است. این بلوک در رشتهٔ **state** کپی شده که در هر مرحلهٔ رمزنگاری یا رمزگشائی تعدیل می‌شود. بعد از آخرین مرحله، **state** در ماتریس خروجی کپی می‌شود. بطریق مشابه، کلید ۱۲۸- بیتی بصورت یک ماتریس مربعی از بایت‌ها تعریف می‌شود. این کلید سپس بر اساس برنامهٔ کلید(key schedule) گسترش می‌یابد. هرکلمه شامل ۴ بایت بوده و کل برنامه کلید، ۴۴ کلمه برای یک کلید ۱۲۸- بیتی را تولید می‌کند. نظم بایت‌ها در یک ماتریس، ستونی است. بنابراین برای مثال چهار بایت اول ورودی متن ساده ۱۲۸- بیتی به رمزنگار، اولین ستون ماتریس **ورودی**، چهار بایت دوم ستون دوم و غیره را تشکیل می‌دهد. بطریق مشابه، اولین ۴ بایت کلید گسترش یافته که یک کلمه را تشکیل می‌دهد، اولین ستون ماتریس **w** را می‌سازد.



شکل ۵-۲ رمزنگاری و رمزگشائی AES

موارد ذیل، نکاتی در مورد AES را روشن می‌سازد:

۱- یکی ازخصوصیات قابل توجه این ساختار این است که یک ساختار Feistel نیست. بخاطر آورید که در ساختار کلاسیک Feistel، یک نیمه از بلوک دیتا برای تغییر نیمهٔ دیگر بکار میرفت و آنگاه دو نیمه جای خود را عوض می‌کردند. AES از ساختارFeistel استفاده نکرده بلکه در هر دُور، کل بلوک دیتا را بصورت موازی بکارگرفته و جایگزینی و جابجائی را در آن انجام می‌دهد.

۲- کلیدی که در ورودی فراهم میشود، بصورت یک رشتهٔ ۴۴- تائی از کلمات ۳۲ بیتی **w**[i] گسترش می‌یابد. در هر دُور ۴ کلمهٔ مجزا (۱۲۸ بیت) بعنوان کلید دُور مورد استفاده قرار می‌گیرد.

۳- از چهار عمل مختلف که یکی از آنها جابجائی و سه‌تای دیگر جایگزینی است استفاده می‌شود:

- **بایت‌ها جابجا شوند**: از یک جدول که S-box نامیده می‌شود استفاده کرده تا بایت به بایت بلوک را جابجا کند.
- **سطرها شیفت داده شوند**: یک جابجائی ساده که ردیف به ردیف انجام می‌شود.
- **ستون‌ها مخلوط شوند**: یک جابجائی که هر بایت یک ستون را بصورت تابعی از تمام بایت‌های همان ستون تغییر می‌دهد.
- **کلید دُور اضافه شود**: یک XOR ساده که بیت‌های بلوک فعلی را با بخشی از کلید گسترش یافته XOR نماید.

۴- ساختار کاملاً ساده است. هم برای رمزنگاری و هم برای رمزگشائی، رمز با یک مرحله اضافه کردن کلید دُور (Add Round Key) شروع شده و بدنبال آن با نه دُور دیگر که هرکدام شامل چهار مرحله است ادامه یافته و در انتها با سه مرحله در دُور دهم خاتمه می‌یابد. شکل ۶-۲ سازمان یک دُور رمزنگاری کامل را نشان می‌دهد.

۵- تنها مرحلهٔ Add Round Key از کلید استفاده می‌کند. بهمین دلیل رمز با مرحلهٔ Add Round Key شروع و خاتمه می‌یابد. هر مرحلهٔ دیگر بدون نیاز به کلید قابل برگشت بوده و بنابراین چیزی به امنیت اضافه نمی‌کند.

۶- مرحلهٔ Add Round Key به‌تنهائی نیرومند نیست. سه مرحلهٔ دیگر بیت‌ها را مخلوط کرده ولی خود امنیتی را ایجاد نمی‌نمایند، زیرا از کلید استفاده نمی‌کنند. میتوان رمز را بصورت رمزنگاری XOR (Add Round Key) یک بلوک و پس از آن درهم ریختن بلوک (سه مرحلهٔ دیگر) و بدنبال آن رمزنگاری XOR و غیره در نظر گرفت. این روش هم بهره‌ور و هم بغایت امن است.

۷- هر مرحله به آسانی برگشت‌پذیر است. برای مراحل جابجائی بایت، شیفت ردیف، و مخلوط کردن ستون، یک تابع معکوس در الگوریتم رمزگشائی بکار رفته است. برای مرحلهٔ Add Round Key، عمل عکس با XOR کردن همان کلید دور به بلوک حاصل می‌گردد زیرا $A \oplus A \oplus B = B$ است.

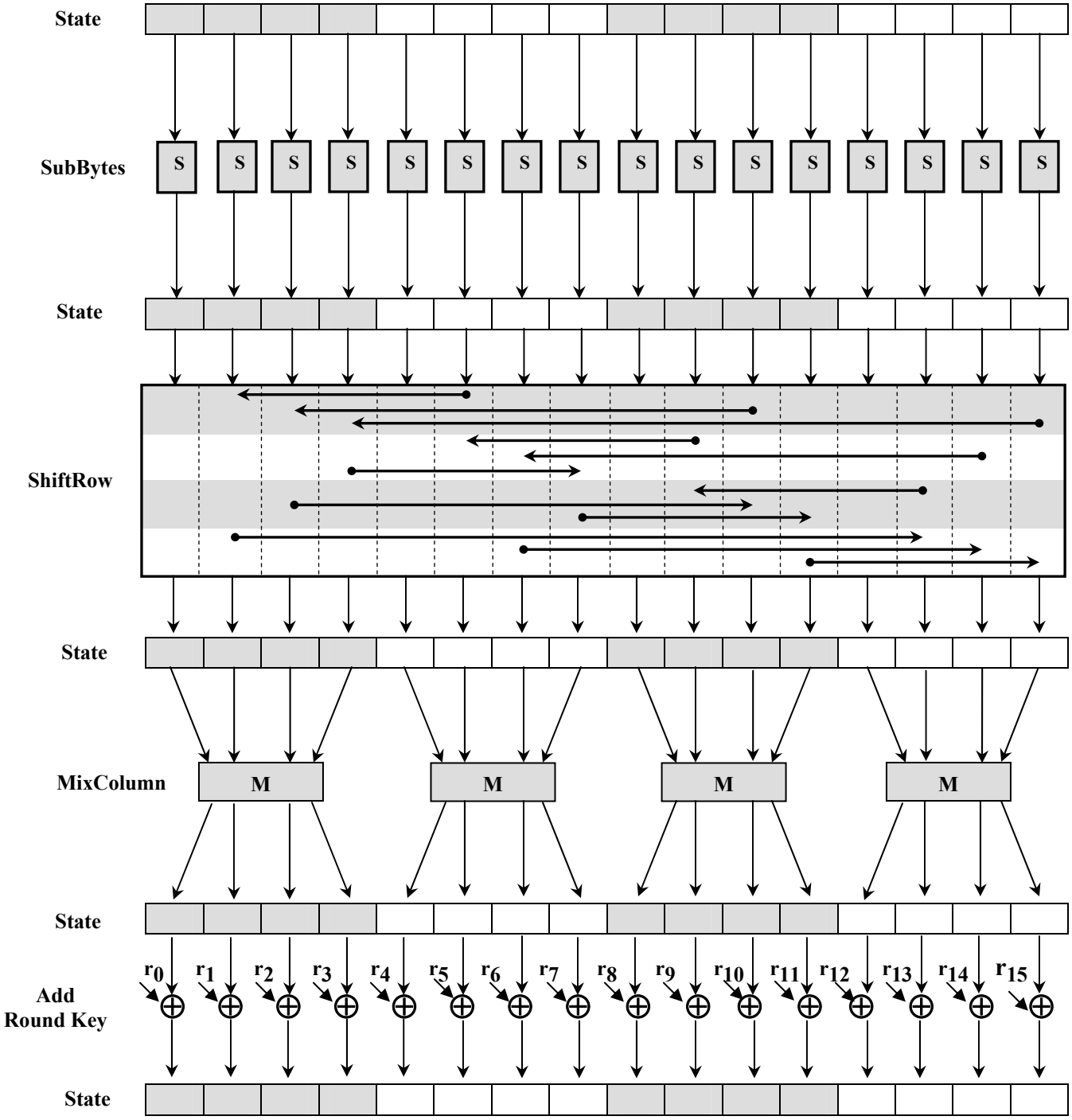
۸- همانند اکثر رمزهای قالبی، الگوریتم رمزگشائی از کلید گسترش‌یافته با نظم معکوس استفاده می‌کند. با وجود این الگوریتم رمزگشائی شبیه الگوریتم رمزنگاری نیست. این نتیجهٔ ساختار خاص AES است.

۹- وقتی روشن شد که هر چهار مرحله بازگشت‌پذیر هستند، آنگاه تأئید اینکه رمزگشائی متن ساده را احیاء خواهد کرد، آسان خواهد بود. شکل ۵-۲ رمزنگاری و رمزگشائی را در دو ستون کنارهم با جهت‌های مختلف نشان داده است. در هر سطح افقی (مثل خط‌چین‌ها در شکل)، state برای هم رمزنگاری و هم رمزگشائی یکی است.

۱۰ - دور نهائی چه در عمل رمزنگاری و چه در عمل رمزگشائی فقط دارای سه مرحله است. بازهم این نتیجهٔ ساختار خاص AES است و لازم است چنین باشد تا رمز بازگشت‌پذیر باشد.

۲-۳ رمزهای دنباله‌ای و RC4

یک *رمز قالبی* در هر زمان یک بلوک از عناصر ورودی را پردازش نموده و یک بلوک خروجی برای آن بلوک ورودی تولید می‌کند. یک *رمز دنباله‌ای*، عناصر ورودی را بطور پیوسته پردازش کرده و همینطور که جلو می‌رود عنصر به عنصر متن رمزشده را تولید می‌کند. اگرچه رمزهای قالبی بسیار متداول‌ترند، ولی در برخی کاربردها یک رمز دنباله‌ای گزینه‌ای مناسب‌تر است. مثال‌هائی از این کاربردها را در بخش‌های بعدی معرفی خواهیم کرد. در این بخش به متداول‌ترین رمز دنباله‌ای متقارن یعنی RC4 نگاهی می‌اندازیم. ابتدا مروری بر ساختار رمزهای دنباله‌ای داشته و سپس RC4 را بررسی خواهیم کرد.



شکل ۶-۲ یک دوّر عملیاتی AES

ساختار رمزهای دنباله‌ای

یک رمز دنباله‌ای معمولاً متن ساده را بصورت بایت- به- بایت رمزنگاری می‌نماید. البته می‌توان رمزهای دنباله‌ای دیگری خلق کرد که داده‌ها را بصورت بیت- به- بیت و یا در واحدهای بزرگ‌تر از بایت در هر زمان رمزنگاری نماید. شکل ۷-۲ نمایش‌دهندهٔ ساختار یک رمز دنباله‌ای است. در این ساختار یک کلید، ورودی یک تولیدکنندهٔ شبه‌تصادفی بیت‌ها بوده که یک

دنبالهٔ ۸- بیتی که ظاهراً تصادفی به نظر می‌رسد را تولید می‌کند. خروجی تولیدکنندهٔ شبه تصادفی، که یک **دنبالهٔ کلید (keystream)**، نامیده می‌شود با دنبالهٔ متن سادهٔ ورودی بصورت یک بایت در هر زمان و بصورت عمل XOR روی بیت‌ها ترکیب می‌شود. برای مثال اگر بایت تولیدشده بتوسط مولد 01101100 و بایت متن ساده 11001100 باشد، آنگاه بایت متن رمزشدهٔ حاصل چنین است:

11001100 ⊕	متن ساده
01101100	دنبالهٔ کلید
10100000	متن رمزشده

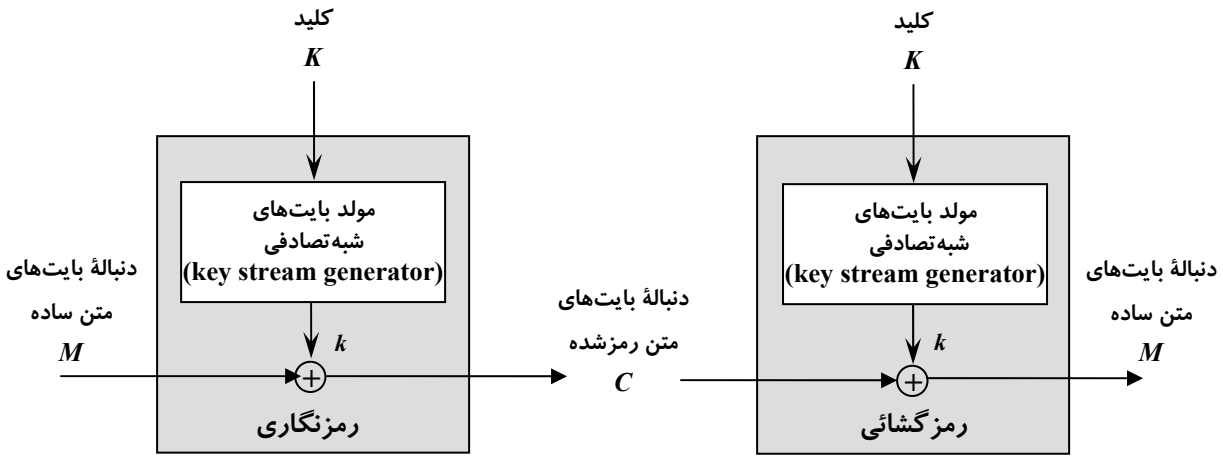
رمزگشائی نیاز به استفاده از همان ردیف شبه‌تصادفی را دارد.

10100000 ⊕	متن رمزشده
01101100	دنبالهٔ کلید
11001100	متن ساده

[KUMA97] ملاحظات مهم زیر در طراحی یک رمز دنباله‌ای را ذکر کرده است:

۱- دنبالهٔ رمز بایستی دارای دورهٔ تناوب بزرگی باشد. یک تولیدکنندهٔ اعداد شبه‌تصادفی از تابعی استفاده می‌کند که یک دنبالهٔ یقینی از بیت‌ها را تولید کرده که نهایتاً بعداز مدتی تکرار می‌شوند. هرچقدر دورهٔ تناوب این تکرار طویل‌تر باشد، عمل شکستن رمز سخت‌تر خواهد بود.

۲- دنبالهٔ کلید بایستی با تقریب بسیار خوب، خواص یک دنبالهٔ عدد تصادفی واقعی را داشته باشد. بعنوان مثال تقریباً بایستی تعداد 1ها و 0ها در این دنباله برابر باشند. اگر دنباله کلید بصورت یک ردیف از بایت‌ها مورد استفاده قرار گیرد، آنگاه تمام ۲۵۶ حالت ممکن بایستی تقریباً بصورت مساوی مورد استفاده قرار گیرند. هر چقدر دنبالهٔ کلید تصادفی‌تر بنظر آید، متن رمزشده تصادفی‌تر بوده و شکستن رمز سخت‌تر خواهد بود.



شکل ۷-۲ دیاگرام رمز دنباله‌ای

۳- با توجه به شکل ۷-۲ می‌توان دریافت که خروجی یک مولد اعداد شبه‌تصادفی به اندازهٔ کلید ورودی وابسته است. برای جلوگیری از حملات همه جانبه، کلید بایستی به اندازهٔ کافی بزرگ باشد. همان ملاحظاتی که در مورد رمزهای قالبی وجود داشت در اینجا نیز صادق‌اند. بنابراین با تکنولوژی کنونی، کلیدی با طول حداقل ۱۲۸ بیت مناسب بنظر می‌رسد.

با یک مولد اعداد شبه‌تصادفی با طرح مناسب، یک رمز دنباله‌ای می‌تواند بهمان اندازهٔ یک رمز قالبی، با همان طول کلید، امن باشد. مزیت اصلی یک رمز دنباله‌ای این است که رمزهای دنباله‌ای تقریباً همیشه سریع‌تر بوده و نسبت به رمزهای قالبی از حجم برنامهٔ کمتری استفاده می‌کنند. رمز RC4 که در این بخش تشریح شده است تنها می‌تواند با چند خط برنامهٔ کامپیوتری پیاده‌سازی شود. جدول ۳-۲ که از اطلاعات [RESC01] اقتباس شده است، زمان اجرای RC4 را با سه رمز قالبی معروف مقایسه کرده است. حسن یک رمز قالبی در این است که شما می‌توانید از کلید رمز بارها استفاده کنید. در رمز دنباله‌ای اگر دو متن ساده با یک کلید یکسان رمزنگاری شوند، آنگاه شکستن رمز غالباً بسیار آسان خواهد بود [DAWS96]. اگر دو دنبالهٔ متن رمزشده با هم XOR شوند، نتیجه با XOR دو متن سادهٔ نظیر آنها یکسان خواهدبود. حال اگر متن ساده، دنبالهٔ کوتاهی همانند شمارهٔ کارت‌های اعتباری و یا ردیف‌های دیگری با خواص شناخته شده باشند، عمل شکستن رمز ممکن است موفقیت‌آمیز باشد.

برای کاربردهائی همانند کانال‌های مخابرهٔ داده‌ها و یا مرور لینک‌های وِب که نیاز به رمزنگاری/ رمزگشائی دنباله‌های دیتا دارند، یک رمز دنباله‌ای می‌تواند گزینهٔ بهتری باشد. برای کاربردهائی همچون انتقال فایل، پست الکترونیک و پایگاه داده که با بلوک‌های دیتا سروکار دارند، رمزهای قالبی می‌توانند مناسب‌تر باشند. با وجود این هر دو نوع رمز تقریباً در هر کاربردی قابل استفاده‌اند.

الگوریتم RC4

RC4 یک رمز دنباله‌ای است که در سال ۱۹۸۷ میلادی بتوسط Ron Rivest برای کمپانی RSA Security طراحی گردید. RC4 یک رمز دنباله‌ای با طول کلید متغیر بوده و عملیات آن روی بایت‌ها انجام می‌شود. الگوریتم بر مبنای استفاده از یک جایگشت تصادفی بنا نهاده شده است. تحلیل این رمز نشان می‌دهد که دورهٔ تناوب رمز با احتمال قریب به یقین بزرگتر از $۱۰^{۱۰۰}$ است [ROBS95a]. برای تولید هر بایت خروجی بین ۸ تا ۱۶ عمل لازم است و انتظار می‌رود که رمزنگاری در نرم‌افزار به سرعت انجام شود. RC4 در استانداردSSL/TLS (Secure Socket Layer/Transport Layer Security) که برای ارتباط بین مرورگرهای وِب و سِرورها تعریف شده است، بکار می‌رود. این رمز همچنین در پروتکل WEP (Wired Equivalent Privacy) و پروتکل جدیدتر WPA (WiFi Protected Access) که بخشی از استانداردهای IEEE 802.11 مربوط به LAN بی‌سیم هستند، مورد استفاده است. RC4 از نظر تجاری مدتها از سوی کمپانی RSA Security پنهان نگاه داشته شده بود. در سپتامبر ۱۹۹۴ این الگوریتم بصورت ناشناس در لیست پستی Cypherpunks قرار گرفت و لو رفت.

الگوریتم RC4 بصورت قابل توجهی ساده بوده و تشریح آن کاملاً آسان است. یک کلید با طول متغیر ۱ تا ۲۵۶ بایت (۸ تا ۲,۰۴۸ بیت) برای آغازیدن یک بردار حالت ۲۵۶- بایتی S با مؤلفه‌های S[0]، S[1]..... S[255] مورد استفاده قرار می‌گیرد. در همهٔ حالات، S شامل جایگشت همهٔ اعداد ۸- بیتی از صفر تا ۲۵۵ است. برای رمزنگاری و رمزگشائی، یک بایت k (شکل ۷-۲ را ببینید) از میان ۲۵۵ مؤلفهٔ S بصورت سیستماتیک انتخاب می‌شود. همینطور که هر مقدار k تولید می‌شود، مؤلفه‌های S یک‌بار دیگر جایگشت می‌یابند.

جدول ۳-۲ مقایسهٔ سرعت پـــردازش رمزهای متقارن روی یک پـــــردازش‌گر Pentium II

نوع رمز	طول کلید	سرعت (Mbps)
DES	۵۶	۹
3DES	۱۶۸	۳
RC2	متغیر	۰/۹
RC4	متغیر	۴۵

آغازیدن S

برای شروع، مقادیر صفر تا ۲۵۵ بصورت صعودی در مؤلفه‌های S قرار داده می‌شـود، یعنـی S[0] = 0، S[1] = 1 و S[255] = 255. همچنین یک بردار موقت T خلق می‌شود. اگر طول کلید K برابر ۲۵۶ بایت باشد، آنگـاه K بـه T منتقـل می‌شود. در غیر اینصورت برای کلیدی با طول keylen بایت، اولین مؤلفه‌های T از K کپی شـده و سـپس K هـر چنـدبار لازم باشد تکرار شده تا T پر شود. این عملیات ابتدائی را می‌توان چنین خلاصه کرد:

```
/* Initialization */
for I = 0 to 255 do
S[i] = i ;
T[i] = K[i mod keylen] ;
```

سپس از T برای جایگشت آغازین S استفاده می‌شود. این امر با S[0] شروع شده و تا S[255] ادامه می‌یابد. هر S[i] با بایت دیگری در S بر اساس روشی که بتوسط T[i] دیکته می‌شود تعویض می‌شود:

```
/* Initial Permutation of S */
J = 0 ;
for I = 0 to 255 do
j = ( j + S[i] + T[i] ) mod 256 ;
Swap (S[i] , S[j]) ;
```

چون تنها عمل روی S یک تعویض محل بایت‌هاست، تنها اثر این امر ایجاد یک جایگشت است. S همچنان شامل تمام اعداد بین صفر تا ۲۵۵ خواهد بود.

تولید دنباله

همین‌که بردار S با مقادیر اولیه پر شد، دیگر از کلید ورودی استفاده نخواهد شد. تولید دنباله شامل عبور از S[0] تا S[255] بوده و هر مقدار S[i] با بایت دیگری در S، برحسب قانونی که بتوسط وضع فعلی S دیکته می‌شود، جایگزین می‌گردد. بعد از اینکه به S[255] رسیدیم، پردازش با شروع مجدد از S[0] ادامه می‌یابد:

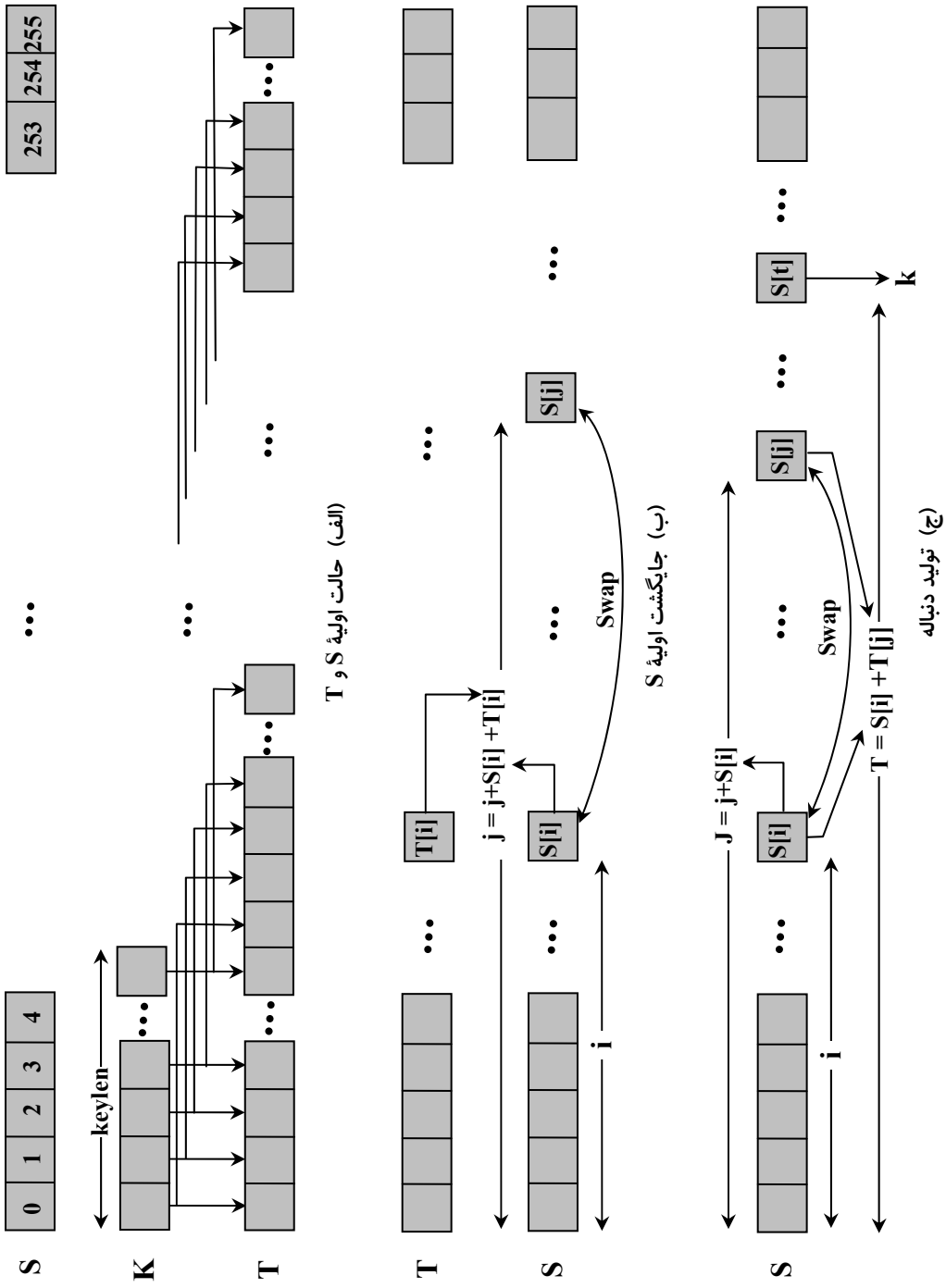
```
/* Stream Generation */
i , j = 0 ;
While (true)
I = (i+1) mod 256 ;
j = (j + S[i]) mod 256
Swap (S[i] , S[j]) ;
T = ( S[i] + S[j] ) mod 256 ;
K = S[t] ;
```

برای رمزنگاری، اندازهٔ k را با بایت بعدی متن ساده XOR می‌کنیم. برای رمزگشائی، اندازهٔ k را با بایت بعدی متن رمزشده XOR می‌کنیم.

شکل ۸-۲ منطق RC4 را نشان می‌دهد.

توانائی RC4

مقاله‌های متعددی نوشته شده‌اند که روش حمله به RC4 را تحلیل کرده‌اند (مثلاً [KNUD98]، [MIST98]، [FLUH00] و [MANT01]). هیچکدام از این روش‌ها برای حمله به RC4 با کلیدی که دارای طول منطقی همچون ۱۲۸ بیت باشد، عملی نیستند. یک مورد جدی‌تر در [FLUH01] مطرح گردید. نویسندگان مقاله نشان دادند که پروتکل WEP که برای فراهم نمودن محرمانگی در شبکه‌های LAN بی‌سیم از پروتکل 802.11 استفاده می‌کند، در برابر حملهٔ بخصوصی آسیب‌پذیر است. در واقع مشکل به RC4 ربطی نداشته بلکه به روشی که کلیدها برای استفاده در ورودی RC4 تولید می‌شوند، مرتبط است. این مشکل بخصوص در سایر کاربردهائی که از RC4 استفاده می‌کنند ظاهر نشده و در WEP نیز با تغییر روش تولید کلیدها، مشکل رفع خواهد شد. این مسأله، مشکل طراحی یک سیستم امن، که هم از تابع رمزنگاری و هم از پروتکل‌هائی که این توابع را بکار می‌گیرند استفاده می‌کند، را خاطرنشان می‌سازد.



شکل ۸-۲ RC4

۲-۴ مُودهای عملیاتی رمزهای قالبی

یک رمز قالبی متقارن، داده‌ها را بصورت یک بلوک در هر زمان پردازش می‌کند. در DES و 3DES طول بلوک ۶۴ بیت است. برای متون ساده با طول بیشتر، لازم است تا متن به بلوک‌های ۶۴- بیتی تقسیم شود (اگر لازم باشد، آخرین بلوک با بیت‌های اضافی کامل می‌شود). ساده‌ترین راه برای این کار چیزی است که آن را مُود کتاب کُد الکترونیکی electronic codebook (ECB) گویند که در آن در هر لحظه، ۶۴ بیت از متن ساده تحت پردازش قرار گرفته و همهٔ بلوک‌های متن با کلید واحدی پردازش می‌شوند. اصطلاح *کتاب کُد* (*codebook*) از این جهت بکار گرفته شده است که برای یک کلید واحد، یک متن رمزشدهٔ یکتا برای هربلوک ۶۴- بیتی دیتا حاصل می‌شود. بنابراین میتوان یک کتاب کُد عظیمی را تصور کرد که در آن برای هر بلوک ۶۴- بیتی ممکن از متن ساده، یک متن رمزشدهٔ نظیر آن وجود داشته باشد.

در ECB، اگر همان بلوک ۶۴- بیتی متن ساده بیش از یکبار در پیام ظاهر شود، همیشه همان متن رمزشدهٔ دفعهٔ اول حاصل خواهد شد. بهمین دلیل برای پیام‌های طولانی، مُود ECB ممکن است امن نباشد. اگر پیام بشدت ساختاریافته باشد، یک شکنندهٔ رمز ممکن است بتواند از این نظم سوءاستفاده کند. بعنوان مثال اگر معلوم باشد که پیام همیشه با میدان‌های از قبل تعریف شدهٔ معینی شروع می‌شود، آنگاه شکنندهٔ رمز ممکن است تعدادی زوج متن ساده- متن رمزشده را در اختیار داشته و روی آنها کار کند. اگر پیام دارای عناصر تکرارشده‌ای باشد که پریود تکرار آنها مضربی از ۶۴ بیت باشد، آنگاه این عناصر می‌توانند بتوسط تحلیل‌گر شناخته شوند. این موارد ممکن است به تحلیل رمز کمک کرده و یا ممکن است فرصتی برای جایگزینی و یا تغییر سازمان بلوک بدست دهند.

برای غلبه بر کمبودهای امنیتی ECB، علاقه‌مند به تکنیکی هستیم که با استفاده از آن، همان بلوک متن ساده در صورت تکرار، بلوک‌های رمزشدهٔ متفاوتی را ایجاد کند. در این قسمت به دو روش مختلف که در FIPS PUB 81 تعریف شده است، نگاهی می‌اندازیم.

مُود زنجیره‌ای رمز قالبی (Cipher Block Chaining Mode)

در مُود زنجیره‌ای رمز قالبی(CBC) ، (شکل ۹-۲)، ورودی الگوریتم رمزنگاری از XOR بلوک متن سادهٔ فعلی با بلوک متن رمزشدهٔ قبلی بدست می‌آید و از کلید واحدی نیز برای همهٔ مراحل استفاده شده است. اثر این امر این است که پردازش ردیف بلوک‌های متن ساده را بهم زنجیر کرده‌ایم. در نتیجه ورودی تابع رمزنگاری برای هر بلوک متن ساده، رابطهٔ ثابتی با خود بلوک متن ساده ندارد. به همین دلیل قالب‌های تکرار شده ۶۴- بیتی در خروجی ظاهر نخواهند شد.

برای رمزگشائی، هر بلوک رمزشده از الگوریتم رمزگشائی عبور می‌کند. نتیجهٔ این عمل با بلوک متن رمزشدهٔ قبلی XOR شده تا بلوک متن ساده بدست آید. برای اطمینان از صحت این روش، می‌توان نوشت:

$$C_i = \mathrm{E}\left([K, [C_{i-1} \oplus P_i]\right)$$

که درآن $\mathrm{E}[K, X]$ رمزنگاری متن سادهٔ X با استفاده از کلید K بوده و $\oplus$ عمل XOR را نشان میدهد. آنگاه

$$\mathrm{D}[K, C_i] = \mathrm{D}(K, \mathrm{E}(K, [C_{i-1} \oplus P_i]))$$
$$\mathrm{D}[K, C_i] = C_{i-1} \oplus P_i$$
$$C_{i-1} \oplus \mathrm{D}(\mathrm{K}, C_i) = C_{i-1} \oplus C_{i-1} \oplus P_i = P_i$$

که شکل ۹-۲ب را تأئید می‌کند.

برای تولید اولین بلوک متن رمزشده، یک بردار آغازگر Initialization Vector (IV) با اولین بلوک متن ساده XOR می‌شود. در زمان رمزگشائی، IV با خروجی الگوریتم رمزگشائی XOR شده تا اولین بلوک متن ساده بدست آید.

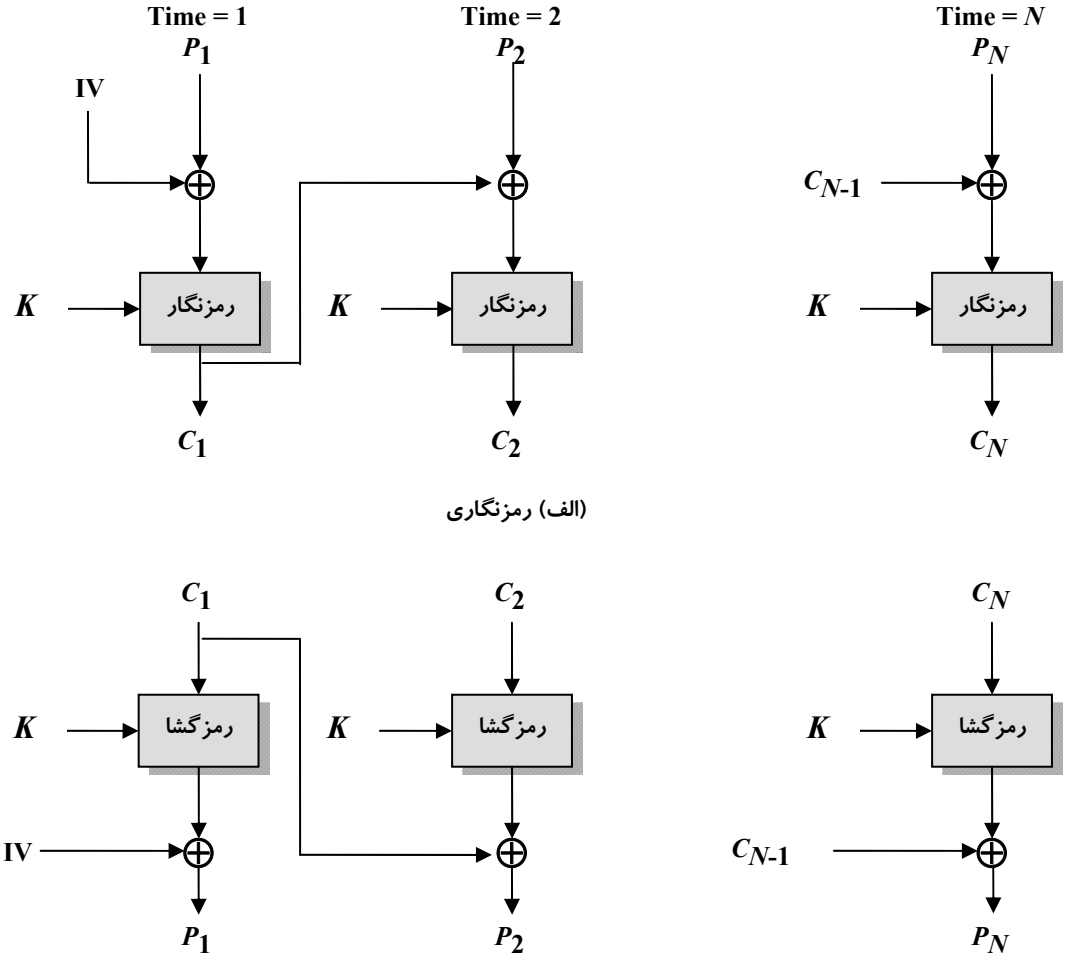
IV بایستی هم برای فرستنده و هم برای گیرنده شناخته شده باشد. برای ایجاد ماکزیمم امنیت، از IV نیز همانند کلید بایستی محافظت نمود. یکی از دلایل حفاظت از IV این است: اگر یک دشمن بتواند گیرنده را به استفاده از مقدار دیگری برای IV وادار کند، آنگاه دشمن خواهد توانست تا بیت‌های انتخاب‌شده در اولین بلوک متن ساده را معکوس نماید. برای روشن شدن مطلب فرض کنید:

$$C_1 = \mathrm{E}\,(K, [\mathrm{IV} \oplus P_1])$$

$$P_1 = \mathrm{IV} \oplus \mathrm{D}\,(K, C_1)$$

با استفاده از نمایش $X[j]$ بعنوان j امین بیت مقدار ۶۴- بیتی X ، آنگاه

$$P_1[j] = \mathrm{IV}[j] \oplus \mathrm{D}\,(K, C_1)[j]$$



شکل ۹-۲ مُود زنجیره‌ای رمز قالبی (CBC)

سپس با استفاده از خواص عمل XOR می‌توان گفت

$$P_1[j]' = \mathrm{IV}[j]' \oplus \mathrm{D}(K, C_1)[j]$$

که علامت پریم(') نمایش یک بیت نفی شده است. این بدین معنی است که اگر دشمن با تخمین بتواند بیت‌های IV را عوض کند، بیت‌های نظیر اندازهٔ دریافت شده P_1 می‌توانند تغییر یابند.

CBC کاربرد گسترده‌ای در مسائل امنیتی دارد که بعداً به آن اشاره خواهد شد.

مُود فیدبک رمز (Cipher Feedback Mode)

این امکان وجود دارد که با استفاده از مُود فیدبک رمز (CFB)، هر رمز قالبی را بصورت یک رمز دنباله‌ای درآورد. در یک رمز دنباله‌ای نیازی نیست که با اضافه کردن بیت‌ها به پیام، تا حد مضربی از بلوک‌ها، آن را کامل کرد. همچنین این رمز میتواند در حالت بلادرنگ کار کند. بنابراین اگر دنباله‌ای از کاراکترها قرار است ارسال شوند، هر کاراکتر میتواند با استفاده از یک رمز دنباله‌ای با گرایش کاراکتری، بلافاصله رمزنگاری و ارسال گردد.

یکی از خصوصیات مطلوب یک رمز دنباله‌ای این است که متن رمزشده دارای همان طول متن ساده است. بنابراین اگر کاراکترهای ۸- بیتی ارسال می‌گردند، هر کاراکتر بایستی با ۸ بیت رمزنگاری شود. اگر بیش از ۸ بیت مورد استفاده قرار گیرد، ظرفیت انتقال تلف خواهد شد.

شکل ۱۰-۲ روش CFB را به تصویر کشیده است. در این شکل فرض شده است که واحد انتقال s بیت است که اندازهٔ معمول آن ۸ میباشد. همانند CBC، واحدهای متن ساده بهم زنجیر شده‌اند بطوری‌که متن رمزشده هر واحد متن ساده، تابعی از تمام متون سادهٔ قبلی است.

در ابتدا رمزنگاری را در نظر بگیرید. ورودی تابع رمزنگاری یک شیفت رجیستر ۶۴- بیتی است که در ابتدا با یک بردار اولیه(IV) پر می‌شود. چپ‌ترین (با اهمیت‌ترین) s بیت خروجی تابع رمزنگاری با اولین واحد متن سادهٔ P_1 بصورت XOR درآمده تا اولین واحد متن رمزشدهٔ C_1 را که متعاقباً ارسال خواهد شد تشکیل دهد. علاوه برآن، محتویات شیفت رجیستر باندازهٔ s بیت به چپ شیفت داده شده و C_1 در راست‌ترین (کم اهمیت‌ترین) s بیت شیفت رجیستر جای می‌گیرد. این امر تا وقتی که تمام واحدهای متن ساده رمزنگاری شوند، ادامه می‌یابد.

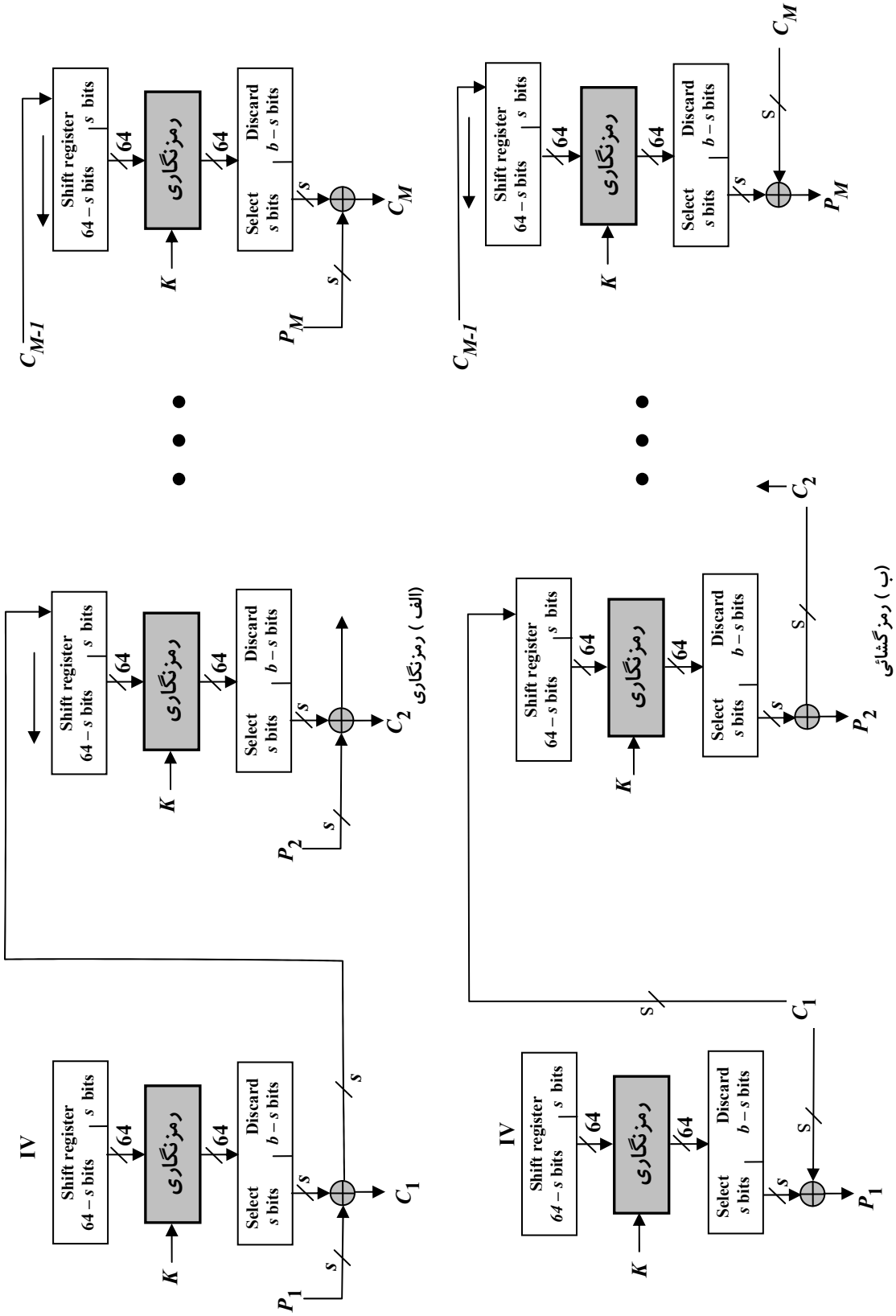
برای رمزگشائی، همین روش مورد استفاده قرار می‌گیرد بجز اینکه متن رمزشده دریافت‌شده با خروجی تابع رمزنگاری XOR شده تا متن ساده را تولید کند. توجه شود که این تابع *رمزنگاری* است که مورد استفاده قرار میگیرد نه تابع رمزگشائی. این مسأله بسادگی قابل توضیح است . فرض کنید $\mathrm{S_S}(X)$ بعنوان با اهمیت‌ترین s بیت X تعریف شود. آنگاه

$$C_1 = P_1 \oplus \mathrm{S_s}[\mathrm{E}(K, \mathrm{IV})]$$

بنابراین

$$P_1 = C_1 \oplus \mathrm{S_s}[\mathrm{E}(\mathrm{K}, \mathrm{IV})]$$

همین استدلال برای قدم‌های بعدی این پردازش نیز صادق است.



شکل ۱۰-۲ مُود فیدبک رمز(CFB) با اندازهٔ s بیت

۲-۵ محل استقرار تجهیزات رمزنگاری

قوی‌ترین و معمول‌ترین روش برای مقابله با حملات امنیتی به شبکه، رمزنگاری است. در استفاده از رمزنگاری لازم است تصمیم بگیریم که چه چیزی را رمزنگاری کرده و لوازم مربوط به رمزنگاری را در کجا قرار دهیم. در این مورد دو انتخاب اصلی وجود دارد: رمزنگاری پیوند(link encryption)، و رمزنگاری سر- به- سر (end-to-end encryption) که استفاده از آنها در عرض یک شبکهٔ سوئیچ بسته‌ای در شکل ۱۱-۲ نشان داده شده است.

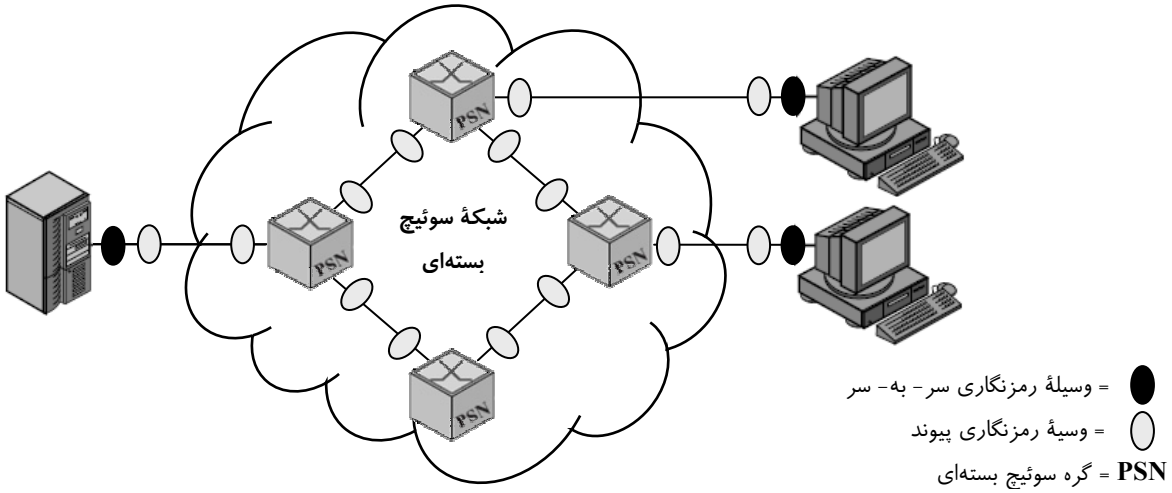
در رمزنگاری پیوند، هر پیوند مخابراتی آسیب‌پذیر، در هر یک از دو انتها با یک وسیلهٔ رمزنگاری تجهیز می‌شود. بنابراین کل ترافیک روی تمام پیوندهای مخابراتی امن خواهند شد. اگرچه در یک شبکهٔ وسیع، این روش نیاز به تعداد زیادی تجهیزات رمزنگاری دارد، ولی در عین حال سطح بالائی از امنیت را ایجاد خواهد کرد. یکی از معایب این روش این است که پیام هر بار که وارد یک سوئیچ بسته‌ای میشود، بایستی رمزگشائی گردد. علت این امر این است که سوئیچ بایستی آدرس موجود در سرآیند بسته (شمارهٔ مدار مجازی) را خوانده تا بتواند آن را مسیریابی نماید. بهمین دلیل پیام از نظر امنیتی در محل سوئیچ آسیب‌پذیر خواهد بود. اگر این شبکه، یک شبکهٔ سوئیچ بسته‌ای همگانی باشد، کاربر کنترلی بر امنیت گره‌ها نخواهد داشت.

در رمزنگاری سر- به- سر، رمزنگاری در دو سیستم انتهائی صورت می‌پذیرد. میزبان یا پایانه منبع، داده‌ها را به رمز در می‌آورد. آنگاه داده‌ها، با فرم رمزنگاری شده و بدون تغییر در عرض شبکه به پایانه و یا میزبان مقصد عبور می‌کند. مقصد با کلیدی که همانند کلید منبع است، پیام را رمزگشائی می‌کند. بنظر میرسد که این روش، انتقال پیام در مقابل حملاتی که به پیوندها و یا سوئیچ‌ها می‌شود را تضمین می‌نماید. با وجود این هنوز یک نقطهٔ سُست باقی است.

حالت زیر را در نظر بگیرید. یک میزبان به یک شبکهٔ سوئیچ بسته‌ای X.25 وصل شده، یک مدار مجازی با میزبان دیگری را برقرار کرده و آماده است تا دیتا را با استفاده از رمزنگاری سر- به- سر برای میزبان دیگر بفرستد. دیتا روی چنین شبکه‌ای بصورت بسته‌هائی منتقل می‌گردد که شامل یک سرآیند و بخش داده‌هاست. میزبان کدام بخش را باید رمزنگاری نماید؟ فرض کنید که میزبان کل بسته که شامل سرآیند نیز هست را رمزنگاری کند. این امر عملی نخواهد بود، زیرا فراموش نکنید که تنها میزبان انتهائی قادر به رمزگشائی است. دراین حالت، گرهٔ سوئیچ بسته‌ای که یک بستهٔ رمزنگاری شده را دریافت می‌کند قادر به خواندن سرآیند آن نبوده و بنابراین نخواهد توانست تا آن را مسیریابی نماید. پس چنین بنظر میرسد که میزبان مبدأ فقط می‌تواند بخش داده‌های بسته دیتا را رمزنگاری کرده و بایستی بخش سرآیند را آزاد گذاشته تا شبکه بتواند با خواندن آن بسته را به مسیر صحیح هدایت کند.

بنابراین در رمزنگاری سر- به- سر، داده‌های کاربر امن می‌مانند. اما چون سرآیند بسته‌ها بصورت ساده منتقل می‌گردند، اُلگوی ترافیک امن نخواهد بود. برای ایجاد امنیت بیشتر، هم رمزنگاری پیوند و هم رمزنگاری سر- به- سر مورد نیازند که این مطلب در شکل ۱۱-۲ نشان داده شده است.

بطور خلاصه، هر وقت از هر دو فرم رمزنگاری استفاده می‌شود، میزبان مبدأ در ابتدا بخش داده‌های کاربر در یک بستهٔ دیتا را با استفاده از یک کلید رمزنگاری سر- به- سر رمزنگاری می‌کند و سپس تمام بسته با استفاده از یک کلید رمزنگاری پیوند به رمز درمی‌آید. همینطور که بسته دیتا در عرض شبکه عبور می‌نماید، هر سوئیچ، بسته را با استفاده از یک کلید رمزنگاری پیوند رمزگشائی کرده تا سرآیند آن را خوانده و سپس مجدداً تمام بسته را برای ارسال روی پیوند بعدی مسیر رمزنگاری می‌نماید. بدین ترتیب تمام یک بستهٔ دیتا، مگر در زمانی که بسته در حافظهٔ یک سوئیچ بسته‌ای قرار داشته، امن است که فقط در آن زمان سرآیند بسته بصورت رمزنشده قابل مشاهده خواهد بود.



شکل ۱۱-۲ رمزنگاری در عرض یک شبکهٔ سوئیچ بسته‌ای

۲-۶ توزیع کلید

برای اینکه رمزنگاری متقارن عملی گردد، طرفین ارتباط بایستی دارای کلید رمز واحدی بوده و این کلید بایستی از دست‌یابی دیگران محافظت گردد. علاوه بر این معمولاً لازم است تا مکرراً کلید را تعویض کرده تا احتمال فاش شدن داده‌ها، در صورت دست‌یابی یک دشمن به کلید، را به حداقل برسانیم. بنابراین قدرت یک سیستم رمزنگاری مرتبط با روش توزیع کلید است. اصطلاح «توزیع کلید» به روش تحویل کلید به دو طرفی اشاره می‌کندکه تمایل به مبادلهٔ دیتا دارند، بدون اینکه دیگران بتوانند کلید را مشاهده نمایند. توزیع کلید را میتوان به چند صورت انجام داد. برای دو طرف A و B:

۱- کلید میتواند بتوسط A انتخاب شده و بصورت فیزیکی به B تحویل گردد.

۲- شخص ثالثی میتواند کلید را انتخاب کرده و بصورت فیزیکی آن را به A و B تحویل دهد.

۳- اگر A و B قبلاً و اخیراً از کلیدی استفاده می‌کرده‌اند، یکی از طرفین میتواند کلید جدید را انتخاب کرده و آن را بصورت رمزنگاری شده با استفاده از کلید قدیم، به طرف دیگر تحویل دهد.

۴- اگر A و B هرکدام یک ارتباط رمزنگاری شده با شخص ثالث C دارند، C می‌تواند یک کلید را از طریق پیوندهای رمزنگاری شده با A و B به آنها تحویل دهد.

روش‌های ۱ و ۲ نیاز به تحویل دستی یک کلید دارند. در رمزنگاری پیوند، این یک نیاز معقول است زیرا هر دستگاه رمزنگاری پیوند تنها می‌خواهد داده‌ها را با شریک خود در طرف دیگر پیوند مبادله نماید. ولی در رمزنگاری سر- به- سر تحویل دستی غیرمعقول است. در یک سیستم توزیع شده، هر میزبان و یا پایانه ممکن است نیاز داشته باشد تا در مبادلهٔ کلید با میزبانان بسیار دیگری مشارکت کند. بنابراین هر دستگاه نیاز به تعدادی کلید داشته که بایستی در طول زمان بصورت پویائی تولید شوند. این مشکل علی‌الخصوص در یک سیستم توزیع شدهٔ پهناور حادتر است.

روش۳ امکانی است که هم برای رمزنگاری پیوند و هم برای رمزنگاری سر- به- سر وجود دارد ولی اگر یک حمله‌کننده یکبار موفق به دست‌یابی به کلیدی گردد، آنگاه همهٔ کلیدهای بعدی نیز لو خواهند رفت. حتی اگر در مورد کلیدهای رمزنگاری پیوند، تعویض‌های مکرری صورت پذیرد، این کار بایستی بطور دستی انجام شود. در مورد تهیهٔ کلیدها برای رمزنگاری سر- به- سر، روش ۴ دارای ارجحیت بیشتری است.

شکل۱۲-۲ روشی را نشان میدهد که گزینه ۴ برای رمزنگاری سر- به- سر را عملی نموده است. در این شکل از رمزنگاری پیوند صرفنظر شده است. این مورد را میتوان بر حسب نیاز اضافه کرد و یا نکرد. در این روش دو نوع کلید تعریف شده است:

- **کلید اجلاس**: وقتی دو سیستم انتهائی (میزبان، پایانه و غیره) تمایل به ارتباط دارند، آنها یک اتصال منطقی (مثل مدار مجازی) را ایجاد می‌کنند. در خلال تداوم اتصال منطقی، تمام داده‌های کاربر با یک کلید اجلاس یکبارمصرف رمزنگاری می‌شود. در خاتمهٔ گفتگو، کلید اجلاس معدوم می‌گردد.
- **کلید دائم**: یک کلید دائم کلیدی است که برای توزیع کلیدهای اجلاس بین واحدها بکار میرود.

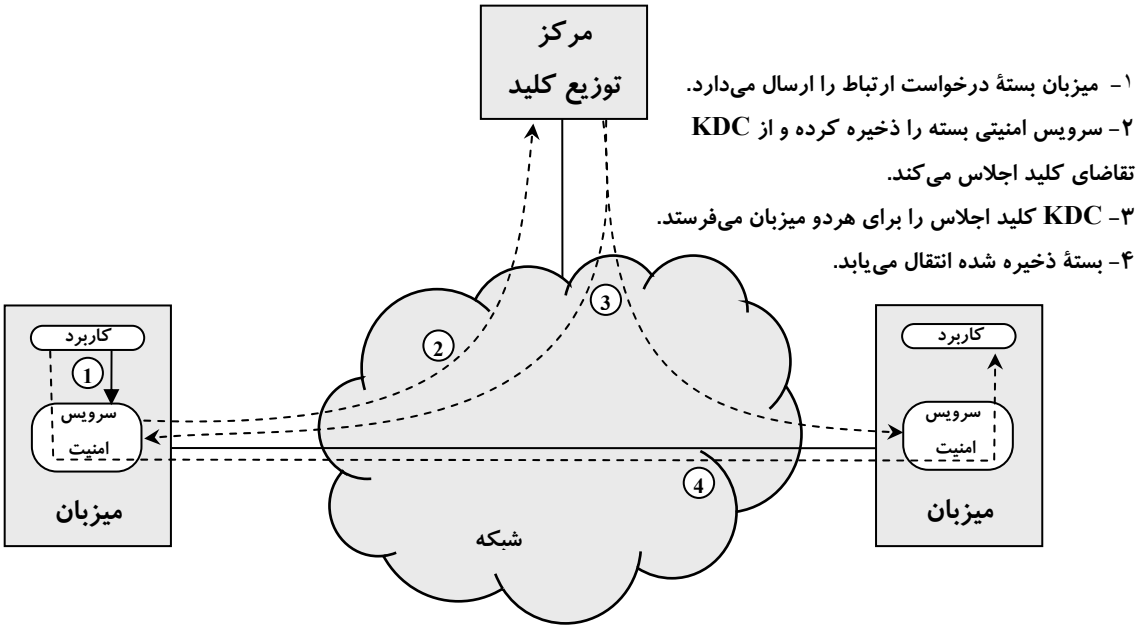
پیکربندی شکل ۱۲-۲ شامل عناصر زیر است:

- **مرکز توزیع کلید(KDC)**: مرکز توزیع کلید تعیین می‌کند که کدام سیستم‌ها مجاز به ارتباط با یکدیگرند. وقتی به دو سیستم اجازه داده شد تا با هم ارتباط یابند، مرکز توزیع کلید یک کلید اجلاس یکبارمصرف را برای این ارتباط فراهم می‌آورد.
- **مدول سرویس امنیتی (SSM)**: این مدول که ممکن است شامل عملکرد در یک لایه پروتکلی باشد، رمزنگاری سر- به- سر را انجام داده و کلید اجلاس را از جانب کاربران بدست می‌آورد.

قدم‌هائی که برای برقراری ارتباط برداشته می‌شود در شکل ۱۲-۲ نشان داده شده است. وقتی یک میزبان می‌خواهد تا ارتباطی با میزبان دیگر برقرار سازد، یک بستهٔ درخواست ارتباط را ارسال میکند (قدم اول). SSM آن بسته را ذخیره کرده و از KDC اجازه میخواهد تا ارتباط را برقرار کند (قدم دوم). ارتباط بین SSM و KDC با یک کلید اصلی که تنها در اختیار SSM و KDC است رمزنگاری می‌شود. اگر KDC تقاضای اتصال را بپذیرد، یک کلید اجلاس تهیه کرده و آن را با استفاده از یک کلید یکتای دائم برای هر SSM، به دو SSM ذیربط تحویل می‌دهد (قدم سوم). اکنون SSM متقاضی ارتباط می‌تواند بستهٔ درخواست ارتباط را رها کرده و یک اتصال بین دو سیستم انتهائی برقرار می‌شود (قدم چهارم). تمام داده‌های کاربر که بین دو سیستم انتهائی رد و بدل می‌شوند بتوسط SSMهای ذیربط و با استفاده از کلید اجلاس یکبار مصرف رمزنگاری می‌شوند.

روش توزیع اتوماتیک کلید، انعطاف‌پذیری و پویائی لازم برای ارتباط تعدادی پایانه با تعدادی میزبان و همچنین ارتباط میزبان‌ها برای مبادلهٔ داده‌ها با یکدیگر را فراهم می‌آورد.

روش دیگری برای توزیع کلید، استفاده از رمزنگاری کلید- عمومی است که در فصل سوم مورد بحث قرار گرفته است.



شکل ۱۲-۲ توزیع اتوماتیک کلید برای پروتکل با گرایش اتصالی

۲-۷ منابع مطالعاتی

عناوین این فصل با جزئیات بیشتری در [STAL06a] پوشش داده شده است. در زمینهٔ الگوریتم‌های رمزنگاری، [SCHN96] یک مرجع کامل است که تقریباً تمام الگوریتم‌ها و پروتکل‌های رمزنگاری که تا زمان نشر این کتاب منتشر شده‌اند در آن توصیف شده است. یکی دیگر از بررسی‌های دقیق و ارزشمند [MENE97] است. یک نگرش عمیق‌تر بهمراه بحث‌های مفصل ریاضی در [STIN06] آمده است.

MENE97 Menezes, A.; van Oorshoot, P.; and Vanstone, S. *Handbook of Applied Cryptography.* Boca Raton, FL: CRC Press, 1997.
SCHN96 Schneier, B. *Applied Cryptography*. New York: Wiley, 1996.
STAL06a Stallings, W. *Cryptography and Network Security: Principles and Practice, Fourth Edition.* Upper Saddle River, NJ: Prentice Hall, 2006.
STIN06 Stinson, D. *Cryptography: Theory and Practice***.** Boca Raton, FL: CRC Press**,** 2006.

وِب سایت‌های مفید

- **AES home page**: صفحهٔ NIST در مورد AES. شامل استاندارد و یک سری اسناد مرتبط دیگر است.
- **AES Lounge**: شامل یک فهرست مفصل از اسناد و مقاله‌ها در مورد AES با قابلیت کپی برداشتن الکترونیک از آنهاست.
- **Block Cipher Modes of Operation**: صفحهٔ NIST با اطلاعات کاملی در مورد مُودهای مورد تأئید NIST.

۲-۸ واژه‌های کلیدی، سؤالات مرورکنندهٔ بحث و مسائل

واژه‌های کلیدی

رمزنگاری	encryption	استاندارد رمزنگاری پیشرفته	Advanced Encyption Standard (AES)
رمزنگاری سر- به- سر	end-to-end encryption	رمز قالبی	block cipher
رمز Feistel	Feistel cipher	حملهٔ همه جانبه	brute-force attack
توزیع کلید	key distribution	مُود زنجیره‌ای رمز قالبی	cipher block chaining (CBC) mode
رمزنگاری پیوند	link encryption	مود فیدبک رمز	cipher feedback (CFB) mode
متن ساده	plaintext	متن رمزشده	ciphertext
کلید اجلاس	session key	شکستن رمز-کشف رمز	cryptoanalysis
رمز دنباله‌ای	stream cipher	رمزنگاری	cryptography
زیرکلید	subkey	استاندارد رمزنگاری دیتا	Data Encryption Standard (DES)
رمزنگاری متقارن	symmetric encryption	رمزگشائی	decryption
DES سه‌گانه	triple DES (3DES)	مُود کتاب کُد الکترونیک	electronic codebook (ECB) mode

سوالات مرورکننده بحث

۲-۱ اجزاء ضروری یک رمز متقارن کدامند؟

۲-۲ دو عمل اساسی که در الگوریتم‌های رمزنگاری از آنها استفاده می‌شود، کدامند؟

۲-۳ برای اینکه دو نفر از طریق یک رمز متقارن با هم ارتباط یابند، چند کلید لازم است؟

۲-۴ اختلاف بین یک رمز قالبی با یک رمز دنباله‌ای چیست؟

۲-۵ دو روش معمول حمله به رمز کدامند؟

۲-۶ چرا بعضی از مُودهای عملیاتی رمز قالبی تنها از رمزنگاری استفاده کرده در حالی که برخی دیگر هم از رمزنگاری و هم از رمزگشائی استفاده می‌کنند؟

۲-۷ رمزنگاری سه‌گانه چیست؟

۲-۸ چرا بخش میانی 3DES رمزگشائی است و رمزنگاری نیست؟

۲-۹ اختلاف بین رمزنگاری پیوند و رمزنگاری سر- به- سر در چیست؟

۲-۱۰ راه‌های ممکن برای توزیع یک کلید سرّی بین دو واحد مرتبط را نام ببرید.

۲-۱۱ اختلاف یک کلید اصلی با یک کلید اجلاس در چیست؟

۲-۱۲ یک مرکز توزیع کلید چیست؟

مسائل

۲-۱ نشان دهید که رمزگشائی Feistel، عکس رمزنگاری Feistel است.

۲-۲ کدام اندازهٔ کلید RC4، بردار حالت S در مرحلهٔ آغازیدن را تغییر نخواهد داد؟ یعنی بعد از جایگشت اولیهٔ S، مؤلفه‌های S برابر مقادیر صفر تا ۲۵۵ بصورت صعودی خواهند بود.

۲-۳ RC4 دارای یک حالت داخلی سرّی است که جایگشت تمام مقادیر ممکن بردار S و دو اندیس i و j است.

الف- با استفاده از یک روش سرراست برای ذخیره نمودن حالت داخلی، چه تعداد بیت مورد استفاده قرار می‌گیرد؟

ب - فرض کنید که به این مسأله با این دید نگاه کنیم که چه مقدار اطلاعات با این حالت مرتبط است. در این صورت لازم است تعیین کنیم که چند حالت مختلف وجود دارد و آنگاه لگاریتم این عدد در مبنای ۲ را حساب کرده تا دریابیم که چند بیت اطلاعات نشان‌دهندهٔ این حالت است. با این روش چند بیت لازم است تا حالت را نشان دهد.

۲-۴ در مُود ECB، اگر خطائی در یک بلوک متن رمزشدهٔ ارسال شده بوجود آید، تنها بلوک متن سادهٔ نظیرآن تحت تأثیر واقع می‌شود. در حالی‌که در مُود CBC، این خطا منتشر می‌شود. بعنوان مثال، یک خطا در C_1 ارسال شده (شکل ۹-۲) بطور آشکار P_1 و P_2 را خراب میکند.

الف - آیا بلوک دیگری بجز P_2 تحت تأثیر خطا قرار می‌گیرد؟

ب – فرض کنید که یک خطا در نسخهٔ ابتدائی P_1 وجود دارد. این خطا در چند بلوک متن رمزشده منتشر می‌شود ؟ اثر این امر در گیرنده چه خواهد بود؟

۲-۵ CBC-Pad یک مِود عملیاتی رمز قالبی است که در رمز قالبی RC5 بکار می‌رود ولی می‌تواند در هر رمز قالبی دیگر نیز از آن استفاده شود. CBC-Pad می‌تواند به هر متن ساده با هر طولی اعمال گردد. طول متن رمز شده نظیر، حداکثر به اندازهٔ یک بلوک از طول متن ساده بیشتر خواهد بود. برای اینکه طول متن ساده مضربی از طول بلوک گردد، بیت‌های لائی به متن اضافه می‌گردد. فرض می‌شود که متن سادهٔ اولیه مضرب صحیحی از بایت‌هاست. به انتهای این متن ساده از 1 تا bb بایت اضافه می‌شود که bb معادل اندازهٔ بلوک برحسب بایت است. بایت‌های لائی همه یکسان بوده و این مقدار برابر تعداد بایت‌ها لائی است. بعنوان مثال اگر ۸ بایت لائی وجود داشته باشد، هر بایت دارای اندازهٔ 00001000 است. چرا از عدم استفاده از لائی اجتناب می‌شود؟ یعنی اگر طول متن سادهٔ اولیه مضرب صحیحی از اندازهٔ بلوک باشد، چرا بازهم از لائی استفاده می‌شود؟

۲-۶ استفاده از بیت‌های لائی همیشه مناسب نیست. مثلاً ممکن است علاقه‌مند باشیم تا دیتای رمزشده را در همان حافظهٔ موقت که دیتای متن ساده را نگهداری می‌نماید ذخیره کنیم. در این حالت طول دیتای رمزشده بایستی با طول دیتای اولیه برابر باشد. مُود عملیاتی مخصوص این کار مُود CTS (Ciphertext Stealing Mode) نام دارد. شکل ۱۳-۲ الف پیاده‌سازی این مُود را نشان می‌دهد.

الف- طرز عمل این مِود را تشریح کنید.

ب- توضیح دهید که C_{n-1} و C_n چگونه رمزگشائی می‌شوند.

۲-۷ شکل ۱۳-۲ب روش دیگری برای تولید یک متن رمزشده با طولی مساوی متن ساده، در حالتی که متن ساده مضرب صحیحی از طول بلوک نیست، را نشان می‌دهد.

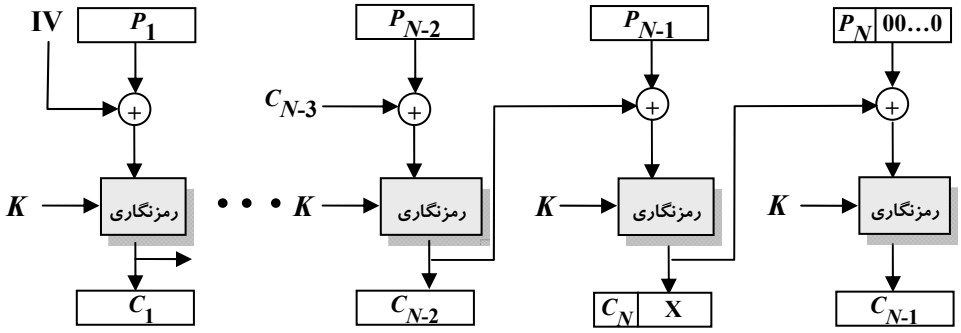
الف- الگوریتم را تشریح کنید.

ب- توضیح دهید که چرا CTS به روشی که در شکل ۱۳-۲ب نشان داده شده است، ارجح است.

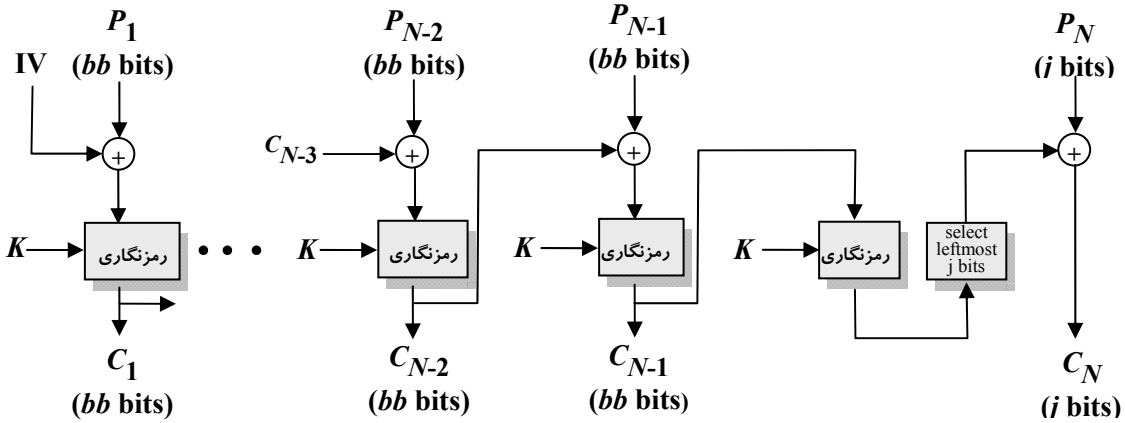
۲-۸ اگر یک بیت خطا در انتقال یک کاراکتر ۸- بیتی رمزشده در مُود CBF اتفاق افتد، این خطا تا چه مسافتی منتشر می‌شود؟

۲-۹ روش‌های توزیع کلید که با استفاده از یک مرکز کنترل دست‌یابی و/ یا مرکز توزیع کلید انجام می‌شوند دارای نقاط آسیب‌پذیر مرکزی‌اند. در مورد امنیت ضمنی چنین تمرکزی بحث نمائید.

۲-۱۰ فرض کنید که فردی راه زیر را برای تأئید اینکه هر دوی شما صاحب یک کلید واحد سرّی می‌باشید پیشنهاد می‌کند. یک دنبالهٔ تصادفی از بیت‌ها با همان طول کلید تولید کرده، آن را با کلید XOR کرده و نتیجه را روی کانال بفرستید. شریک شما، بلوک ورودی را با کلید XOR نموده و آن را پس می‌فرستد. شما بیت‌های دریافتی را کنترل کرده و اگر آنها برابر دنبالهٔ تصادفی اولیه شما بودند تأئید می‌کنید که کلید شریک شما همان کلید شماست و با این روش هیچیک از شما خود کلید را انتقال نداده‌اید. آیا در این روش عیبی وجود دارد؟



(الف) مُود CTS



(ب) روش دیگر

شکل ۱۳-۲ مُودهای رمز قالبی برای متون ساده‌ای که مضرب صحیحی از طول بلوک نیستند

فصل ۳

رمزنگاری کلید- عمومی و اعتبارسنجی پیام

لاوه بر محرمانگی پیام، اعتبارسنجی پیام(message authentication) نیز یک وظیفهٔ مهم امنیتی است. این فصل سه جنبه از اعتبارسنجی پیام را مورد بحث قرار می‌دهد. ابتدا به استفاده از کُدهای اعتبارسنجی پیام و توابع درهم‌ساز (hash functions) برای فراهم آوردن اعتبارسنجی پیام نگاه می‌کنیم. سپس به اصول رمزنگاری کلید- عمومی و دو الگوریتم مخصوص آن نظرمی‌افکنیم. این الگوریتم‌ها در مبادلهٔ کلیدهای رمزنگاری سنّتی مفید هستند. پس از آن به استفاده از رمزنگاری کلید- عمومی برای تولید امضاء دیجیتال توجه می‌کنیم که نوع ارتقاء یافتهٔ اعتبارسنجی پیام است. بالاخره مجدداً نگاهی به مقولهٔ مدیریت کلید می‌اندازیم.

۳-۱ نحوهٔ برخورد با اعتبارسنجی پیام

رمزنگاری، در برابر حملات امنیتی غیرفعال (استراق سمع) حفاظت ایجاد می‌کند. نیاز متفاوت دیگر این است که در برابر حملات فعال (جعل اسناد و داده‌ها)، ایجاد حفاظت نمائیم. حفاظت در مقابل چنین حملهٔهائی را اعتبارسنجی پیام گویند.

یک پیام، فایل، سند، و یا مجموعهٔ دیگری از داده‌ها را وقتی معتبر خوانند که دست اول بوده و از یک منبع قانونی منشأ گرفته باشد. اعتبارسنجی پیام روشی است که به طرفین درگیر در ارتباط اجازه می‌دهد تا معتبر بودن پیام را تأئید نمایند. دو جنبهٔ مهم امر، یکی تحقیق در مورد دست نخورده بودن پیام و دومی معتبر بودن خود منبع است. همچنین ممکن است علاقه‌مند باشیم تا بهنگام بودن پیام (اینکه عمداً به تأخیر نیفتاده و بازخوانی نشده باشد) و یا نظم آن نسبت به سایر پیام‌هائی که بین دوطرف ردوبدل می‌شود را تحقیق کنیم.

اعتبارسنجی با استفاده از رمزنگاری متقارن

اعتبارسنجی را میتوان بسادگی با استفاده از رمزنگاری متقارن انجام داد. اگر فرض کنیم که تنها فرستنده و گیرنده یک کلید مشترک در اختیار دارند (که بایستی چنین باشد)، آنگاه تنها فرستندهٔ واقعی است که قادر به رمزنگاری موفقیت‌آمیز پیام برای طرف مقابل است. علاوه برآن اگر پیام شامل یک کُد تشخیص خطا و یک شمارهٔ ردیف باشد، گیرنده مطمئن خواهد بود که تغییری در پیام رخ نداده و نظم پیام نیز صحیح می‌باشد. همچنین اگر پیام دارای یک برچسب زمانی باشد، گیرنده مطمئن خواهد شد که پیام بیش از حد معقولی که معمولاً شبکه در آن تأخیر ایجاد میکند دیر دریافت نشده است.

اعتبارسنجی پیام بدون رمزنگاری پیام

در این بخش، چند روش اعتبارسنجی پیام که متکی به رمزنگاری نیستند را بررسی می‌کنیم. در تمام این روش‌ها برای انتقال، یک دنبالهٔ (tag) اعتبارسنجی به پیام وصل میشود. خود پیام به رمز درنیامده و می‌تواند مستقل از عمل اعتبارسنجی در مقصد خوانده شود.

نظر به اینکه روش‌های مورد بحث در این بخش پیام را به رمز درنمی‌آورند، محرمانگی پیام فراهم نمی‌گردد. با توجه به اینکه رمزنگاری متقارن اعتبارسنجی را فراهم می‌آورد و با توجه به اینکه با حضور محصولات آمادهٔ موجود، رمزنگاری متقارن بطور گسترده‌ای مورد استفاده قرار می‌گیرد، چرا به سهولت از چنین روشی که هم محرمانگی و هم اعتبارسنجی را ایجاد میکند استفاده نمی‌کنیم؟ [DAVI89] در سه حالت، اعتبارسنجی به دور از محرمانگی را ارجح می‌شمارد:

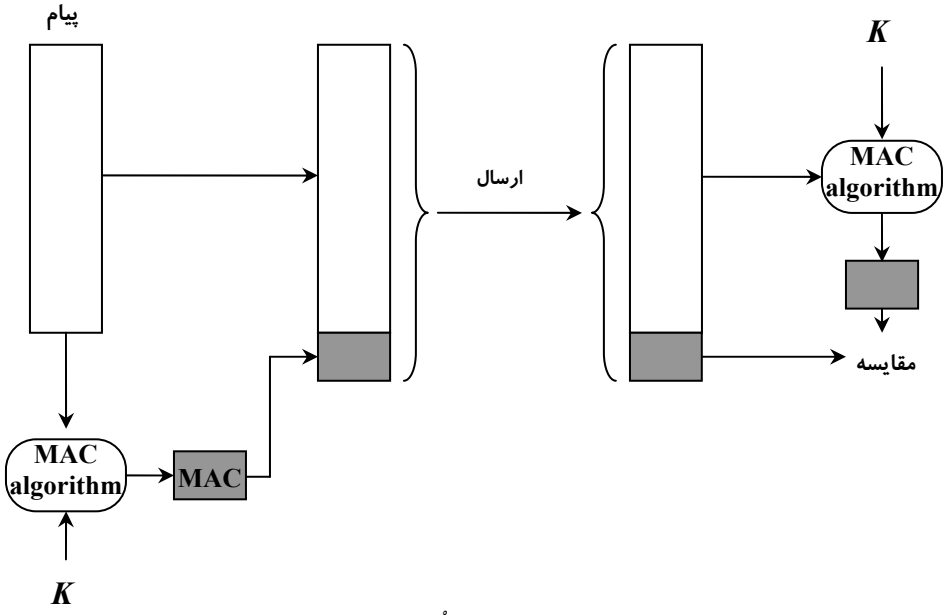
۱- کاربردهائی وجود دارند که در آنها یک پیام به مقاصد متعددی ارسال می‌شود. مثلاً اخطار به کاربران شبکه در مورد اینکه شبکه فعلاً قطع است، و یا آلارم یک مرکز کنترل از این جمله‌اند. در این حالت داشتن تنها یک مقصد مسئول اعتبارسنجی ارزان‌تر و قابل اعتمادتر است. بنابراین پیام بایستی بصورت یک متن ساده بهمراه یک دنبالهٔ اعتبارسنجی پیام پخش شود. اگر نتیجهٔ اعتبارسنجی منفی باشد سیستم مسئول اعتبارسنجی، سیستم‌های دیگر مقصد را بتوسط یک هشدار عمومی خبردار خواهد کرد.
۲- سناریوی ممکن دیگر، تبادل داده‌ها بین دوطرفی است که یکی از آنها بار سنگینی داشته و وقت لازم برای رمزگشائی همهٔ پیام‌های ورودی را ندارد. اعتبارسنجی بر اساس انتخاب نمونه صورت پذیرفته و پیام‌ها بطور تصادفی برای کنترل اعتبار برگزیده می‌شوند.
۳- اعتبارسنجی یک برنامهٔ کامپیوتری با متن ساده، سرویس پرجاذبه‌ای است. برنامهٔ کامپیوتر میتواند بدون اینکه هر بار رمزگشائی شود، اجرا شود که خود صرفه‌جوئی بزرگی در منابع پردازش‌گر است. در عین حال اگر یک دنبالهٔ اعتبارسنجی پیام به برنامه منتقل گردد، میتوان آن را در هر زمان لازم برای اطمینان از اصالت پیام کنترل نمود.

بنابراین برای برآورده نمودن نیازهای امنیتی، جائی برای استفادهٔ توأم از اعتبارسنجی و رمزنگاری وجود دارد.

کُد اعتبارسنجی پیام (MAC)

یکی از روش‌های اعتبارسنجی شامل استفاده از یک کلید سرّی برای تولید یک بلوک کوچک دیتا، بنام کُد اعتبارسنجی پیام (Message Authentication Code) است که به پیام وصل می‌شود. در این روش فرض براین است که دو طرف ارتباط، مثلاً A و B، یک کلید سرّی مشترک مثل K_{AB} را در اختیار دارند. وقتی A دارای پیامی به مقصد B است، کُد اعتبارسنجی پیام را بصورت تابعی از پیام و کلید، محاسبه می‌کند: $\mathrm{MAC}_M = \mathrm{F}(K_{AB}, M)$. پیام باضافهٔ کُد برای گیرندهٔ مورد نظر ارسال می‌شود. گیرنده با استفاده از همان کلید، همان محاسبات را روی پیام ورودی انجام داده و یک کُد اعتبارسنجی جدید بدست می‌آورد. کُد دریافت شده با کُد محاسبه شده مقایسه می‌شود (شکل ۱- ۳). اگر فرض کنیم که تنها گیرنده و فرستنده از کلید سرّی باخبرند و اگر کُد دریافت شده با کُد محاسبه شده تطبیق داشته باشد، آنگاه

۱-گیرنده مطمئن می‌شود که پیام تغییر نیافته است. اگر دشمنی پیام را تغییر داده ولی کُد را تغییر نداده باشد، آنگاه کُد محاسبه شدهٔ گیرنده با کُد دریافت شده فرق خواهد داشت. چون فرض شده است که دشمن از کلید سرّی بی‌خبر است، او نمی‌تواند کُد را طوری تغییر دهد که با تغییرات پیام همخوانی داشته باشد.
۲-گیرنده مطمئن می‌شود که پیام از سوی فرستندهٔ قانونی ارسال شده است. چون کس دیگری از کلید سرّی مطلع نیست، هیچ‌کس نمی‌تواند یک پیام جعلی با کُد صحیح را تهیه نماید.
۳- اگر پیام دارای یک شمارهٔ ردیف باشد (مثل آنچه در X.25 ، HDLC و TCP مورد استفاده است)، آنگاه گیرنده می‌تواند از نظم پیام مطمئن شود زیرا یک دشمن نمی‌تواند شمارهٔ ردیف را بصورت موفقیت‌آمیزی تغییر دهد.



شکل ۱-۳ اعتبارسنجی پیام با استفاده از کُد اعتبارسنجی پیام (MAC)

برای تولید کُد اعتبارسنجی، می‌توان از الگوریتم‌های متعددی استفاده کرد. NIST در مشخصهٔ FIPS PUB 113، استفاده از DES را توصیه می‌کند. DES برای تولید یک نسخهٔ رمزنگاری شده از پیام بکاررفته، و آخرین بیت‌های متن رمزشده بعنوان کُد مورد استفاده قرار می‌گیرد. یک کُد ۱۶ یا ۳۲- بیتی، کُدی مرسوم است.

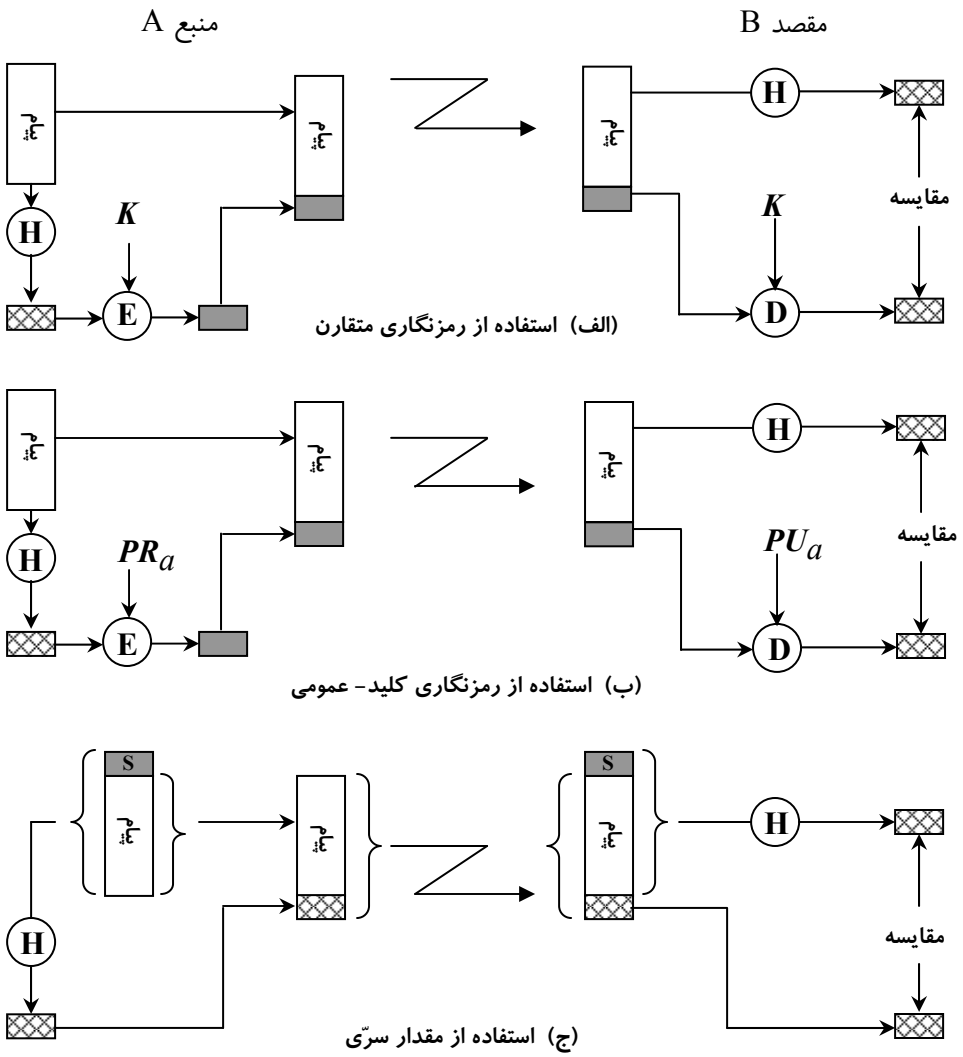
روشی که هم اکنون ذکر شد، شبیه رمزنگاری است. یک اختلاف آن این است که برخلاف آنچه در رمزگشائی لازم است، الگوریتم اعتبارسنجی لازم نیست برگشت‌پذیر باشد. چنین برمی‌آید که بدلیل خواص ریاضی تابع اعتبارسنجی، این تابع در مقابل شکستن رمز آسیب‌پذیری کمتری نسبت به رمزنگاری دارد.

تابع درهم‌ساز یک- طرفه

رقیب دیگری برای کُد اعتبارسنجی پیام، تابع درهم‌ساز یک- طرفه است. همانندکُد اعتبارسنجی پیام، یک تابع درهم‌ساز یک پیام M با طول متغیر بعنوان ورودی را گرفته و یک چکیدهٔ (digest) پیام $H(M)$ با طول ثابت را خارج می‌کند. برخلاف MAC، یک تابع درهم‌ساز یک کلید سرّی را بعنوان ورودی نمی‌خواهد. برای اعتبارسنجی پیام، چکیدهٔ پیام بصورتی به همراه پیام ارسال می‌شود که چکیده معتبر بماند.

شکل ۲-۳ سه روش که پیام می‌تواند بتوسط آنها اعتبارسنجی شود را نشان می‌دهد. چکیدهٔ پیام می‌تواند با استفاده از رمزنگاری متقارن (بخش الف) به رمز درآید. اگر فرض شود که تنها فرستنده و گیرنده کلید سرّی را در اشتراک دارند، آنگاه اعتبارسنجی محرز است. پیام همچنین می‌تواند با استفاده از رمزنگاری کلید- عمومی به رمز درآید (بخش ب) که این مقوله در بخش ۵-۳ توضیح داده شده است. روش کلید- عمومی دارای دو مزیت است: اول اینکه علاوه بر اعتبارسنجی پیام، یک امضاء دیجیتال نیز تولید می‌شود و دوم اینکه نیازی به توزیع کلید بین طرفین ارتباط نیست.

مزیت این دو روش نسبت به روش‌هائی که تمام پیام را رمزنگاری می‌کنند این است که محاسبات کمتری مورد نیاز است. با وجود این علاقهٔ زیادی نسبت به خلق تکنیک‌هائی وجود دارد که بطور کلی از رمزنگاری صرفنظر شود. دلایل متعددی برای این علاقه در [TSUD92] ذکر شده است:



شکل ۲-۳ اعتبارسنجی پیام با استفاده از یک تابع درهم‌ساز یک- طرفه

- نرم‌افزار رمزنگاری تا حد زیادی کند است. اگرچه میزان داده‌هائی که برای هر پیام باید رمزنگاری شود کم است، ولی ممکن است یک جریان پیوسته از پیام‌ها وارد سیستم شده و از آن خارج گردد.
- هزینه‌های سخت‌افزاری رمزنگاری قابل چشم‌پوشی نیست. DES روی تراشه‌های ارزان قیمتی پیاده‌سازی شده است ولی اگر قرار باشد که همهٔ گره‌های یک شبکه قابلیت رمزنگاری را داشته باشند، هزینه بالا خواهد رفت.
- سخت‌افزار رمزنگاری وقتی بهینه‌تر می‌شود که اندازهٔ بلوک‌های دیتا بزرگ‌تر شود. برای بلوک‌های کوچک دیتا، بخش قابل توجهی از زمان برای سربارهای شروع/ فراخوانی صرف خواهد شد.
- یک الگوریتم رمزنگاری ممکن است به ثبت رسیده و استفاده از آن بدون اجازه امکان‌ناپذیر باشد

شکل ۲-۳ج تکنیکی را نشان می‌دهد که از یک تابع درهم‌ساز استفاده شده ولی از رمزنگاری بمنظور اعتبارسنجی پیام استفاده نمی‌کند. این روش فرض می‌کند که دوطرف ارتباط، مثل A و B، یک مقدار سری S_{AB} را در اشتراک دارند. وقتی A

می‌خواهد پیامی را برای B بفرستد، او تابع درهم‌ساز را روی اتصال پیام و مقدار سرّی اعمال می‌کند ($MD_M = \mathrm{H}(S_{AB} \| M)$ || علامت این است که مقادیر دو سمت این علامت پشت سرهم قرار می‌گیرند). A آنگاه $[M \| MD_M]$ را برای B می‌فرستد. چون B، S_{AB} را در اختیار دارد، او می‌تواند $\mathrm{H}(S_{AB} \| M)$ را مجدداً محاسبه کرده و MD_M را تأئید کند. چون خود مقدار سری S_{AB} ارسال نمی‌شود، برای یک دشمن میسر نخواهد بود که پیام را دستکاری نماید. همچنین تا زمانی که مقدار سرّی محفوظ باشد، برای دشمن میسر نخواهد بود که یک پیام تقلبی ایجاد کند.

نوع تعدیل شده‌ای از تکنیک سوم، بنام HMAC برای امنیت IP انتخاب شده است (فصل ششم). این روش همچنین برای SNMPv3 (فصل هشتم) نیز تعیین گردیده است.

۳-۲ توابع درهم‌ساز امن و HMAC

تابع درهم‌ساز امن یک- طرفه یا تابع hash امن، نه تنها در اعتبارسنجی پیام بلکه در امضاءهای دیجیتال نیز حائز اهمیت است. این بخش را با بررسی خواص مورد نیاز یک تابع درهم‌ساز امن شروع می‌کنیم. سپس یکی از مهم‌ترین توابع درهم‌ساز یعنی SHA را مورد بررسی قرار می‌دهیم. در پایان HAMC را معرفی خواهیم کرد.

لازمه‌های تابع درهم‌ساز

هدف یک تابع درهم‌ساز این است که یک «اثرانگشت» از یک فایل، یک پیام، و یا بلوک دیگری از دیتا بوجود آورد. برای این که یک تابع درهم‌ساز H به درد اعتبارسنجی پیام بخورد بایستی دارای خواص زیر باشد:

۱- H بتواند به یک بلوک داده با هر اندازه‌ای اعمال گردد.
۲- H یک خروجی با طول ثابت ایجاد کند.
۳- محاسبهٔ $\mathrm{H}(x)$ برای هر x نسبتاً ساده بوده و بتوان آن را بصورت سخت‌افزاری و نرم‌افزاری اجرا نمود.
۴- برای هر مقدار h، پیداکردن x بطوری که $\mathrm{H}(x) = h$ باشد از نظر محاسباتی مقدور نباشد.
۵- برای هر بلوک x، پیداکردن یک مقدار $y \neq x$ بطوری‌که $\mathrm{H}(y) = \mathrm{H}(x)$ باشد، از نظر محاسباتی مقدور نباشد. این مورد را گاهی **مقاومت ضعیف در برابر تصادم (weak collision resistance)** نامند.
۶- پیدا کردن زوجی مانند (x,y) بطوری که $\mathrm{H}(x) = \mathrm{H}(y)$ باشد از نظر محاسباتی مقدور نباشد.این مورد را گاهی **مقاومت قوی در برابر تصادم (strong collision resistance)** نامند.

سه خاصیت اول مربوط به نیازهای کاربردهای عملی یک تابع درهم‌ساز برای اعتبارسنجی پیام است. خاصیت چهارم یک خاصیت «یک- طرفه» است: این ساده است تا کُد مربوط به یک پیام را تولید کرد ولی عملاً محال است که با داشتن کُد، پیام را بدست آورد. این خاصیت در صورتی که تکنیک اعتبارسنجی شامل استفاده از یک مقدار سرّی باشد (شکل۲-۳ج)، مهم است. اگرچه خود مقدار سرّی ارسال نمی‌شود ولی اگر تابع درهم‌ساز یک- طرفه نباشد، یک دشمن بسهولت می‌تواند مقدار سرّی را کشف کند: اگر دشمن بتواند به یک انتقال گوش فراداده و یا آن را مشاهده نماید، او پیام M و کُد درهم شدهٔ $MD_M = \mathrm{H}(S_{AB} \| M)$ را به چنگ می‌آورد. دشمن آنگاه تابع درهم‌ساز را معکوس کرده تا $S_{AB} \| M = \mathrm{H}^{-1}(MD_M)$ را بدست آورد. حال چون دشمن هم M و هم $S_{AB} \| M$ را در اختیار دارد، بسادگی می‌تواند S_{AB} را استخراج نماید.

خاصیت پنجم تضمین می‌کند که امکان‌پذیر نیست تا پیام دیگری با همان اندازهٔ hash پیام اصلی بدست آورد. این امر از تقلب در زمانی که یک کُد hash رمزشده بکار می‌رود جلوگیری می‌کند (شکل‌های ۲-۳الف و ب). اگر این خاصیت جاری نباشد، یک حمله‌کننده قادر به انجام عملیات زیر خواهد بود: اول، یک پیام باضافهٔ کُد hash رمزنگاری شدهٔ آن را ملاحظه یا استراق سمع کند. دوم، یک کُد hash رمزنشده از پیام را تهیه کند. سوم، یک پیام دیگر با همان کُد hash تهیه نماید.

یک تابع hash که پنج خاصیت اول لیست قبل را داشته باشد، تابع hash ضعیف می‌خوانند. اگر علاوه برآن خاصیت ششم نیز ارضاء گردد، آنگاه تابع hash را قوی می‌نامند. خاصیت ششم از یک دسته حملات پیچیده که حملهٔ روز تولد خوانده می‌شوند، جلوگیری می‌نماید. جزئیات این حمله فراتر از افق دید این کتاب است. این حمله توانائی یک تابع درهم‌ساز m- بیتی را از 2^m به $2^{m/2}$ کاهش می‌دهد. [YUVA79] و یا [STAL06a] را ملاحظه نمائید.

علاوه بر فراهم نمودن اعتبارسنجی، یک چکیدهٔ پیام صحت پیام را نیز تضمین می‌کند. عمل آن همانند دنبالهٔ کنترل فریم (FCS) است: اگر هر بیتی از پیام در حین انتقال بطور تصادفی تغییر نماید، چکیدهٔ پیام عوض خواهد شد.

توابع درهم‌ساز ساده

تمام توابع درهم‌ساز بر اساس اصول عمومی زیر کار می‌کنند. ورودی (پیام، فایل، و غیره) بصورت دنباله‌ای از بلوک‌های n- بیتی تلقی می‌گردد. ورودی بصورت یک بلوک در هر زمان و به روش تکرار مورد پردازش قرار گرفته تا یک تابع درهم‌سازی شدهٔ n- بیتی را تولید کند.

یکی از ساده‌ترین توابع درهم‌سازی، XOR کردن بیت- به- بیت هر بلوک است. این را می‌توان بصورت زیر نشان داد:

$$C_i = b_{i1} \oplus b_{i2} \oplus \ldots \oplus b_{im}$$

که در آن

C_i = امین بیت کُد hash $1 \le i \le n$

m = تعداد بلوک‌های n- بیتی در ورودی

b_{ij} = امین بیت در jامین بلوک

$\oplus$ = عمل XOR

شکل ۳-۳ این عملیات را نشان می‌دهد. هر بیت کُد hash بسادگی یک بیت توازن (parity) بوده که تست افزونگی عمودی (VRC) نیز خوانده می‌شود. این عمل برای داده‌های تصادفی بعنوان یک تست صحت دیتا کاملاً مؤثر است. هریک از مقادیر n- بیتی کُد درهم‌سازی شده بطور مساوی محتمل‌اند. بنابراین احتمال اینکه یک خطا در دیتا باعث تغییرنکردن کُد hash آن شود، 2^{-n} است. اگر فرمت دیتا قابل پیش‌بینی‌تر باشد، تابع کمتر مؤثر خواهد بود. مثلاً در بیشتر فایل‌های متنی نرمال، بیت پرارزش هراُکتت همیشه صفر است. بنابراین اگر از یک مقدار ۱۲۸- بیتی hash استفاده شود، بجای اینکه تأثیر آن $۲^{-۱۲۸}$ باشد، تابع hash چنین دیتائی دارای تأثیری برابر $۲^{-۱۱۲}$ خواهد بود.

راه ساده‌ای برای بهبود کار این است که پس از پردازش هربلوک، یک گردش و یا شیفت حلقوی باندازهٔ یک بیت روی مقدار hash انجام دهیم. روش را می‌توان بصورت زیر خلاصه کرد:

۱- در ابتدا مقدار n- بیتی hash را برابر صفر قرار دهید.

۲- هر بلوک n- بیتی دیتا را پس از آن متوالیاً به روش زیر پردازش نمائید:

الف- مقدار جاری hash را یک بیت به سمت چپ بچرخانید.

ب- بلوک را با مقدار hash، XOR نمائید.

اثر این امر «تصادفی»تر کردن و غلبه بر هر نوع نظمی است که در ورودی وجود دارد.

اگرچه روش دوم معیار خوبی برای سنجش صحت دیتا را فراهم می‌آورد، ولی وقتی همانند شکل‌های ۲-۳الف و ب یک کُد hash رمزنگاری شده بهمراه یک پیام متن ساده بکار می‌رود، از نظر امنیت داده‌ها بی‌ارزش است. برای یک پیام داده شده، کار آسانی است تا پیام جدیدی که همان کُد hash را تولید می‌کند خلق کرد: بسادگی پیام مورد نظر جدید را تهیه کرده و آنگاه یک بلوک n- بیتی که باعث شود تا پیام جدید باضافهٔ بلوک، کُد hash مطلوب را ایجاد نماید به آن اضافه کنید.

اگرچه یک XOR ساده و یا یک XOR چرخش یافته(RXOR) در حالتی که فقط کُد hash رمزنگاری می‌شود کافی نیست، ولی ممکن است هنوز تصور کنید که در صورتی‌که هم پیام و هم کُد hash دنبال آن رمزنگاری گردند، چنین تابع ساده‌ای می‌تواند مفید واقع شود. اما بایستی دقت کرد. تکنیکی که در ابتدا بتوسط دایرهٔ ملی استانداردهای آمریکا پیشنهاد گردید از یک XOR ساده که به بلوک‌های ۶۴- بیتی پیام اعمال می‌شد و سپس از مُود زنجیره‌ای رمز قالبی (CBC) استفاده می‌کرد، تشکیل شده بود. روش را می‌توان چنین توصیف کرد: با داشتن پیامی که شامل دنباله‌ای از بلوک‌های ۶۴- بیتی $X_1, X_2, \ldots, X_n$ است، کُد hash آن را بصورت XOR بلوک- به- بلوک تمام بلوک‌ها تعریف کرده و کُد hash را بعنوان آخرین بلوک به پیام وصل کنید:

$$C = X_{N+1} = X_1 \oplus X_2 \oplus \ldots \oplus X_N$$

سپس تمام پیام بعلاوهٔ کُد hash را با استفاده از مُود CBC رمزنگاری کرده تا پیام رمزنگاری شده $Y_1, Y_2, \ldots, Y_{N+1}$ بدست آید. [JUEN85] به چندین راه که در آنها متن رمزشدهٔ این پیام می‌تواند طوری دستکاری شده که بتوسط کُد hash قابل آشکارسازی نباشد، اشاره می‌کند. بعنوان مثال، برحسب تعریف CBC (شکل ۹-۲) داریم

$$X_1 = IV \oplus \mathrm{D}(K, Y_1)$$
$$X_i = Y_{i-1} \oplus \mathrm{D}(K, Y_i)$$
$$X_{N+1} = Y_N \oplus \mathrm{D}(K, Y_{N+1})$$

	بیت 1	بیت 2	• • •	بیت n
بلوک 1	b_{11}	b_{21}		b_{n1}
بلوک 2	b_{12}	b_{22}		b_{n2}
	•	•	•	•
	•	•	•	•
	•	•	•	•
بلوک *m*	b_{1m}	b_{2m}		b_{nm}
کُد Hash	C_1	C_2		C_n

شکل ۳-۳ تابع hash ساده با استفاده از XOR بیت- به- بیت

اما X_{N+1} کُد hash است:

$$X_{N+1} = X_1 \oplus X_2 \oplus \ldots \oplus X_N$$
$$= [IV \oplus D(K, Y_1)] \oplus [Y_1 \oplus D(K, Y_2)] \oplus \ldots \oplus [Y_{N-1} \oplus D(K, Y_N)]$$

چون عبارات جملهٔ قبل را می‌توان با هر نظمی XOR نمود، نتیجه می‌گیریم که اگر بلوک‌های متن رمزشده تحت هر جایگشتی قرار گیرند، کُد hash تغییر نخواهد کرد.

تابع درهم‌ساز امن SHA-1

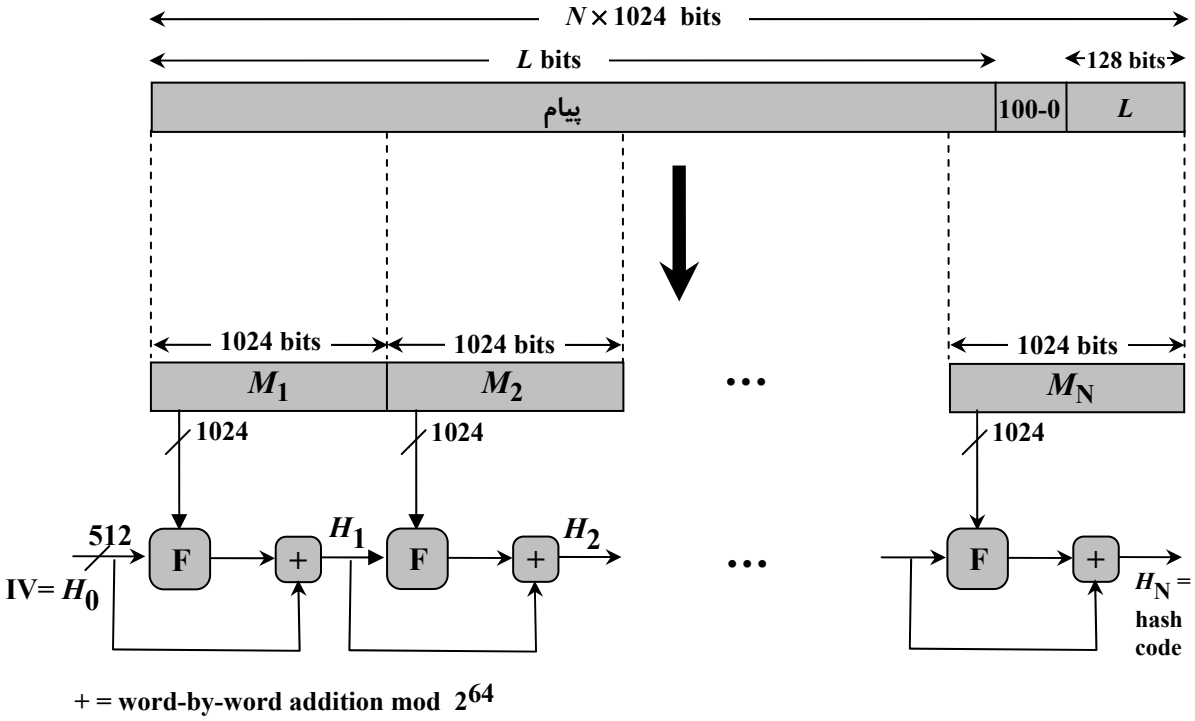
الگوریتم درهم‌ساز امن Secure Hash Algorithm (SHA) بتوسط انستیتوی ملی استانداردها و تکنولوژی آمریکا (NIST) تولید و بعنوان یک استاندارد فدرال پردازش اطلاعات (FIPS 180) در سال ۱۹۹۳ منتشر گردید. نسخهٔ بازنگری شدهٔ آن بعنوان FIPS 180-1 در سال ۱۹۹۵ منتشر شد و معمولاً از آن با نام SHA-1 یاد می‌شود. SHA-1 همچنین در RFC 3174 تعریف شده است که عموماً شامل همان تعاریف FIPS 180-1 بوده ولی یک کُد به زبان C نیز به آن اضافه شده است.

SHA-1 یک اندازهٔ hash با طول ۱۶۰ بیت ایجاد می‌کند. در سال ۲۰۰۲، NIST یک نسخهٔ جدید از استاندارد، FIPS 180-2 را تولید کرد که سه نسخهٔ جدید از SHA با طول‌های ۲۵۶، ۳۸۴ و ۵۱۲ بیت برای اندازهٔ hash را تعریف کرده است. این توابع SHA-256، SHA-384 و SHA-512 خوانده می‌شوند (جدول ۱-۳). این نسخه‌های جدید همان ساختار زیربنائی را داشته و از همان حساب پیمانه‌ای و عملیات منطقی باینری SHA-1 استفاده می‌کنند. در سال ۲۰۰۵، NIST قصد خود نسبت به کنارگذاشتن SHA-1 و اتکاء بر سایر نسخه‌های SHA تا سال ۲۰۱۰ را اعلام کرد. کمی بعد از آن، یک تیم تحقیقاتی حمله‌ای را توصیف کرد که درآن با استفاده از 2^{69} عملیات می‌توان دو پیام را پیدا کرد که دارای اندازهٔ hash یکسان باشند. این تعداد عملیات بسیار کمتر از 2^{80} عملیاتی است که تصور می‌شد برای پیداکردن یک تصادم در SHA-1 لازم است [WANG05]. این نتیجه نشان می‌دهد که گذر به سایر نسخ SHA بایستی سریع‌تر صورت پذیرد.

در این بخش توصیفی از SHA-512 را ارائه می‌دهیم. سایر نسخه‌ها کاملاً مشابه این نسخه‌اند. الگوریتم بعنوان ورودی یک پیام با ماکزیمم طول کمتر از 2^{128} را گرفته و یک چکیدهٔ پیام ۵۱۲- بیتی را تولید می‌کند. ورودی در بلوک‌های ۱۰۲۴- بیتی مورد پردازش قرار می‌گیرد. شکل ۴-۳ پردازش کامل یک پیام برای تولید یک چکیده را نشان می‌دهد. پردازش شامل قدم‌های زیر است:

جدول ۱-۳ مقایسهٔ پارامترهای SHA

	SHA-1	SHA-256	SHA-384	SHA-512
اندازهٔ چکیدهٔ پیام	160	256	384	512
اندازهٔ پیام	$<2^{64}$	$<2^{64}$	$<2^{128}$	$<2^{128}$
اندازهٔ بلوک	512	512	1024	1024
اندازهٔ کلمه	32	32	64	64
تعداد قدم‌ها	80	64	80	80
امنیت	80	128	192	256



شکل۴-۳ تولید چکیدهٔ پیام با استفاده SHA-1

- **قدم اول: بیت‌های لائی (padding) را به پیام وصل کنید**. بیت‌های لائی طوری به بیت‌های پیام اضافه می‌شوند که طول آن 1024 modulo 896 گردد. بیت‌های لائی همیشه به پیام اضافه می‌گردند حتی اگر پیام خود دارای طول مطلوب باشد. بنابراین تعداد بیت‌های لائی بین ۱ تا ۱,۰۲۴ خواهد بود. لائی شامل یک بیت 1 و بدنبال آن تعداد لازم بیت‌های 0 است.
- **قدم دوم: طول پیام را وصل کنید**. یک بلوک ۱۲۸- بیتی به انتهای پیام وصل می‌شود. این بلوک بصورت یک عدد صحیح ۱۲۸- بیتی بدون علامت (با ارزش‌ترین بایت در ابتدا واقع می‌شود) بوده و شامل طول پیام اولیه (قبل از اضافه شدن) است.

اجرای دو قدم اول، پیامی را بدست می‌دهد که طول آن مضرب صحیحی از ۱,۰۲۴ بیت است. در شکل ۴-۳ پیام توسعه‌یافته بصورت یک دنباله از بلوک‌های ۱,۰۲۴ بیتی $M_1, M_2, ..., M_N$ نشان داده شده بطوری‌که طول کلی پیام توسعه‌یافته $N \times 1024$ بیت است.

- **قدم سوم: حافظهٔ hash را با مقادیر اولیه پرکنید**. یک حافظهٔ موقت ۵۱۲- بیتی برای نگهداری مقادیر میانی و انتهائی تابع hash بکار می‌رود. این حافظهٔ موقت می‌تواند بصورت ۸ رجیستر ۶۴- بیتی (a, b, c, d, e, f, g, h) نشان داده شود. این حافظه‌های موقت در ابتدا با اعداد صحیح ۶۴- بیتی زیر (بصورت هکزادسیمال) پر می‌شوند:

a = 6A09E667F3BCC908
b = BB67AE8584CAA73B
c = 3C6EF372FE94F82B
d = A54FF53A5F1D36F1
e = 510E527FADE682D1
f = 9B05688C2B3E6C1F
g = 1F83D9ABFB41BD6B
h = 5BE0CDI9137E2179

این مقادیر با فرم big-endian ذخیره می‌شوند که در آن با اهمیت‌ترین بایت یک کلمه، در محل پائین‌ترین (چپ‌ترین) آدرس بایت قرارمی‌گیرد. این کلمات با انتخاب ۶۴ بیت اول بخش‌های اعشاری جذر اولین هشت عدد اول بدست آمده‌اند.

- **قدم چهارم: پیام را در بلوک‌های ۱۰۲۴- بیتی (۱۲۸-کلمه‌ای) پردازش کنید.** قلب الگوریتم مدولی است که شامل ۸۰ دُور پردازش است. این مدول درشکل ۴-۳ با F نشان داده شده است. منطق عمل درشکل ۵-۳ رسم شده است.

هر دُور بعنوان ورودی، یک بلوک ۵۱۲- بیتی abcdefgh حافظه موقت را گرفته و محتویات حافظهٔ موقت را به روز درمی‌آورد. بعنوان ورودی اولین دُور، حافظهٔ موقت دارای اندازهٔ hash میانی H_{i-1} است هر دور t از یک اندازهٔ ۶۴- بیتی W_t استفاده می‌کند که از بلوک ۱۰۲۴- بیتی در حال پردازش (M_i) مشتق شده است. هر دُور همچنین از یک ثابت جمع شونده K_t استفاده کرده که درآن $0 \le t \le 79$ نمایش دهندهٔ یکی از ۸۰ دُور است. این کلمات اولین ۶۴ بیت بخش‌های اعشاری ریشهٔ سوم اولین هشت عدد اول می‌باشند ثابت‌ها یک الگوی تصادفی ۶۴- بیتی را ایجاد می‌کنند که قاعدتاً هرگونه نظم در دیتای ورودی را از بین خواهد برد.

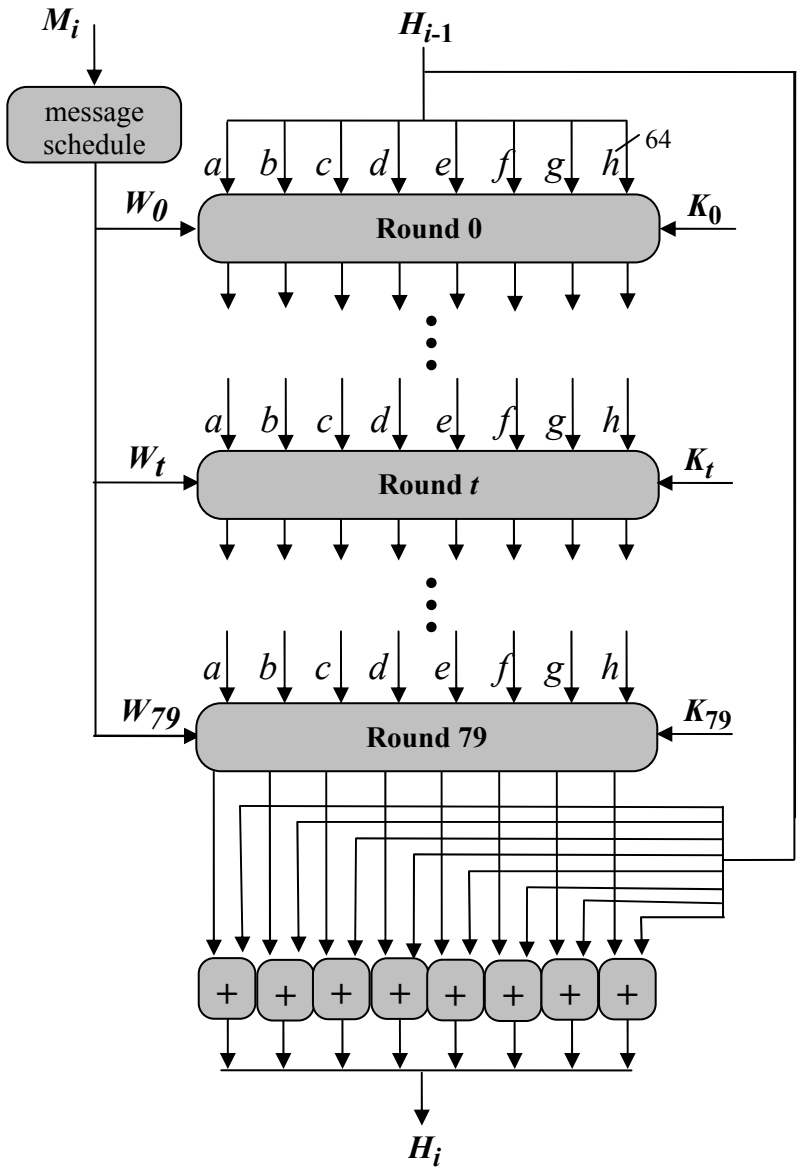
خروجی هشتادمین دُور به ورودی اولین دُور (H_{i-1}) اضافه شده تا H_i را تولید کند. جمع برای هریک از ۸ کلمهٔ موجود در حافظهٔ موقت با هر کلمهٔ نظیر در H_{i-1} جداگانه و بصورت modulo 2^{64} انجام می‌شود.

- **قدم پنجم: خروجی.** بعدازاینکه تمام N بلوک ۱۰۲۴- بیتی پردازش گردید، خروجی N امین مرحله چکیدهٔ ۵۱۲- بیتی پیام را تشکیل می‌دهد.

الگوریتم SHA-512 دارای این خاصیت است که هر بیت کُد hash تابعی از تمام بیت‌های ورودی است. تکرار پیچیدهٔ تابع اصلی F نتایجی را تولید می‌کند که بخوبی مخلوط شده‌اند. یعنی غیرمحتمل است که دو پیامی که بصورت تصادفی انتخاب شده‌اند، حتی اگر نظم مشابه‌ای از خود نشان دهند، دارای همان کُد hash یکسان باشند. مگر ضعف پنهان شده‌ای در SHA-512 وجود داشته باشد که تا کنون هویدا نشده باشد والا مسألهٔ رسیدن به دو پیامی که دارای چکیدهٔ یکسانی باشند نیاز به عملیاتی با حجم 2^{256} دارد. به همین ترتیب مورد پیداکردن پیامی با یک چکیدهٔ داده شده، نیاز به 2^{512} عملیات دارد.

سایر توابع درهم‌ساز امن

همانگونه که در مورد رمزهای قالبی متقارن نیز چنین بود، طراحان توابع درهم‌ساز امن تمایلی نداشته‌اند که از یک ساختار به اثبات رسیده عدول نمایند. DES بر مبنای رمز Feistel بناشده است. همهٔ رمزهای قالبی پس از آن نیز تلویحاً از طراحی Feistel تبعیت کرده‌اند، زیرا این طراحی را می‌توان طوری تغییر داد که در برابر تهدیدهای کشف رمزی که جدیداً کشف شده نیز مقاوم گردد. اگر بجای آن از یک طراحی کاملاً نو برای یک رمز قالبی متقارن استفاده می‌شد، این نگرانی وجود داشت که ساختار جدید راه‌های حملهٔ جدیدی که تا کنون نسبت به آن فکر نشده بود را بگشاید. بدلیل مشابه، بیشتر توابع درهم‌ساز جدید و مهم نیز از همان ساختار اساسی شکل ۴-۳ که تابع درهم‌ساز تکرارشونده خوانده شده و در ابتدا بتوسط Merkle[MERK79,MERK89] پیشنهاد شده است تبعیت می‌کنند. محرک این ساختار تکرارشونده از مشاهدات Merkle[MERK89] و Damgard[DAMG89] سرچشمه می‌گیرد که اگر تابع مربوط به یک بلوک منفرد، که **تابع فشرده‌سازی** خوانده می‌شود، در مقابل تصادم مقاوم باشد، نتیجهٔ تابع درهم‌شده و تکرارشدهٔ آن هم، همان خاصیت را خواهد داشت. بنابراین ساختار را می‌توان برای تولید یک تابع درهم‌ساز امن، برای عمل روی پیامی با هر طول بکار برد. مسألهٔ طراحی یک تابع درهم‌ساز امن به مسألهٔ طراحی یک تابع فشرده‌ساز مقاوم در مقابل تصادم که روی ورودی‌هائی با طول ثابت کار می‌کنند، برمی‌گردد. ثابت شده است که این برخورد، روش مطمئنی است و طرح‌های جدیدتر تنها ساختار را پالایش کرده و به طول کُد افزوده‌اند.



شکل ۵-۳ پردازش SHA-512 یک بلوک ۱۰۲۴- بیتی

در این بخش نگاهی به دو تابع درهم‌ساز امن دیگر، که علاوه بر SHA پذیرش تجاری یافته‌اند، می‌اندازیم.

الگوریتم چکیدهٔ پیام MD5

الگوریتم چکیدهٔ پیام MD5 (RFC 1321) بتوسط Ron Rivest طراحی گردید. تا چندین سال قبل که هنوز نگرانی‌های مربوط به حملهٔ همه جانبه و کشف رمز جدی نبود، MD5 پراستفاده‌ترین الگوریتم درهم‌سازی امن بود. الگوریتم بعنوان ورودی، یک پیام با طول اختیاری را پذیرفته و بعنوان خروجی یک چکیدهٔ ۱۲۸- بیتی از پیام را تولید می‌کند. ورودی بصورت بلوک‌های ۵۱۲- بیتی مورد پردازش قرار می‌گیرد.

همینطور که سرعت پردازش‌گرها افزایش یافته است، امنیت یک کُد ۱۲۸- بیتی نیز زیر سؤال رفته است. میتوان نشان دادکه پیچیدگی رسیدن به دو پیام متفاوت که دارای یک چکیده باشند در مرز $۲^{۶۴}$ عملیات قرار داشته در حالی‌که پیچیدگی پیداکردن یک پیام با یک چکیدهٔ مورد نظر در حد $۲^{۱۲۸}$ عملیات است. رقم قبلی برای امنیت خیلی کوچک است. علاوه برآن تعدادی عملیات شکستن رمز صورت پذیرفته است که آسیب‌پذیری MD5 در مقابل کشف رمز را نشان می‌دهد [BERS92,BOER93,DOBB96].

Whirlpool

Whirlpool [BARR03,STAL06b] بتوسط Vincent Rijmen اهل بلژیک و سهیم در اختراع الگوریتم Rijndael (الگوریتمی که بعنوان استاندارد پیشرفتهٔ رمزنگاری AES پذیرفته شد) و Paulo Barreto که یک رمزنگار برزیلی است طراحی شده است. Whirlpool یکی از تنها دو تابع درهم‌سازی است که مورد تأئید NESSIE (New European Schemes for Signatures, Integrity, and Encryption) می‌باشد [PREN02]. پروژهٔ NESSIE یک تلاش اتحادیهٔ اروپا برای فراهم کردن یک مجموعه از توابع نیرومند و متنوع رمزنگاری است که شامل رمزهای قالبی، رمزهای دنباله‌ای، رمزهای متقارن، توابع درهم‌ساز و کُدهای اعتبارسنجی پیام می‌باشد.

Whirlpool بر مبنای استفاده از یک رمزقالبی برای تابع فشرده‌سازی قرار دارد. Whirpool از یک رمز قالبی استفاده می‌کند که بطور اخص برای استفاده در تابع hash طراحی شده است و غیرمحتمل است که هرگز بعنوان یک تابع رمزنگاری تنها بکار رود. علت این امر این است که طراحان می‌خواسته‌اند از یک رمز قالبی با امنیت و بهره‌وری AES ولی با یک طول hash که امنیت بالقوه‌ای برابر SHA-512 را فراهم نماید، استفاده کنند. نتیجهٔ این کار تولید رمز قالبی W است که ساختاری مشابه AES داشته و از همان توابع ابتدائی استفاده می‌کند ولی اندازهٔ بلوک و اندازهٔ کلید آن ۵۱۲ بیت است.

الگوریتم بعنوان ورودی یک پیام با ماکزیمم طول کمتر از $۲^{۲۵۶}$ بیت را قبول کرده و یک چکیدهٔ پیام ۵۱۲- بیتی را تولید می‌کند. ورودی در بلوک‌های ۵۱۲- بیتی پردازش می‌گردد

HMAC

در سال‌های اخیر تمایل فزاینده‌ای به تولید یک MAC از یک کُد hash رمزی همانند SHA-1 وجود داشته است. محرک‌های این علاقه چنین بوده‌اند:

- توابع رمزی hash معمولاً سریع‌تر از الگوریتم‌های رمزنگاری متقارن مثل DES اجرا می‌شوند.
- کُدهای کتابخانه‌ای توابع hash بصورت گسترده‌ای در دسترس‌اند.

یک تابع درهم‌ساز مثل SHA-1 برای استفاده بصورت MAC طراحی نشده است و بنابراین نمی‌تواند مستقیماً برای این منظور بکار رود، زیرا متکی به یک کلید سرّی نیست. پیشنهادهای متعددی برای بکارگرفتن یک کلید سرّی در یک الگوریتم درهم‌سازی موجود داده شده است. روشی که بیشترین پذیرش را داشته است HMAC [BELL96a,BELL96b] است. HMAC تحت عنوان RFC 2104 منتشرشده و بعنوان یک انتخاب قطعی برای امنیت IP پذیرفته شده است. از HMAC همچنین در پروتکل‌های اینترنت دیگری چون امنیت لایهٔ حمل و نقل (TLS که بزودی جایگزین SSL خواهد شد) و اسناد امن الکترونیک (SET) استفاده می‌شود.

اهداف طراحی HMAC

RFC 2104 اهداف طراحی زیر برای HMAC را لیست کرده است:

- از توابع hash موجود بدون تغییرات استفاده کند، علی‌الخصوص توابعی که از نظر نرم‌افزاری خوب عمل کرده و برای آنها برنامهٔ آزاد و فراهم وجود دارد.
- اجازه دهد تا در صورتی که توابع hash امن‌تری پیدا شده و یا مورد نیاز واقع شوند، آن توابع بتوانند جایگزین تابع hash موجود در آن گردند.
- بتواند عملکرد اولیهٔ تابع hash را حفظ کرده، بدون اینکه کیفیت آن تنزل قابل ملاحظه‌ای یابد.
- بتواند کلیدها را به روش ساده‌ای مورد استفاده قرار دهد.
- تحلیل رمزنگاری قابل‌فهمی از قدرت مکانیسم اعتبارسنجی، بر اساس فرضیات معقول نسبت به تابع hash ارائه دهد.

اولین دو هدف در مورد پذیرش HMAC دارای اهمیت است. HMAC با تابع درهم‌ساز بصورت یک «جعبهٔ سیاه» برخورد می‌کند. این امر دو حسن دارد. اول اینکه یک فرم اجرائی از یک تابع hash می‌تواند بصورت یک مدول اجرائی در HMAC بکار رود. در این صورت بخش عمدهٔ کُد برنامهٔ HMAC از قبل آماده بوده و می‌تواند بدون دست خوردن مورد استفاده قرار گیرد. ثانیاً اگر روزی بخواهیم تا یک تابع hash در یک اجرای HMAC را عوض کنیم، تنها کار لازم این است که مدول تابع hash جاری را برداشته و مدول جدید را جایگزین آن نمائیم. این کار را می‌توان در صورتی که تابع hash سریع‌تری مورد نیاز باشد انجام داد. مهم‌تر اینکه اگر امنیت تابع hash موجود در HMAC به مخاطره افتد، امنیت HMAC را بسادگی می‌توان با جایگزین کردن تابع hash آن با یک تابع hash امن‌تر جبران نمود.

آخرین هدف طراحی در لیست قبل در واقع مزیت عمدهٔ HMAC بر سایر روش‌های پیشهادشده مبتنی بر درهم‌سازی بوده است. HMAC در صورتی امن خواهد بود که تابع hash درونی آن دارای توان‌های رمزنگاری معقولی باشد. بعداً در این بخش به این نکته برمی‌گردیم، ولی فعلاً ساختار HMAC را بررسی می‌کنیم.

الگوریتم HMAC

شکل ۶-۳ عملیات HMAC را نشان می‌دهد. واژه‌های زیر را تعریف می‌کنیم:

H = تابع hash لحاظ شده در HAMC (مثل SHA-1)

M = پیام ورودی به HMAC (شامل بیت‌های لائی مشخص شده در تابع hash)

Y_i = بلوک i ام M $0 \le i \le (L\text{-}1)$

L = تعداد بلوک‌ها در M

b = تعداد بیت‌ها در یک بلوک

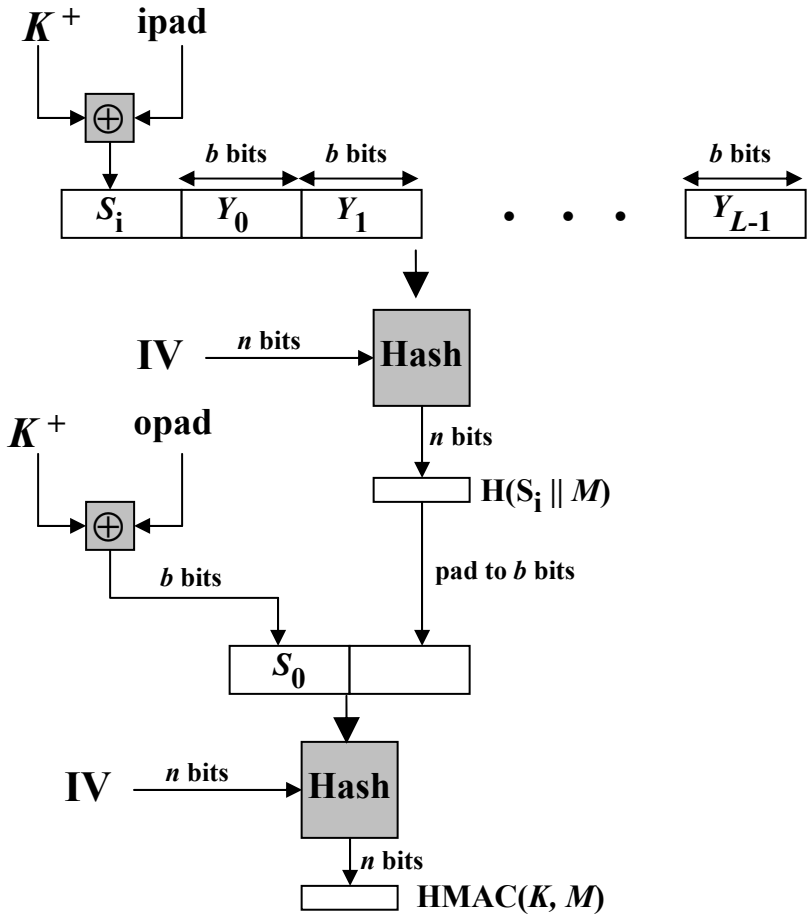
n = طول کُد hash تولیدشده بتوسط تابع درهم‌ساز

K =کلید سرّی. اگر طول کلید از b بزرگتر است، کلید وارد تابع hash شده تا یک کلید n- بیتی تولید شود. طول توصیه شده برای کلید بزرگتر و یا مساوی n است.

K^+ = K که بتوسط قراردادن صفرهائی در سمت چپ آن طویل‌تر شده تا طول آن به b بیت برسد.

ipad = 00110110 (معادل 36 هکزادسیمال) که $b/8$ بار تکرار شده است.

opad = 01011100 (معادل 5C هکزادسیمال) که $b/8$ بار تکرار شده است.



شکل ۶-۳ ساختار HMAC

با این تعاریف HMAC را می‌توان بصورت زیر نشان داد:

$$\text{HMAC}(K, M) = \text{H}[(K^+ \oplus \text{opad}) \,||\, \text{H}[(K^+ \oplus \text{ipad}) \,||\, \text{M}]]$$

که با کلمات چنین می‌شود:

۱- صفرهائی را به سمت چپ K اضافه کنید تا یک دنبالهٔ b- بیتی K^+ حاصل شود (مثلاً اگر K دارای طول ۱۶۰ بیت بوده و $b = ۵۱۲$ باشد، آنگاه K با ۴۴ بایت 0 کامل می‌شود).

۲- K^+ را بیت- به- بیت با ipad بصورت XOR درآورده تا بلوک b- بیتی S_i حاصل شود.

۳- M را به S_i وصل (جمع رشته‌ای) کنید.

۴- H را به دنبالهٔ تولیدشده در قدم سوم اعمال نمائید.

۵- K^+ را با opad، XOR نموده تا بلوک b- بیتی S_o حاصل شود.

۶- نتیجهٔ hash قدم چهارم را به S_o وصل (جمع رشته‌ای) کنید.

۷- H را به دنبالهٔ تولیدشده در قدم ششم اعمال کرده و نتیجه را خارج سازید.

توجه شود که XOR با ipad باعث عوض شدن نیمی از بیت‌های K می‌شود. بطریق مشابه، XOR کردن با opad باعث تعویض نیمی از بیت‌های K در موقعیت‌های متفاوت دیگری می‌شود. در واقع با عبوردادن S_i و S_o از الگوریتم درهم‌سازی، بطور شبه‌تصادفی دو کلید از K تولید کرده‌ایم.

برای پیام‌های طولانی، زمان اجرای HMAC بایستی تقریباً برابر زمان اجرای تابع hash درون آن باشد. HMAC سه اجرای تابع hash درون خود را در بردارد (برای S_i و S_o و بلوکی که از hash درونی حاصل می‌شود).

۳-۳ اصول رمزنگاری کلید- عمومی

در ردیف اهمیت رمزنگاری متقارن، رمزنگاری کلید- عمومی قرار دارد که در اعتبارسنجی پیام و توزیع کلید مورد استفاده است. در این بخش ابتدا نگاهی به مفهوم رمزنگاری کلید- عمومی انداخته و سپس نگاهی ابتدائی به مقولهٔ توزیع کلید می‌اندازیم. بخش ۳-۴ دوقلم از مهم‌ترین الگوریتم‌های کلید عمومی یعنی RSA و Diffie-Hellman را بررسی می‌کند. در بخش ۳-۵ امضاءهای دیجیتال را معرفی می‌کنیم.

ساختار رمزنگاری کلید- عمومی

رمزنگاری کلید- عمومی که برای اولین بار بتوسط Diffie وHellman درسال۱۹۷۶درمعرض عموم قرار گرفت[DIFF76]، اولین جهش انقلابی واقعی در رمزنگاری در طول هزاران سال است. از یک دید، الگوریتم‌های کلید- عمومی بجای اینکه شامل عملیات ساده‌ای بر روی بیت‌ها باشند، مبتنی بر توابع ریاضی‌اند. مهم‌تر این‌که رمزنگاری کلید- عمومی بر خلاف رمزنگاری سنّتی که تنها از یک کلید استفاده می‌کند شامل استفاده از دو کلید مجزا بوده و نامتقارن است. استفاده از دو کلید نتایج فوق‌العاده‌ای در زمینهٔ محرمانگی، توزیع کلید و اعتبارسنجی به بارمی‌آورد.

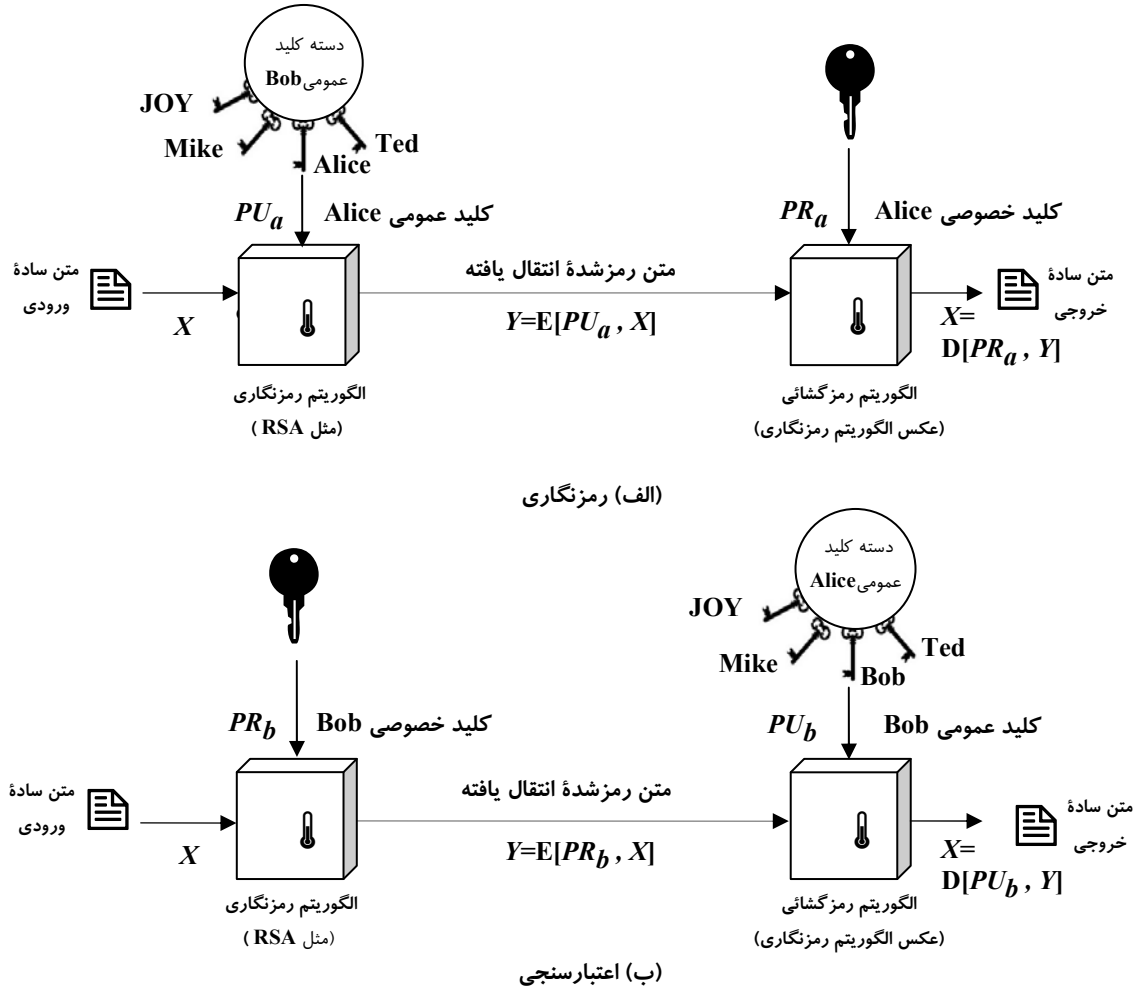
قبل از این‌که جلوتر برویم، بایستی در ابتدا به چندین اشتباه معمول در مورد رمزنگاری کلید- عمومی اشاره کنیم. یکی این‌که گفته می‌شود رمزنگاری کلید- عمومی در مقابل شکستن رمز امن‌تر از رمزنگاری سنّتی است. در واقع امنیت هر روش رمزنگاری بستگی به: (۱) طول کلید و (۲) کار محاسباتی لازم برای شکستن رمز دارد. از نظر اصولی هیچ چیز در مورد رمزنگاری سنّتی و یا رمزنگاری کلید- عمومی وجود ندارد که یکی را نسبت به دیگری در مقابل شکستن رمز ارجحیت دهد. اشتباه دوم این است که رمزنگاری کلید- عمومی را یک تکنیک عام تصور می‌کنند که رمزنگاری سنّتی را از میدان بدر کرده است. برعکس، بعلت سربارهٔ محاسباتی روش‌های رمزنگاری کلید- عمومی موجود، هیچ پیش‌بینی احتمالی در مورد زمان واگذاری رمزنگاری سنّتی نمی‌توان کرد. بالاخره این احساس وجود دارد که در استفاده از رمزنگاری کلید- عمومی، توزیع کلید در مقایسه با عملیات پیچیده‌ای که در مراکز توزیع کلید رمزنگاری سنّتی وجود دارد، کار ساده‌ای است. واقعیت این است که در این رمزنگاری جدید هم نوعی پروتکل که معمولاً شامل یک عامل مرکزی است مورد نیاز بوده و روش‌های امر نه ساده‌تر و نه بهره‌ورتر از آنهائی هستند که برای رمزنگاری سنّتی بکار می‌روند.

یک روش رمزنگاری کلید- عمومی دارای شش جزء است (شکل ۳-۷الف):

- **متن ساده**: این پیام خوانا و یا دیتائی است که به ورودی الگوریتم داده می‌شود.
- **الگوریتم رمزنگاری**: الگوریتم رمزنگاری، تبدیل‌های مختلفی را روی دیتا انجام می‌دهد.

- **کلید عمومی و کلید خصوصی**: یک جفت کلید است که طوری انتخاب می‌شوند که اگر از یکی برای رمزنگاری استفاده شود، از دیگری برای رمزگشائی استفاده خواهد شد. تبدیل‌های واقعی انجام‌شده بتوسط الگوریتم رمزنگاری، وابسته به کلید عمومی و یا کلید خصوصی است که در ورودی از آن استفاده شده است.
- **متن رمزشده**: این پیام درهم‌ریخته‌ای است که بعنوان خروجی تولید می‌شود. این پیام بستگی به متن ساده و کلید دارد. برای یک پیام واحد، دو کلید مختلف دو متن رمزشدهٔ مختلف ایجاد خواهند کرد.
- **الگوریتم رمزگشائی**: این الگوریتم متن رمزشده و کلید ذیربط را پذیرفته و متن سادهٔ اولیه را تولید می‌کند.

همانطور که از نام آن برمی‌آید، از بین دو کلید، کلید عمومی برای همه شناخته‌شده بوده و می‌توانند آن را بکار گیرند در حالی‌که کلید خصوصی تنها برای صاحب آن شناخته شده است. یک الگوریتم رمزنگاری کلید- عمومی با مصرف عام، متکی بر یک کلید برای رمزنگاری و یک کلید متفاوت ولی مرتبط برای رمزگشائی است.



شکل ۷-۳ رمزنگاری کلید- عمومی

قدم‌های اصلی اجرای روش بقرار زیراند:

۱- هر کاربر یک زوج کلید که برای رمزنگاری و رمزگشائی پیام‌ها بکار خواهد رفت را تولید می‌کند.
۲- هر کاربر یکی از دو کلید را در یک مخزن عمومی و یا فایل قابل دسترس دیگر قرار می‌دهد. این همان کلید عمومی است. کلید نظیر آن مخفی نگاه داشته خواهد شد. همان‌طور که شکل ۷-۳الف نشان می‌دهد، هر کاربر مجموعه‌ای از کلیدهای عمومی که از دیگران گرفته است را در اختیار دارد.
۳- اگر Bob بخواهد یک پیام خصوصی برای Alice بفرستد، Bob پیام را با استفاده از کلید عمومی Alice رمزنگاری خواهد کرد.
۴- وقتی Alice پیام را دریافت نمود، او پیام را با استفاده از کلید خصوصی خود رمزگشائی می‌کند. هیچ گیرندهٔ دیگری نمی‌تواند پیام را رمزگشائی کند زیرا تنها Alice از کلید خصوصی Alice مطلع است

در چنین روشی، تمام شرکت‌کنندگان به کلیدهای عمومی دسترسی دارند و کلیدهای خصوصی بطور محلی و بتوسط هر شرکت‌کننده تولید می‌شود و بنابراین لازم نیست تا توزیع شوند. تا زمانی که یک کاربر از کلید خصوصی خود محافظت می‌نماید، ارتباطات ورودی امن خواهند بود. در هرزمان، یک کاربر می‌تواند کلید خصوصی خود را عوض کرده و کلید عمومی مرتبط با آن را برای جایگزینی کلید قبلی به اطلاع عموم برساند.

کلید استفاده‌شده در رمزنگاری متقارن را معمولاً **کلید سرّی** می‌گویند. دو کلیدی که در رمزنگاری کلید- عمومی بکار می‌رود را **کلید عمومی** و **کلید خصوصی** می‌نامند. کلید خصوصی بدون استثناء سرّی نگاه داشته می‌شود ولی از این جهت آن را کلید خصوصی، و نه کلید سرّی، می‌گویند که با رمزنگاری سنّتی اشتباه نشود.

کاربردهائی برای سیستم‌های رمزنگاری کلید- عمومی

قبل از این‌که جلوتر برویم، لازم است یکی از جنبه‌های سیستم‌های رمزنگاری کلید- عمومی که ممکن است باعث گمراهی شود را روشن نمائیم. سیستم‌های کلید- عمومی با استفاده از یک نوع الگوریتم رمزنگاری با دو کلید که یکی از آنها خصوصی نگاه داشته شده و دیگری در دسترس عموم است، مشخص می‌گردد. بر حسب اینکه نوع کاربرد چیست، فرستنده یا از کلید خصوصی خود و یا از کلید عمومی گیرنده و یا از هردو کلید برای انجام نوعی عمل رمزنگاری استفاده می‌کند. در مفهوم وسیع، موارد استفاده از سیستم‌های رمزنگاری کلید- عمومی را می‌توانیم به سه دسته تقسیم کنیم:

- **رمزنگاری/ رمزگشائی**: فرستنده یک پیام را با کلید عمومی گیرنده به رمز درمی‌آورد.
- **امضاء دیجیتال**: فرستنده یک پیام را با کلید خصوصی خود «امضاء» می‌کند. امضاء با اعمال یک الگوریتم رمزنگاری به پیام و یا به بلوک کوچکی از دیتا که تابعی از پیام است انجام می‌شود.
- **مبادلهٔ کلید**: دوطرف برای مبادلهٔ یک کلید اجلاس همکاری می‌کنند. چندین روش مختلف که شامل کلید یا کلیدهای خصوصی یکی از طرفین و یا هردوی آنهاست وجود دارد.

بعضی از الگوریتم‌ها برای هرسه کاربرد مناسب‌اند در حالی‌که برخی دیگر تنها برای یک یا دو کاربرد به درد می‌خورند. جدول ۲-۳ کاربردهائی که مورد حمایت الگوریتم‌های مورد بحث در این فصل یعنی RSA و Diffie Hellman هستند را نشان می‌دهد. جدول همچنین شامل استاندارد امضاء دیجیتال (DSS) و رمزنگاری خَم بیضوی نیز هست که بعداً در همین فصل به آنها اشاره خواهد شد.

جدول ۲-۳ کاربردهای سیستم‌های رمزنگاری کلید- عمومی

الگوریتم	رمزنگاری/ رمزگشائی	امضاء دیجیتال	مبادلهٔ کلید
RSA	بلی	بلی	بلی
Diffie-Hellman	خیر	خیر	بلی
DSS	خیر	بلی	خیر
Elliptic Curve	بلی	بلی	بلی

لازمه‌های رمزنگاری کلید- عمومی

سیستم رمزنگاری نشان‌داده‌شده در شکل ۷-۳ وابسته به یک الگوریتم رمزنگاری است که بر مبنای دو کلید مرتبط بنا نهاده شده است. Diffie و Hellman چنین سیستمی را مفروض دانسته و بدون این‌که نشان دهند که واقعاً چنین الگوریتمی وجود دارد، شرایطی که چنین الگوریتم‌هائی باید داشته باشند را چنین بیان کرده‌اند[DIFF76]:

۱- از نظر محاسباتی برای طرف B ساده است تا یک زوج کلید (کلید عمومی PU_b و کلید خصوصی PR_b) تولید کند.

۲- از نظر محاسباتی برای فرستندهٔ A ساده است تا با دانستن کلید عمومی و پیامی که باید رمزنگاری شود(M)، متن رمزشدهٔ نظیر را تولیدکند.

$$C = \mathrm{E}\,(PU_b, M)$$

۳- از نظر محاسباتی برای گیرندهٔ B ساده است تا متن رمزشدهٔ نتیجه را با استفاده از کلید خصوصی رمزگشائی نماید:

$$M = \mathrm{D}\,(PR_b, C) = \mathrm{D}\,[PR_b, \mathrm{E}\,(PU_b, M)]$$

۴- از نظر محاسباتی برای یک دشمن میسر نیست که با داشتن کلید عمومی PU_b، کلید خصوصی PR_b را بدست آورد.

۵- از نظر محاسباتی برای یک دشمن میسر نیست که با داشتن کلید عمومی PU_b و یک متن رمزشده C، پیام اولیهٔ M را استخراج نماید.

می‌توانیم شرط مشخصی را که اگرچه مفید است ولی برای کاربردهای کلید عمومی لازم نیست اضافه نمائیم:

۶- هریک از دو کلید مربوط می‌تواند برای رمزنگاری بکار رود و از کلید دیگر برای رمزگشائی استفاده خواهد شد.

$$M = \mathrm{D}\,[PU_b, \mathrm{E}\,(PR_b, M)] = \mathrm{D}\,[PR_b, \mathrm{E}\,(PU_b, M)]$$

۳-۴ الگوریتم‌های رمزنگاری کلید- عمومی

دو مورد از پراستفاده‌ترین الگوریتم‌های کلید- عمومی، RSA و Diffie- Hellman هستند. ما در این بخش هر دوی آنها را بررسی نموده و دو الگوریتم دیگر را نیز بطور مختصر معرفی خواهیم کرد.

الگوریتم رمزنگاری کلید- عمومی RSA

یکی از اولین روش‌های کلید- عمومی است که در سال ۱۹۷۷ بتوسط Adi Shamir،Ron Rivest و Len Adleman در دانشگاه MIT شکل گرفت و برای اولین بار در ۱۹۷۸ منتشر گردید[RIVE78]. روش RSA از آن زمان بعنوان پذیرفته‌شده‌ترین و پرکاربردترین روش رمزنگاری کلید- عمومی در اوج قرارداشته است. RSA یک رمز قالبی است که درآن متن ساده و متن رمزشده اعداد صحیحی بین 0 تا n-1 برای مقداری از n می‌باشند.

برای بلوک متن سادهٔ M و بلوک متن رمزشدهٔ C، رمزنگاری و رمزگشائی بشکل زیر است:

$$C = M^{e} \bmod n$$
$$M = C^{d} \bmod n = (M^{e})^{d} \bmod n = M^{ed} \bmod n$$

هم گیرنده و هم فرستنده بایستی از مقادیر n و e مطلع بوده ولی فقط گیرنده اندازهٔ d را می‌داند. این یک الگوریتم رمزنگاری کلید- عمومی با کلید عمومی $PU = \{e,n\}$ و کلید خصوصی $PR = \{d,n\}$ است. برای این‌که این الگوریتم برای رمزنگاری کلید- عمومی رضایت‌بخش باشد، نیازهای زیر بایستی برآورده شوند:

۱- بایستی ممکن باشد مقدار e، d و n را طوری پیدا نمود که برای جمیع مقادیر $M < n$، $M^{ed} = M \bmod n$ باشد.
۲ - محاسبهٔ M^{e} و C^{d} برای جمیع مقادیر $M < n$ نسبتاً آسان باشد.
۳ - با داشتن e و n، تعیین d مقدور نباشد.

دو نیاز اول به سهولت برآورده می‌شوند. نیاز سوم برای مقادیر بزرگ e و n قابل حصول است.

شکل ۸-۳ الگوریتم RSA را خلاصه کرده است. در ابتدا دو عدد اول p و q انتخاب و حاصلضرب آنها n محاسبه می‌گردد که پیمانهٔ (module) رمزنگاری و رمزگشائی است. سپس به کمیت $\phi(n)$ نیاز داریم که تابع Euler نامیده شده و تعداد اعداد صحیح مثبت کوچک‌تر از n و اول نسبت به n است. آنگاه یک عدد صحیح e طوری انتخاب می‌گردد که نسبت به $\phi(n)$ اول باشد [یعنی بزرگترین مقسوم‌علیه مشترک e و $\phi(n)$ مساوی ۱ باشد]. بالاخره d بعنوان معکوس ضربی n و بصورت modulo $\phi(n)$ محاسبه می‌شود. می‌توان نشان داد که d و e دارای خواص مطلوب هستند.

فرض کنید که کاربر A کلید عمومی خود را معـرفی کرده و کاربر B می‌خواهد پیـــام M را برای A بفرستد. B مقدار $C = M^{e} \pmod n$ را محاسبه کرده و C را ارسال می‌کند. کاربر A پس از دریافت مـتن رمزشـده، $M = C^{d} \bmod n$ را محاسبه و آن را از رمز درمی‌آورد.

مثالی از [SING99] در شکل ۹-۳ نشان داده شده است. برای این مثال کلیدها بصورت زیر تولید شده‌اند:

۱- دو عدد اول $p = 17$ و $q = 11$ را انتخاب کنید.
۲- $n = pq = 17\times11 = 187$ را محاسبه نمائید.
۳- $\phi(n) = (p\text{-}1)(q\text{-}1) = 16\times10 = 160$ را محاسبه نمائید.
۴- e را طوری انتخاب کنید که e نسبت به $\phi(n) = 160$ اول بوده و کمتر از آن هم باشد مثلاً $e = 7$.
۵- d را طوری انتخاب کنید که $de \bmod 160 = 1$ باشد. اندازهٔ صحیح $d = 23$ است زیرا $23\times7 = 1\times160+1$.

تولید کلید

p و q را انتخاب کنید. p و q هردو اول‌اند و $p \neq q$

$n = p \times q$ را محاسبه نمائید.

$\phi(n) = (p\text{-}1)(q\text{-}1)$ را بدست آورید.

عدد صحیح e را انتخاب کنید. بزرگترین مقسوم‌علیه مشترک $\phi(n)$ و e برابر ۱ است.

d را حساب کنید. $de \bmod \phi(n) = 1$

کلید عمومی $PU = \{e , n\}$

کلید خصوصی $PR = \{d , n\}$

رمزنگاری

متن ساده : $M < n$

متن رمزشده : $C = M^{e} \bmod n$

رمزگشائی

متن رمزشده : C

متن ساده : $M = C^{d} \bmod n$

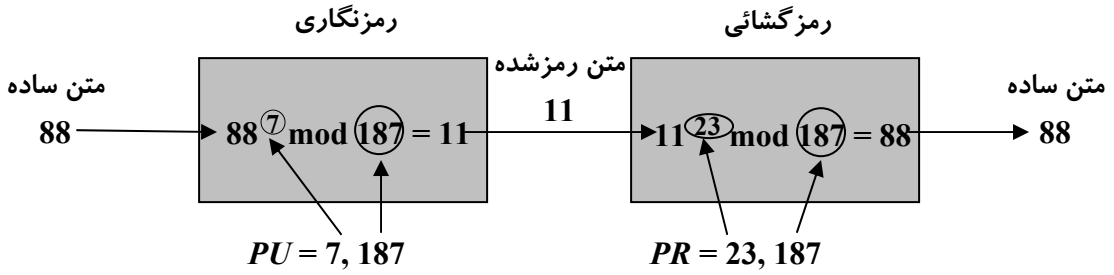
شکل ۸-۳ الگوریتم RSA

کلیدهای بدست‌آمده برابر کلید عمومی $PU = \{7,187\}$ و $PR = \{23,187\}$ هستند. مثال، استفاده از این کلیدها را برای یک ورودی متن ساده $M = 88$ نشان می‌دهد. برای رمزنگاری لازم است $C = 88^7 \bmod 187$ را حساب کنیم. با بکارگیری خواص محاسبات پیمانه‌ای، محاسبه چنین انجام می‌شود:

$$88^7 \bmod 187 = [(88^4 \bmod 187) \times (88^2 \bmod 187) \times (88^1 \bmod 187)] \bmod 187$$
$$88^1 \bmod 187 = 88$$
$$88^2 \bmod 187 = 7744 \bmod 187 = 77$$
$$88^4 \bmod 187 = 59{,}969{,}536 \bmod 187 = 132$$
$$88^7 \bmod 187 = (88 \times 77 \times 132) \bmod 187 = 894{,}432 \bmod 187 = 11$$

برای رمزگشائی $M = 11^{23} \bmod 187$ را محاسبه می‌کنیم.

$$11^{23} \bmod 187 = [(11^1 \bmod 187) \times (11^2 \bmod 187) \times (11^4 \bmod 187) \times (11^8 \bmod 187) \times (11^8 \bmod 187)] \bmod 187$$
$$11^1 \bmod 187 = 11$$
$$11^2 \bmod 187 = 121$$
$$11^4 \bmod 187 = 14{,}641 \bmod 187 = 55$$
$$11^8 \bmod 187 = 214{,}358{,}881 \bmod 187 = 33$$
$$11^{23} \bmod 187 = (11 \times 121 \times 55 \times 33 \times 33) \bmod 187 = 79{,}720{,}254 \bmod 187 = 88$$



شکل ۹-۳ مثال الگوریتم RSA

دو روش برای شکست دادن الگوریتم RSA متصوّر است. اولی جستجوی کامل است: تمام کلیدهای ممکن را امتحان کنید. بنابراین هرچقدر تعداد بیت‌های e و d زیادتر باشد، الگوریتم امن‌تر خواهد بود. از طرفی چون هم در تولید کلید و هم در رمزنگاری/ رمزگشائی پای محاسبات پیچیده‌ای در میان است، هرچقدر اندازهٔ کلید بزرگتر باشد سیستم کندتر عمل خواهد کرد.

بیشتر بحث‌های مربوط به شکستن رمز RSA روی فاکتورگیری n به دو عدد اول متمرکز شده است. برای یک n بزرگ با فاکتورهای اول بزرگ، فاکتورگیری یک عمل مشکل است که البته در حال حاضر به سختی قبل نیست. جلوهٔ ویژهٔ این امر در سال ۱۹۷۷ اتفاق افتاد که سه مخترع RSA، خوانندگان مجلهٔ *Scientific American* را تشویق کردند تا رمزی را که آنها در ستون «بازی‌های ریاضی» Martin Gardner [GARD77] چاپ کرده بودند کشف کنند. آنها برای کشف رمز و استخراج متن سادهٔ پیام، جایزه‌ای برابر ۱۰۰ دلار تعیین کرده بودند و پیش‌بینی نموده بودند که این کار زودتر از یک میلیون میلیون سال (10^{12} سال) عملی نمی‌باشد. در آوریل ۱۹۹۴ یک گروه که روی اینترنت کار می‌کردند و ۱,۶۰۰ کامپیوتر را بکارگرفته بودند، پس از هشت ماه کار مدعی این جایزه شدند[LEUT94]. این مبارزه از یک کلید عمومی با اندازهٔ ۱۲۹ رقم دهدهی (طول n) و یا حدوداً ۴۲۸ بیت استفاده کرده بود. این نتیجه بهیچوجه RSA را بی اعتبار نمی‌کند بلکه معنی آن این است که بایستی از کلیدهای طولانی‌تر استفاده کرد. امروزه یک کلید با طول ۱,۰۲۴ بیت (تقریباً ۳۰۰ رقم دهدهی)، یک کلید مستحکم برای تقریباً تمام کاربردها بحساب می‌آید.

مبادلهٔ کلید Diffie-Hellman

نطفهٔ الگوریتم کلید- عمومی در یک مقاله بتوسط Diffie وHellman شکل گرفت که رمزنگاری کلید- عمومی را تعریف کرده است[DIFF76] و معمولاً از آن بعنوان مبادلهٔ کلید Diffie- Hellman یاد می‌شود. تعدادی از محصولات تجاری از این تکنیک مبادلهٔ کلید استفاده می‌کنند.

هدف الگوریتم این است که دو کاربر را قادر سازد که یک کلید سرّی را بصورت امن مبادله نمایند تا بعداً از این کلید برای رمزنگاری آتی پیام‌ها استفاده شود. خود الگوریتم منحصر به مبادلهٔ کلیدهاست.

مؤثربودن الگوریتم Diffie-Hellman متکی به پیچیدگی محاسبه لُگاریتم‌های گسسته است. لُگاریتم گسسته را می‌توان مختصراً چنین تعریف کرد: ابتدا یک ریشهٔ اولیه (primitive) یک عدد اول p را بعنوان عددی که توان‌های آن همهٔ اعداد صحیح 1 تا p-1 را تولید می‌کند تعریف می‌کنیم. یعنی اگر α یک ریشهٔ اولیهٔ عدد اول p باشد، آنگاه اعداد $\alpha \bmod p$، $\alpha^2 \bmod p$، و .... $\alpha^{p-1} \bmod p$ کاملاً متمایز بوده و شامل اعداد صحیح 1 تا p-1 با جایگشت‌هائی می‌باشند.

برای هر عدد صحیح b کوچک‌تر از p و یک ریشهٔ اولیهٔ α از عدداول p، می‌توان یک نمای یکتای i را طوری پیدا کرد که

$$b = \alpha^i \bmod p \qquad 0 \le i \le (p\text{-}1)$$

نمای i را لُگاریتم گسستهٔ یا اندیس b برای پایهٔ α به پیمانهٔ p نامند. این مقدار را به شکل $\mathrm{dlog}_{\alpha,p}(b)$ نشان می‌دهیم.

الگوریتم

با این زمینه می‌توانیم مبادلهٔ کلید Diffie-Hellman، که در شکل ۱۰-۳ خلاصه شده است، را تعریف کنیم. در این روش دو عدد شناخته شده وجود دارد: یک عدد اول q و یک عدد صحیح α که ریشهٔ اولیهٔ q است. فرض کنید که کاربران A و B بخواهند تا کلیدی را مبادله نمایند. کاربر A یک عدد صحیح $X_A < q$ را بصورت تصادفی انتخاب کرده و $Y_A = \alpha^{X_A} \bmod q$ را حساب می‌کند. بطریق مشابه، کاربر B بصورت مستقل یک عدد صحیح تصادفی $X_B < q$ را انتخاب کرده و $Y_B = \alpha^{X_B} \bmod q$ را حساب می‌کند. هرطرف مقدار X را سرّی نگاه داشته و مقدار Y را بطور آشکار در اختیار طرف مقابل می‌گذارد. کاربر A کلید را بصورت $K = (Y_B)^{X_A} \bmod q$ محاسبه نموده و کاربر B نیز کلید را بصورت $K = (Y_A)^{X_B} \bmod q$ محاسبه می‌کند. این دو محاسبه نتایج یکسانی را تولید می‌کنند.

$$\begin{aligned}
K &= (Y_B)^{X_A} \bmod q \\
&= (\alpha^{X_B} \bmod q)^{X_A} \bmod q \\
&= (\alpha^{X_B})^{X_A} \bmod q \\
&= \alpha^{X_B X_A} \bmod q \\
&= (\alpha^{X_A})^{X_B} \bmod q \\
&= (\alpha^{X_A} \bmod q)^{X_B} \bmod q \\
&= (Y_A)^{X_B} \bmod q
\end{aligned}$$

نتیجه این است که دو طرف یک کلید سرّی را مبادله کرده‌اند. علاوه براین چون X_A و X_B سرّی هستند، یک دشمن فقط می‌تواند با q، a، Y_A و Y_B کارکند و مجبور است برای تعیین کلید، یک لُگاریتم گسسته را حساب کند. برای مثال برای حمله به کلید سرّی کاربر B، دشمن بایستی $X_B = \mathrm{dlog}_{\alpha,q}(Y_B)$ را حساب کند. دشمن آنگاه خواهد توانست کلید K را بهمان نحوی که کاربر B آن را محاسبه می‌کند بدست آورد.

مقادیر قابل دسترس عمومی

عدد اول q

عدد α $\alpha < q$ بوده و α ریشهٔ سادهٔ q است.

تولید کلید کاربر A

مقدار خصوصی X_A را انتخاب کنید. $X_A < q$

مقدار عمومی Y_A را حساب کنید. $Y_A = \alpha^{X_A} \bmod q$

تولید کلید کاربر B

مقدار خصوصی X_B را انتخاب کنید. $X_B < q$.

مقدار عمومی Y_B را حساب کنید. $Y_B = \alpha^{X_B} \bmod q$

تولید کلید سرّی به توسط کاربر A

$K = (Y_B)^{X_A} \bmod q$

تولید کلید سرّی به توسط کاربر B

$K = (Y_A)^{X_B} \bmod q$

شکل ۱۰-۳ الگوریتم مبادلهٔ کلید Diffie-Hellman

امنیت مبادلهٔ کلید Diffie- Hellman در این حقیقت نهفته است که در حالی‌که محاسبهٔ نماهائی با مدول یک عدد اول نسبتاً ساده است، محاسبهٔ لُگاریتم گسسته کاری بس مشکل است. برای اعداد اول بزرگ، این کار غیرعملی است.

برای مثال، مبادلهٔ کلید بر اساس استفاده از عدد اول $q = 353$ و یک ریشهٔ اولیهٔ 353 که در این مورد $\alpha = 3$ است قرار دارد. A و B کلیدهای سرّی $X_A = 97$ و $X_B = 233$ را به‌ترتیب انتخاب نموده و هرکدام کلید عمومی خود را محاسبه می‌کنند:

A چنین حساب می‌کند $Y_A = 3^{97} \bmod 353 = 40$

B چنین حساب می‌کند $Y_B = 3^{233} \bmod 353 = 248$

پس از این که آنها کلیدهای عمومی خود را مبادله کردند، هریک خواهد توانست تا کلید سرّی مشترک را حساب کند:

A چنین حساب می‌کند $K = (Y_B)^{X_A} \bmod 353 = 248^{97} \bmod 353 = 160$

B چنین حساب می‌کند $K = (Y_A)^{X_B} \bmod 353 = 40^{233} \bmod 353 = 160$

فرض می‌کنیم که یک حمله کننده اطلاعات زیر را در اختیار داشته باشد:

$$q = 353 ;\ \alpha = 3 ;\ Y_A = 40 ;\ Y_B = 248$$

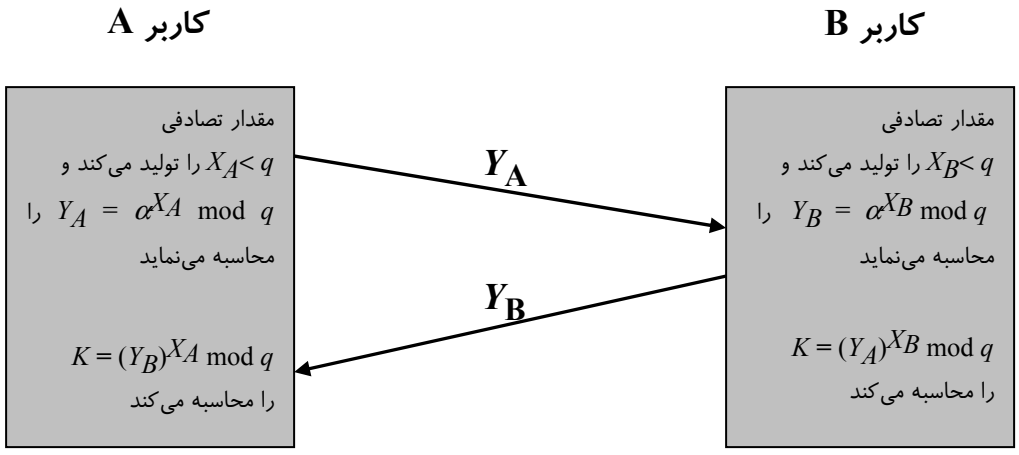
در این مثال ساده، با یک حملهٔ همه جانبه ممکن است که کلید سرّی 160 را بدست آورد. علی‌الخصوص حمله‌کنندهٔ E می‌تواند کلید مشترک را با حل معادلهٔ $3^a \bmod 353 = 40$ و یا معادلهٔ $3^b \bmod 353 = 248$ پیدا کند. روش جستجوی همه جانبه این است که تمام توان‌های 3 به پیمانهٔ 353 (modulo 353) را محاسبه کنیم تا نتیجه 40 و یا 248 شود. نتیجهٔ مطلوب وقتی است که به 97 برسیم زیرا $3^{97} \bmod 353 = 40$ است.

با اعداد بزرگ‌تر، حل مسأله مقدور نخواهد بود.

پروتکل‌های مبادلهٔ کلید

شکل ۱۱-۳ یک پروتکل ساده که از محاسبات Diffie-Hellman استفاده می‌کند را نشان می‌دهد. فرض کنید که کاربر A می‌خواهد ارتباطی را با کاربر B برقرار کرده و از یک کلید سرّی برای رمزنگاری پیام‌ها در این ارتباط استفاده نماید.کاربر A می‌تواند یک کلید سرّی یکبارمصرف X_A را تولید کرده، Y_A را حساب نموده و آن را برای کاربر B بفرستد. کاربر B می‌تواند در پاسخ یک کلید سرّی X_B تولید نموده، Y_B را محاسبه کرده و Y_B را برای کاربر A بفرستد. هر دو کاربر حالا می‌توانند کلید را محاسبه نمایند. مقادیر q و α بایستی قبلاً معلوم باشند. در حالت دیگر کاربر A می‌تواند مقادیر q و α را انتخاب کرده و آنها را در پیام اول بگنجاند.

بعنوان استفادهٔ دیگری از الگوریتم Diffie-Hellman فرض کنید که گروهی از کاربران (مثلاً تمام کاربران یک شبکهٔ LAN) هرکدام یک مقدار سرّی بادوام X_A را تولید کرده و اندازهٔ عمومی Y_A را محاسبه نمایند. این مقادیر عمومی به همراه مقادیر عمومی q و α در یک فهرست مرکزی نگهداری می‌شوند. در هر زمان کاربر B می‌تواند به مقدار آشکار کاربر A دسترسی یافته، یک کلید سرّی را حساب کرده و از آن برای ارسال یک پیام رمزی برای کاربر A استفاده نماید. اگر فهرست مرکزی قابل اطمینان باشد، آنگاه این نوع ارتباط هم محرمانگی و هم تا حدودی اعتبارسنجی را فراهم می‌کند. چون فقط A و B می‌توانند کلید را تعیین نمایند، هیچ کاربر دیگری نمی‌تواند پیام را بخواند (محرمانگی). دریافت‌کنندهٔ A می‌داند که تنها کاربرB می‌توانسته است پیامی با این کلید را خلق کرده باشد (اعتبارسنجی). اما این تکنیک در برابر حملات بازخوانی (replay) امن نیست.



شکل ۱۱-۳ مبادلهٔ کلید Diffie-Hellman

حملهٔ Man-in-the-Middle

پروتکل نشان داده شده در شکل ۱۱-۳ در برابر حملهٔ man-in-the-middle ناامن است. فرض کنید که Alice و Bob می‌خواهند کلیدهائی را مبادله کنند و Darth دشمن فرضی است. حمله چنین جلو می‌رود:

۱- Darth با تولید دو کلید خصوصی X_{D1} و X_{D2} و سپس محاسبهٔ کلیدهای عمومی Y_{D1} و Y_{D2} خود را برای حمله آماده می‌کند.
۲- Alice اندازهٔ Y_A را برای Bob ارسال می‌کند.
۳- Darth اندازهٔ Y_A را دزدیده و Y_{D1} را برای Bob می‌فرستد. همچنین Darth $K2 = (Y_A)^{X_{D2}} \bmod q$ را محاسبه می‌کند.
۴- Bob اندازهٔ Y_{D1} را دریافت کرده و $K1 = (Y_{D1})^{X_B} \bmod q$ را محاسبه می‌کند.
۵- Bob اندازهٔ Y_B را برای Alice می‌فرستد.
۶- Darth اندازهٔ Y_B را دزدیده و Y_{D2} را برای Alice ارسال می‌کند. همچنین Darth $K1 = (Y_B)^{X_{D1}} \bmod q$ را محاسبه می‌کند.
۷- Alice اندازهٔ Y_{D2} را دریافت کرده و $K2 = (Y_{D2})^{X_A} \bmod q$ را محاسبه می‌کند.

در این نقطه، Bob و Alice تصور می‌کنند که یک کلید سرّی را در اشتراک دارند درحالی که واقعیت این است که Bob و Darth کلید K1 را در اشتراک داشته و Alice و Darth نیز کلید K2 را در اشتراک دارند. تمام ارتباطات آتی بین Bob و Alice به طریق زیر لو خواهند رفت:

۱- Alice یک پیام رمزنگاری شدهٔ $E(K2, M)$: M را ارسال می‌کند.
۲- Darth پیام رمزنگاری شده را دزدیده و آن را رمزگشائی کرده تا M را بدست آورد.
۳- Darth مقدار $E(K1, M)$ یا $E(M')$ را برای Bob ارسال می‌کند که درآن M' هر پیام دلخواهی است.

در بند ۲، Darth تنها بسادگی می‌خواهد روی ارتباطات شنود داشته باشد. در بند ۳، Darth می‌خواهد که پیام‌های ارسالی برای Bob را تغییر دهد.

پروتکل مبادلهٔ کلید به چنین حمله‌ای آسیب‌پذیر است زیرا هویت طرفین ارتباط در آن احراز نمی‌گردد. این آسیب‌پذیری را می‌توان با استفاده از امضاء دیجیتال و گواهی‌نامه‌های کلید- عمومی که بعداً در این فصل و فصل ۴ در مورد آنها بحث خواهد شد،از بین برد.

سایر الگوریتم‌های رمزنگاری کلید- عمومی

دو الگوریتم کلید- عمومی دیگر نیز پذیرش تجاری یافته‌اند: DSS و رمزنگاری خَم بیضوی.

استاندارد امضاء دیجیتال Digital Signiture Standard (DSS)

انستیتوی ملی استانداردها و تکنولوژی امریکا (NIST)، استاندارد فدرال پردازش اطلاعات FIP PUB 186 را با نام استاندارد امضاء دیجیتال (DSS) منتشر کرده است. DSS از SHA-1 استفاده کرده و یک تکنیک جدید امضاء دیجیتال بنام الگوریتم امضاء دیجیتال (DSA) را معرفی می‌کند. DSS ابتدا در سال ۱۹۹۱ پیشنهاد گردید و در سال ۱۹۹۳ در پاسخ به

نظراتی که در مورد امنیت روش مطرح گردیده بود مورد بازنگری قرار گرفت. تغییر کوچکی هم در سال ۱۹۹۶ در آن بوجود آمد. DSS از الگوریتمی استفاده می‌کند که تنها برای فراهم‌آوردن تابع امضاء دیجیتال طراحی شده است. این روش نمی‌تواند برای رمزنگاری یا مبادلهٔ کلید بکار رود.

رمزنگاری خَم بیضوی Elliptic-Curve Cryptography (ECC)

اکثر محصولات و استانداردهائی که از رمزنگاری کلید- عمومی و امضاءهای دیجیتال استفاده می‌کنند، RSA را بکار می‌برند. اندازهٔ طول بیت برای RSA امن در طول سال‌های اخیر افزایش یافته و این امر بار پردازش سنگین‌تری را روی کاربردهائی که از RSA استفاده می‌کنند اعمال کرده است. این معضل دارای جلوه‌های متعددی است که علی‌الخصوص در سایت‌های تجارت الکترونیک که اسناد مالی زیادی باید بصورت امن مبادله شوند، نمود بیشتری دارد. اخیراً یک سیستم رقیب، RSA را به مبارزه طلبیده است: رمزنگاری خَم بیضوی (ECC). هم‌اکنون ECC در تلاش‌های استانداردسازی که شامل IEEE P1363 است ویژگی‌های خود را نشان داده است.

جاذبهٔ اصلی ECC در مقایسه با RSA این است که بنظر می‌رسد تکنیک جدید همان امنیت را برای اندازهٔ بیت بسیار کمتری بوجود می‌آورد و در نتیجه سربارهٔ پردازش کم می‌شود. از طرف دیگر اگرچه تئوری ECC مدتهاست که مطرح بوده است، تنها در سال‌های اخیر است که محصولات مرتبط با آن به بازار آمده و علاقهٔ زیادی برای نفوذ در این الگوریتم و کشف نقاط ضعف آن ظاهر شده است. بنابراین سطح اطمینان به ECC هنوز به اندازهٔ RSA بالا نیست.

توضیح مبانی ECC مشکل‌تر از RSA و Diffie-Hellman بوده و توصیف کامل ریاضی آن فراتر از حیطهٔ این کتاب است. مبنای روش، استفاده از یک ساختار ریاضی، بنام خَم بیضوی است.

۳-۵ امضاءهای دیجیتال

از رمزنگاری کلید- عمومی می‌توان بصورت دیگری همانند شکل ۷-۳ب استفاده کرد. فرض کنید که Bob بخواهد تا پیامی را برای Alice بفرستد و اگرچه مهم نیست که پیام سرّی بماند ولی اصرار دارد که Alice مطمئن شود که پیام واقعاً از طرف اوست. در این مورد Bob از کلید خصوصی خود برای رمزنگاری پیام استفاده می‌کند. وقتی Alice متن رمزشده را دریافت می‌دارد، او متوجه می‌شود که می‌تواند پیام را با کلید عمومی Bob رمزگشائی کند و بدین ترتیب اثبات می‌شود که پیام بتوسط Bob رمزنگاری شده است. هیچ شخص دیگری کلید خصوصی Bob را ندارد و بنابراین شخص دیگری نمی‌توانسته است متن رمزشده‌ای را خلق کند که با کلید عمومی Bob باز شود. بنابراین کل پیام رمزشده بصورت یک **امضاء دیجیتال** عمل می‌کند. علاوه برآن غیرممکن است که بتوان پیام را بدون دسترسی به کلید خصوصی Bob تغییر داد و بنابراین اعتبار پیام چه ازنظر منبع ارسال و چه از نظر اصالت تأئید می‌گردد.

در روش قبل، تمام پیام رمزنگاری می‌شودکه اگرچه هم نویسنده و هم محتوای پیام تأئید می‌گردد ولی به حجم حافظهٔ زیادی نیاز دارد. هر سند را بایستی بصورت متن ساده نگهداری نمود تا در صورت نیاز به آن مراجعه کرد. یک کپی از متن رمزشده را نیز بایستی حفظ کرد تا در صورت وجود تناقض بتوان، مبدأ و محتوا را با اصل تطبیق داد. راه بهره‌ورتری برای کسب همین نتایج این است که بلوک کوچکی از بیت‌ها که تابعی از پیام است را رمزنگاری کرد. چنین بلوکی که اعتبارسنج خوانده می‌شود بایستی دارای این خاصیت باشد که امکان نداشته باشد که بتوانا سند را تغییر داد ولی اعتبارسنج تغییر نکند. اگر اعتبارسنج با کلید خصوصی فرستنده رمزنگاری شده باشد، بعنوان یک امضاء که مبدأ، محتوا، و نظم را تأئید می‌کند عمل

خواهد کرد. یک کُد hash امن همانند SHA-1 می‌تواند برای این مقصود بکار رود. شکل ۲-۳ب این سناریو را نشان می‌دهد.

مهم است تأکید کنیم که عمل رمزنگاری که هم‌اکنون تشریح گردید، محرمانگی را فراهم نمی‌آورد. یعنی پیامی که ارسال می‌شود را نمی‌توان تغییر داد ولی می‌توان آن را استراق سمع کرد. این امر در مورد امضائی که مبتنی بر بخشی از پیام است روشن است، زیرا بقیهٔ پیام بصورت متن ساده ارسال می‌گردد. ولی حتی در صورت رمزنگاری کامل پیام، بازهم هیچ حفاظتی در برابر محرمانگی وجود ندارد زیرا هر ناظری می‌تواند پیام را با استفاده از کلید عمومی فرستنده رمزگشائی کند.

۳-۶ مدیریت کلید

یکی از نقش‌های عمدهٔ رمزنگاری کلید- عمومی، موضوع توزیع کلید است. در واقع استفاده از رمزنگاری کلید- عمومی برای این مقصود دارای دو جنبه است:

- توزیع کلیدهای عمومی
- استفاده از رمزنگاری کلید- عمومی برای توزیع کلیدهای سرّی

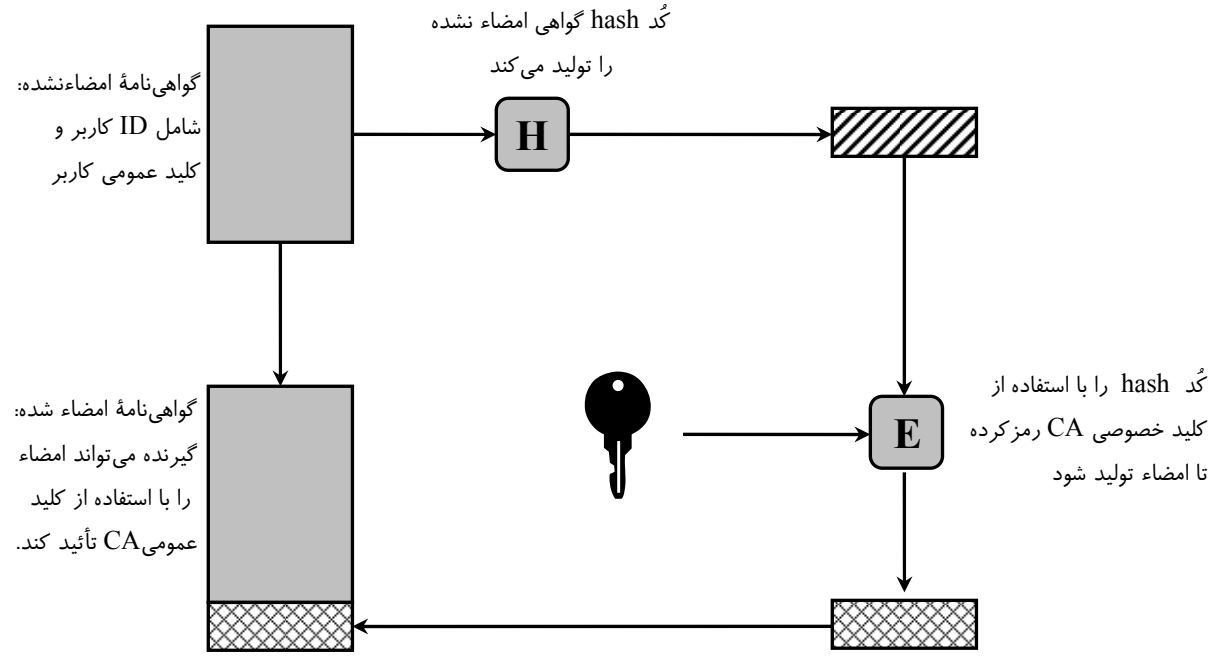
هریک از این دو مقوله را بترتیب بررسی می‌کنیم.

گواهی‌نامه‌های کلید- عمومی

همان‌طور که از اسم آن برمی‌آید، نکتهٔ رمزنگاری کلید- عمومی این است که کلید عمومی، عمومی است. بنابراین با الگوریتم پذیرفته‌شده‌ای همانند RSA، هر شرکت‌کننده می‌تواند کلید عمومی خود را برای هر شرکت‌کنندهٔ دیگری فرستاده و یا آن را برای اطلاع جمعی ارسال نماید. اگرچه این روش مناسب است ولی دارای یک ضعف عمده است. هر شخص دیگری نیز می‌تواند چنین کاری را انجام دهد. یعنی کاربر دیگری می‌تواند تظاهر نماید که کاربر A بوده و یک کلید عمومی را برای کاربر دیگر و یا جمع کاربران ارسال نماید. تا زمانی که کاربر A این تقلب را کشف کرده و به سایر شرکت‌کنندگان اطلاع دهد، فرد متقلب قادر خواهد بود تا تمام پیام‌های رمزنگاری شده به مقصد A را خوانده و از کلیدهای تقلبی برای اعتبارسنجی استفاده کند.

راه حل این مشکل، استفاده از گواهی‌نامهٔ کلید- عمومی است. یک گواهی‌نامه شامل یک کلید عمومی بعلاوه شناسهٔ کاربر(User ID) صاحب کلید است که مجموعهٔ آن بتوسط یک طرف ثالث مورد اعتماد امضاء شده باشد. معمولاً طرف ثالث یک مسئول صدور گواهی‌نامه(Certificate Authority (CA است که همانند یک واحد دولتی و یا یک مؤسسهٔ تجاری، مورد اعتماد جمعیت کاربران است. یک کاربر می‌تواند کلید عمومی خود را با روش امنی به CA ارائه نموده و یک گواهی‌نامه دریافت دارد. کاربر آنگاه می‌تواند این گواهی‌نامه را انتشار دهد. هر کسی که به کلید عمومی کاربر نیاز دارد می‌تواند این گواهی‌نامه را گرفته و اعتبار آن را تأئید کند. شکل ۱۲-۳ این شیوه را نشان می‌دهد.

روشی که برای صدور گواهی‌نامه‌های کلید- عمومی پذیرش جهانی یافته است، استاندارد X.509 نام دارد. از گواهی‌نامه‌های X.509 در بیشتر کاربردهای امنیت شبکه مثل امنیت IP، لایهٔ سوکت امن (SSL)، اسناد امن الکترونیک (SET)، و S/MIME استفاده می‌شود که همهٔ اینها در فصول بعدی کتاب مورد بحث قرار گرفته است. X.509 بصورت کامل در فصل ۴ بررسی شده است.



شکل ۱۲-۳ استفاده از گواهی‌نامهٔ کلید- عمومی

توزیع کلیدهای سرّی از طریق کلید عمومی

در رمزنگاری سنّتی، نیاز اصلی طرفین برای ارتباط امن این است که یک کلید سرّی را در اشتراک داشته باشند. فرض کنید که Bob بخواهد عملی در رابطه با ارسال پیام انجام دهدکه او را قادر سازد تا با هرشخص دیگری که دسترسی به اینترنت، و یا شبکهٔ دیگری که در اشتراک آن دو است، دارد e-mail امن ردوبدل نماید. فرض کنید که Bob بخواهد تا این عمل را با استفاده از رمزنگاری سنّتی انجام دهد. در رمزنگاری سنّتی، Bob و طرف مقابلش مثلاً Alice بایستی روشی را پیدا کنندکه بتوانند یک کلید سرّی را بدون این‌که کسی متوجه شود ردوبدل نمایند. آنها چگونه باید این کار را انجام دهند؟ اگر Alice در اطاق مجاور Bob باشد، Bob می‌تواند کلید را تهیه کرده و آن را روی یک تکه کاغذ نوشته و یا روی یک دیسکت ذخیره نموده و شخصاً به Alice بدهد. اما اگر Alice در سوی دیگر دنیا باشد، چه باید کرد؟ او می‌تواند این کلید را با استفاده از رمزنگاری متقارن به رمز درآورد و آن را از طریق e-mail برای Alice بفرستد، اما این بدین معنی است که Bob و Alice بایستی برای رمزنگاری این کلید سرّی جدید، یک کلید سرّی مشترک داشته باشند. علاوه برآن Bob و هر کس دیگری که از این بستهٔ نرم‌افزاری e-mail جدید استفاده می‌کند برای ارتباط با هر شخص دیگر دچار همین مشکل‌اند. هرجفت مرتبط، بایستی یک کلید سرّی یکتا در اشتراک داشته باشند.

یک راه حل، مبادلهٔ کلید Diffie-Hellman است. در واقع از این روش استفادهٔ زیادی می‌شود. ولی در این روش یک نقطهٔ ضعف وجود دارد و آن این است که در ساده‌ترین حالت خود Diffie-Hellman هیچ نوع اعتبارسنجی برای دو طرف ارتباط ایجاد نمی‌کند.

راه حل قوی دیگر استفاده از گواهی‌نامه‌های کلید- عمومی است. وقتی Bob می‌خواهد تا با Alice ارتباط برقرار کند می‌تواند چنین کند:

۱- پیامی را آماده نماید.
۲- آن پیام را با یک کلید اجلاس یکبارمصرف بصورت متقارن رمزنگاری نماید.
۳- کلید اجلاس را با استفاده از کلید عمومی Alice و از طریق رمزنگاری کلید- عمومی به رمز درآورد.
۴- کلید اجلاس رمزشده را به پیام وصل کرده و آن را برای Alice بفرستد.

تنها Alice قادر به رمزگشائی کلید اجلاس و بنابراین استخراج پیام اولیه است. اگر Bob کلید عمومی Alice را از طریق گواهی‌نامهٔ کلید- عمومی Alice بدست آورد، آنگاه Bob مطمئن خواهد بود که آن کلید، یک کلید معتبر است.

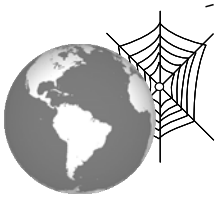
۳-۷ منابع مطالعاتی

بررسی کامل توابع درهم‌ساز و کُدهای اعتبارسنجی پیام را می‌توان در [STIN06] و [MENE97] پیدا کرد.

آنچه که بعنوان منابع مطالعاتی در فصل ۲ معرفی گردید، هم رمزنگاری سنّتی و هم رمزنگاری کلید- عمومی را پوشش می‌دهند. [DIFF88] بصورت مفصّل چندین تلاش انجام شده برای بکارگیری الگوریتم‌های رمزنگاری دو- کلیدی و تکامل تدریجی تعدادی از پروتکل‌های مبتنی بر آنها را بررسی کرده است. [CORM01] خلاصهٔ خواندنی و مفیدی از تمام الگوریتم‌های مرتبط با تأئید، محاسبه و شکستن رمز RSA را فراهم نموده است.

CORM01 Cormen, T.;Leiserson, C.; Rivest, R.; and Stein, C. *Introduction to Algorithms*. Cambridge, MA: MIT Press, 2001.
DIFF88 Diffie, W. "The First Ten Years of Public- Key Cryptography." *Proceedings of the IEEE*, May 1988. Reprinted in [SIMM92].
MENE97 Menezes, A.; Oorshcot, P.; and Vanstone, S. *Handbook of Applied Cryptography*. Boca Raton, FL: CRC Press, 1997.
SIMM92 Simmons, G., ed. *Contemporary Cryptology*: *The Science of Information Integrity*. Piscataway, NJ: IEEE Press, 1992.
STIN06 Stinson, D. *Cryptography: Theory and Practice*. Boca Raton, FL: CRC Press, 2006.

وِب سایت‌های مفید



- NIST Secure Hashing Page: استانداردهای SHA FIPS و اسناد مرتبط.
- Whirlpool: اطلاعات زیادی در مورد Whirlpool.
- RSA Laboratories: مجموعهٔ مفصلی از مطالب فنی در مورد RSA و سایر عناوین رمزنگاری.

۳-۸ واژه‌های کلیدی، سؤالات مرورکنندهٔ بحث و مسائل

واژه‌های کلیدی

کلید خصوصی	private key	مبادلهٔ کلید DH	Diffie-Hellman key exchange
کلید عمومی	public key	امضاء دیجیتال	digital signature
گواهی‌نامهٔ کلید- عمومی	public-key certificate	استاندارد امضاء دیجیتال	Digital Signature Standard(DSS)
رمزنگاری کلید- عمومی	public-key encryption	رمزنگاری خَم بیضوی	elliptic-curve cryptography(ECC)
نوعی الگوریتم درهم‌سازی	RIPEMD-160	نوعی الگوریتم درهم‌سازی	HMAC
مشهورترین الگوریتم رمزنگاری کلید- عمومی	RSA	مبادلهٔ کلید	key exchange
کلید سرّی	secret key	نوعی الگوریتم درهم‌سازی	MD5
تابع درهم‌ساز اَمن	secure hash function	اعتبارسنجی پیام	message authentication
نوعی الگوریتم درهم‌سازی	SHA-1	کُد اعتبارسنجی پیام	message authentication code (MAC)
مقاومت قوی در برابر تصادم	strong collision resistance	چکیدهٔ پیام	message digest
مقاومت ضعیف در برابر تصادم	weak collision resistance	تابع درهم‌ساز یک- طرفه	one-way hash function

سوالات مرورکننده بحث

۳-۱ سه روش برخورد با اعتبارسنجی پیام را نام ببرید.

۳-۲ کُد اعتبارسنجی پیام چیست؟

۳-۳ سه روشی که در شکل ۳-۲ نشان داده شده است را بطور مختصر تشریح کنید.

۳-۴ یک تابع درهم‌ساز چه خواصی داشته باشد تا برای اعتبارسنجی پیام مفید باشد؟

۳-۵ در مقولهٔ یک تابع hash، یک تابع فشرده‌ساز چیست؟

۳-۶ اجزاء اصلی یک سیستم رمزنگاری کلید- عمومی کدامند؟

۳-۷ سه مورد استفاده از یک سیستم رمزنگاری کلید- عمومی را نام برده و توضیح دهید.

۳-۸ اختلاف بین یک کلید خصوصی با یک کلید سرّی در چیست؟

۳-۹ امضاء دیجیتال چیست؟

۳-۱۰ یک گواهی‌نامهٔ کلید- عمومی چیست؟

۳-۱۱ چگونه می‌توان از رمزنگاری کلید- عمومی برای توزیع کلید استفاده کرد؟

مسائل

۳-۱ یکی از پراستفاده‌ترین MACها که الگوریتم اعتبارسنجی دیتا (Data Authentication Algorithm) خوانده می‌شود مبتنی بر DES است. این الگوریتم هم از انتشارات FIPS بوده (FIPS PUB 113)، و هم استاندارد ANSI است (X9.17). می‌توان چنین تعریف کرد که الگوریتم از مُود عملیاتی زنجیره‌ای رمز قالبی(CBC) با بردار اولیهٔ صفر استفاده می‌کند (شکل ۹-۲). دیتائی که قرار است اعتبارسنجی گردد (مثل پیام، رکورد، فایل یا برنامه) به بلوک‌های ۶۴- بیتی مجاور هم P_1، P_2، و P_N تقسیم می‌شود. اگر لازم باشد بلوک نهائی با صفرهائی در سمت راست پر شده تا ۶۴ بیت آن کامل شود. MAC یا شامل بلوک رمزشدهٔ کامل C_N و یا M بیت سمت چپی بلوک است ($16 \leq M \leq 64$). نشان دهید که همین نتیجه را می‌توان با استفاده از مُود فیدبک رمز (CFB) نیز به دست آورد.

۳-۲ یک تابع ۳۲- بیتی hash را بصورت جمع رشته‌ای دو تابع ۱۶- بیتی XOR و RXOR که در بخش ۲-۳ بعنوان « دو تابع سادهٔ hash » تعریف شده است، در نظر بگیرید.

الف- آیا این جمع کنترلی(checksum)، تمام خطاهائی که بعلت تغییر تعداد فردی از بیت‌ها حاصل می‌شود را تشخیص می‌دهد؟ توضیح دهید.

ب- آیا این جمع کنترلی، تمام خطاهائی که بعلت تغییر تعداد زوجی از بیت‌ها حاصل می‌شود را تشخیص می‌دهد؟ اگر اینطور نیست، الگوی خطاهائی که باعث شکست این جمع کنترلی می‌شود را مشخص کنید.

ج- نسبت به مؤثربودن این تابع درهم‌ساز برای استفاده بعنوان یک تابع hash اعتبارسنجی نظر دهید.

۳-۳ فرض کنید که $H(m)$ یک تابع درهم‌ساز مقاوم در برابر تصادم بوده که یک پیام با طول هرچند بیت را به به یک اندازهٔ hash با طول n- بیت نگاشت می‌کند.آیا این درست است که برای تمام پیام‌های x و x' که $x \neq x'$ است، $H(x) \neq H(x')$ است؟ پاسخ خود را تشریح کنید.

۳-۴ **الف**- تابع درهم‌ساز زیر را در نظر بگیرید. پیام‌ها بصورت ردیفی از اعداد دهدهی هستند، $M = (a_1, a_2, \ldots, a_t)$. اندازهٔ hash بصورت $\left(\sum_{i=1}^{t} a_i\right) \bmod n$ برای یک مقدار n که از قبل تعریف شده است محاسبه می‌شود. آیا این تابع درهم‌ساز هیچیک از لازمه‌های یک تابع درهم‌ساز که در بخش ۲-۳ لیست شده است را ارضاء می‌کند؟ پاسخ خود را توضیح دهید.

ب- قسمت (الف) را برای تابع $h = \left(\sum_{i=1}^{t} (a_i)^2\right) \bmod n$ تکرار کنید.

ج- تابع hash قسمت (ب) را برای $M = (189,632,900,722,349)$ و $n = 989$ محاسبه کنید.

۳-۵ این مسأله یک تابع درهم‌ساز شبیه به SHA را معرفی می‌کند که بجای عمل روی دیتای باینری بر روی حروف عمل می‌کند. این تابع بنام *toy tetragraph hash* (tth) نامیده می‌شود. اگر پیامی که شامل ردیفی از حروف است را داشته باشیم، tth یک اندازهٔ hash که شامل ۴ حرف است را تولید می‌کند. در ابتدا tth پیام را با صرفنظرکردن جاهای خالی بین کلمات، علائم و حروف بزرگ، بصورت بلوک‌های ۱۶- حرفی درمی‌آورد. اگر طول پیام بر ۱۶ قابل قسمت نباشد، لائی null به اندازهٔ لازم به انتهای آن اضافه می‌شود. یک دنباله چهارتائی عددی که با اندازهٔ (0,0,0,0) آغاز می‌شود پیوسته نگهداری می‌گردد. این دنباله برای پردازش اولین بلوک، در ورودی یک تابع فشرده‌ساز قرار می‌گیرد. تابع فشرده‌ساز شامل دو مرحله است. **مرحلهٔ ۱** : بلوک بعدی متن را گرفته و آن را بصورت یک بلوک ۴ × ۴ ردیفی درآورده و به عدد تبدیل کنید (A = 0, B = 1 و غیره). مثلاً برای بلوک ABCDEFGHIJKLMNOP خواهیم داشت:

A	B	C	D
E	F	G	H
I	J	K	L
M	N	O	P

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

آنگاه هر ستون را بصورت mod 26 جمع کرده و نتیجه را با دنبالهٔ عددی چهارتائی باز هم بصورت mod 26 جمع کنید. در این مثال دنبالهٔ چهارتائی جدید بدست آمده برابر (24,2,6,10) می‌شود. **مرحلهٔ ۲** : از ماتریس مرحلهٔ ۱ استفاده کرده و اولین ردیف را ۱ خانه به چپ، دومین ردیف را ۲ خانه به چپ ، سومین ردیف را ۳ خانه به چپ چرخانده و ترتیب ردیف آخر را معکوس کنید. در مورد مثال ما:

B	C	D	A
G	H	E	F
L	I	J	K
P	O	N	M

1	2	3	0
6	7	4	5
11	8	9	10
15	14	13	12

حال، هر ستون را بصورت mod 26 جمع کرده و نتیجه را نیز بهمین صورت به دنبالهٔ چهارتائی مرحلهٔ قبل اضافه کنید. دنبالهٔ چهارتائی جدید (5,7,9,11) است. این دنباله حالا برای پردازش بلوک بعدی متن در ورودی تابع فشرده‌سازی مرحلهٔ ۱ قرار خواهد گرفت. پس از اینکه آخرین بلوک پیام پردازش گردید، دنبالهٔ چهارتائی نهائی را به حروف تبدیل کنید. برای مثال اگر پیام ABCDEFGHIJKLMNOP بوده است، اندازهٔ hash مقدار FHJL خواهد شد.

الف- شکل‌هائی قابل مقایسه با شکل‌های ۴-۳ و ۵-۳ رسم کنید تا منطق کلی tth و منطق تابع فشرده‌سازی را نشان دهد.

ب- تابع hash پیام ۴۸- حرفی "I leave twenty million dollars to my friendly cousin Bill" را محاسبه کنید.

ج- برای اینکه ضعف tth آشکار شود، یک بلوک ۴۸- حرفی دیگر که همان اندازهٔ hash قسمت (ب) را ایجاد کند پیدا کنید. *راهنمائی*: از حرف A زیاد استفاده کنید.

۳-۶ این امکان وجود دارد که از یک تابع درهم‌ساز برای ساخت یک رمز قالبی با ساختاری مشابه DES استفاده کرد. با توجه به این‌که یک تابع درهم‌ساز، یک- طرفه بوده ولی یک رمز قالبی بایستی برگشت‌پذیر باشد (برای رمزگشائی)، چگونه این امر ممکن است؟

۳-۷ قبل از کشف روش‌های رمزنگاری کلید- عمومی، مثل RSA، اثبات شده بود که رمزنگاری کلید- عمومی در تئوری می‌تواند وجود داشته باشد. توابع $f_1(x_1) = z_1$، $f_2(x_2,y_2) = z_2$ و $f_3(x_3,y_3) = z_3$ که در آنها تمام مقادیر اعداد صحیح و $1 \le x_i, y_i, z_i \le N$ می‌باشند را در نظر بگیرید. تابع f_1 را می‌توان با بردار M1 با طول N که درآن k امین عنصر اندازهٔ $f_1(k)$ است، نشان داد. بطریق مشابه، f_2 و f_3 را می‌توان با ماتریس‌های M2 و M3 که ماتریس‌های $N \times N$ هستند، نمایش داد. هدف این است که عمل رمزنگاری/ رمزگشائی را با مراجعه به جداولی که این جداول دارای اندازهٔ بسیار بزرگ N هستند نشان داد. این جداول بطور غیرعملی بسیار بزرگ بوده ولی از نظر تئوری می‌توانند ساخته شوند. روش چنین است: M1 را با جایگشت تصادفی تمام اعداد صحیح بین 1 و N بسازید. یعنی هر عدد صحیح فقط تنها یکبار در M1 وارد شود. M2 را چنان بسازید که هر ردیف شامل جایگشت تصادفی N عدد صحیح قبلی باشد. بالاخره M3 را چنان بسازید که شرط زیر را ارضاء کند:

برای تمام مقادیر k و p با $1 \le k, p \le N$ داشته باشیم: $f3(f2(f1(k),p),k) = p$

عبارت بالا با کلمات چنین بیان می‌شود:

۱- M1 یک ورودی k را گرفته و خروجی x را تولید می‌کند.

۲- M2 ورودی‌های x و p را گرفته و خروجی z را تولید می‌کند.

۳- M3 ورودی‌های z و k را گرفته و خروجی p را تولید می‌کند.

وقتی سه جدول ساخته شدند، انتشار می‌یابند.

الف- بایستی روشن باشد که می‌توان M3، بطوری که شرایط قبل را ارضاء کند، تشکیل داد. بعنوان مثال M3 را در حالت سادهٔ زیر بسازید:

M1 =

5
4
2
3
1

M2 =

5	2	3	4	1
4	2	5	1	3
1	3	2	4	5
3	1	4	2	5
2	5	3	4	1

M3 =

قرارداد: عنصر i ام M1 متناظر با $k = i$ است. ردیف i ام M2 متناظر با $x = i$ است و ستون j ام M2 متناظر با $p = j$ است. ردیف i ام M3 متناظر با $z = i$, و ستون j ام M3 متناظر با $k = j$ است.

ب- استفاده از این مجموعهٔ جداول برای انجام عمل رمزنگاری و رمزگشائی بین دو کاربر را توضیح دهید.

ج- استدلال نمائید که این یک روش امن است.

۳-۸ با استفاده از الگوریتم RSA، رمزنگاری و رمزگشائی، همانند شکل ۹-۳، برای مقادیر زیر را انجام دهید:

الف- $p = 3; \quad q = 11, e = 7 ; M = 5$

ب- $p = 5; \quad q = 11, e = 3 ; M = 9$

ج- $p = 7; \quad q = 11, e = 17 ; M = 8$

د- $p = 11; \quad q = 13, e = 11 ; M = 7$

هـ- $p = 17; \quad q = 31, e = 7 ; M = 2$

راهنمائی: رمزگشائی آنچنان که تصور می‌شود سخت نیست.کمی زیرکی بکار برید.

۳-۹ در یک سیستم کلید- عمومی که از RSA استفاده می‌کند، متن رمزشده $C = 10$ را که برای کاربری با کلید عمومی $n = 35$ و $e = 5$ ارسال شده است استراق سمع می‌کنید. متن سادهٔ M چیست؟

۳-۱۰ در یک سیستم RSA، کلید عمومی یک کاربر $n = 3599$ و $e = 31$ است. کلید خصوصی این کاربر چیست؟

۳-۱۱ فرض کنید که یک سری بلوک‌هائی در دسترس‌اند که با الگوریتم RSA کُد شده‌اند ولی کلید خصوصی را در اختیار نداریم. فرض کنید $n = pq$ و e کلید عمومی است. همچنین فرض کنید که شخصی بما اطلاع می‌دهد که یکی از بلوک‌های متن ساده دارای فاکتور مشترکی با n است. آیا این امر بهرترتیب کمکی به ما می‌کند؟

۳-۱۲ نشان دهید که چگونه RSA می‌تواند با ماتریس‌های M1،M2، و M3 مسألهٔ ۷-۳ نشان داده شود.

۳-۱۳ روش زیر را در نظر بگیرید:

۱- عدد فرد E را انتخاب کنید.

۲- دو عدد اول P و Q را طوری انتخاب کنید که $(P\text{-}1)(Q\text{-}1)\text{-}1$ بطور مساوی قابل تقسیم به E باشد.

۳- P و Q را ضرب کنید تا N بدست آید.

۴- $D = [(P\text{-}1)(Q\text{-}1)(E\text{-}1) + 1] / E$ را حساب کنید.

آیا این روش معادل RSA است؟ پاسخ خود را توجیه کنید.

۳-۱۴ استفاده از RSA با یک کلید شناخته شده را برای ساخت یک تابع درهم‌ساز یک- طرفه در نظر بگیرید. آنگاه یک پیام که شامل دنباله‌ای از بلوک‌هاست را بصورت زیر پردازش کنید: بلوک اول را رمزنگاری نمائید. نتیجه را با بلوک دوم XOR نموده و مجدداً رمزنگاری کنید و بهمین ترتیب ادامه دهید. با حل مسألهٔ زیر نشان دهید که این روش امن نیست: اگر یک پیام شامل دو بلوک B1 و B2 بوده و کُد hash آن چنین باشد

$$RSAH(B1,B2) = RSA(RSA(B1) \oplus B2)$$

اگر بلوک اختیاری C1 داده شده باشد، C2 را چنان اختیار کنید که $RSAH(C1, C2) = RSAH(B1, B2)$. بنابراین تابع درهم‌ساز شرط مقاومت ضعیف در برابر تصادم را ارضاء نمی‌کند.

۳-۱۵ فرض کنید که Bob از سیستم رمزنگاری RSA با یک مدول بسیار بزرگ n استفاده می‌کند که فاکتور کردن آن در زمان معقول قابل تصور نیست. فرض کنید که Alice پیامی را برای Bob می‌فرستد که در آن هر یک از حروف الفباء با یک عدد بین صفر و ۲۵ نمایش داده شده و سپس هر عدد بطور مجزا با الگوریتم RSA با e بزرگ و n بزرگ رمزنگاری شده است. آیا این روش امن است؟ اگر جواب منفی است، بهره‌ورترین روش حمله بر ضد این نوع رمزنگاری چیست؟

۳-۱۶ یک روش Diffie-Hellman با یک عدد اول $q = 11$ و ریشهٔ اولیهٔ آن $\alpha = 2$ را درنظر بگیرید.

الف- اگر کاربر A دارای کلید عمومی $Y_A = 9$ باشد، کلید خصوصی X_A مربوط به این کاربر چیست؟

ب- اگر کاربر B دارای کلید عمومی $Y_B = 3$ باشد، کلید مشترک سرّی K چیست؟

قسمت دوم

کاربردهای امنیت شبکه

در قسمت اول، رمزهای مختلف را بررسی کرده و به استفاده از آنها برای محرمانگی، اعتبارسنجی، مبادلهٔ کلید و وظایف مرتبط با این کاربردها اشاره نمودیم. قسمت دوم ابزارهای مهم امنیت شبکه و کاربردهائی که از این ابزارها استفاده می‌کنند را مورد بررسی قرار می‌دهد. این ابزارها را می‌توان در یک شبکهٔ منفرد، اینترانت یک سازمان، و یا اینترنت بکاربرد.

فصل ۴ کاربردهای اعتبارسنجی

فصل ۴ به بررسی دو عنوان از مهم‌ترین مشخصه‌های اعتبارسنجی زمان حاضراختصاص دارد. Kerberos یک پروتکل اعتبارسنجی مبتنی بر رمزنگاری متقارن بوده که حمایت و کاربرد گسترده‌ای در سیستم‌های متنوع دارد. X.509 یک الگوریتم اعتبارسنجی را تعیین کرده و یک تسهیلات گواهی‌نامه‌ای را فراهم می‌سازد. این تسهیلات کاربران را قادر می‌سازد تا گواهی‌نامه‌های کلید- عمومی را طوری به دست آوردندکه یک جمعیت از کاربران به اعتبار کلیدهای عمومی اعتماد داشته باشند. این تسهیلات زیربنای برخی از کاربردهاست.

فصل ۵ امنیت پست الکترونیک

پست الکترونیک پراستفاده‌ترین کاربرد توزیع شده بوده و تمایل فزاینده‌ای در مورد فراهم آوردن سرویس‌های اعتبارسنجی و محرمانگی بعنوان بخشی از تسهیلات پست الکترونیک بوجود آمده است. فصل ۵ به دو روش که احتمالاً در بخش امنیت پست الکترونیک حاکمیت خواهند یافت نگاه می‌کند. Pretty Good Privacy(PGP) یک روش پراستفاده است که متکی بر هیچ مقام مسئول و یا سازمانی نیست. در نتیجه این روش همان قدر که برای پیکربندی شبکه‌هائی که بتوسط سازمان‌ها اداره می‌شوند کارآئی دارد، در مورد مصارف فردی نیز دارای استفاده است. (Secure/Mutipurpose Internet Mail Extensions) S/MIME صرفاً بعنوان یک استاندارد اینترنت طراحی شده است.

فصل ۶ امنیت IP

پروتکل اینترنت (IP) عنصر مرکزی اینترنت و اینترانت‌های خصوصی است. در نتیجه امنیت سطح IP برای طراحی هر روش امنیتی مبتنی بر عملیات بین‌شبکه‌ای مهم است. فصل ۶ نگاهی به روش امنیت IP انداخته که به منظور کار با IP جاری و IP نسل بعد که IPv6 خوانده می‌شود طراحی شده است.

فصل ۷ امنیت WEB

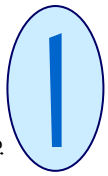
رشد انفجارگونهٔ استفاده از تارجهان‌گستر برای تجارت الکترونیک و انتشار همه جانبهٔ اطلاعات باعث شده است تا نیاز مبرمی برای استقرار یک امنیت قوی مبتنی بر وِب بوجود آید. فصل ۷ این مورد جدید امنیتی و مهم را مورد بررسی قرار داده و به دو استاندارد کلیدی یعنی لایهٔ سوکت امن (SSL) و اسناد الکترونیکی امن (SET) نظر می‌کند.

فصل ۸ امنیت مدیریت شبکه

با استفادهٔ روزافزون از سیستم‌های مدیریت شبکه برای کنترل شبکه‌های متنوع، نیاز فزاینده‌ای به استقرار قابلیت‌های امنیت در این سیستم‌ها بوجود آمده است. فصل ۸ بر پراستفاده‌ترین روش مدیریت شبکه، یعنی پروتکل سادهٔ مدیریت شبکه (SNMP) متمرکز است. نسخهٔ اول SNMP تنها یک تسهیلات اعتبارسنجی ابتدائی مبتنی بر کلمهٔ عبور دارد. SNMPv2 قابلیت‌های فراوان‌تری را بوجود آورده و SNMPv3 یک تسهیلات امنیتی فراگیر برای محرمانگی و اعتبارسنجی ایجاد می‌کند که می‌تواند به همراه با SNMPv1 و SNMPv2 بکار رود.

فصل ۴

کاربردهای اعتبارسنجی



ین فصل برخی از عملیات اعتبارسنجی که برای پشتیبانی از اعتبارسنجی سطح کاربرد و امضاء دیجیتال طراحی شده است را بررسی می‌کند.

بحث را با نگاهی به یکی از ابتدائی‌ترین سرویس‌ها، که پراستفاده‌ترین آنها نیز بوده است و Kerberos خوانده می‌شود، آغاز می‌کنیم. سپس سرویس اعتبارسنجی فهرست راهنمای X.509 را مورد مطالعه قرار می‌دهیم. این استاندارد بعنوان بخشی از سرویس فهرست راهنما که این استاندارد حامی آن است دارای اهمیت بوده ولی مهم‌تر این‌که بعنوان خشت اصلی مورد استفاده در سایر استانداردها، همانند S/MIME، که در فصل ۵ از آن یاد خواهد شد نیز دارای کاربرد است. در پایان مفهوم زیرساخت کلید- عمومی (PKI) را بررسی می‌کنیم.

۴-۱ KERBEROS

Kerberos یک سرویس اعتبارسنجی است که بعنوان بخشی از پروژهٔ آتنه (Athena) در دانشگاه MIT طراحی شده است. مشکلی را که Kerberos مورد توجه قرار می‌دهد چنین است: یک محیط گستردهٔ باز را در نظر بگیرید که در آن کاربرانی که در ایستگاه‌های مختلف کاری حضور دارند، علاقه‌مند به دست‌یابی به سرویس‌های مختلفی که روی سِرورهای متعدد کل شبکه قرار دارند می‌باشند. ما تمایل داریم که سِرورها بتوانند دست‌یابی‌های مربوط به کاربران معتبر را بمیزان دلخواه محدود نموده، و همچنین قادر باشند اعتبار کاربر متقاضی سرویس را بسنجند. در چنین محیطی، نمی‌توان به یک ایستگاه کاری از نظر شناسایی صحیح کاربران خودش برای دست‌یابی به سرویس‌های شبکه اعتماد کرد. علی‌الخصوص سه تهدید زیر همیشه وجود دارند:

- یک کاربر ممکن است به یک ایستگاه کاری بخصوص دسترسی یافته و چنین وانمود کند که کاربر دیگری است که از آن ایستگاه تماس گرفته است.
- یک کاربر ممکن است آدرس شبکهٔ یک ایستگاه کاری را طوری تغییر دهد که باعث شود تقاضاهایی که از این ایستگاه ارسال می‌شوند، بر حسب ظاهر مربوط به ایستگاه دیگری تلقی گردند.
- یک کاربر ممکن است با عمل شنود روی یک خط، حمله‌ای از نوع بازخوانی (replay) انجام داده، وارد سِرور شده و یا عملیات را مختل سازد.

در هریک از این موارد، یک کاربر غیرمعتبر ممکن است به سرویس‌ها و داده‌هایی دست یابد که مجاز به دست‌یابی به آنها نیست. بجای قراردادن پروتکل‌های اعتبارسنجی دشوار در هر سِرور، Kerberos یک سِرور اعتبارسنجی متمرکز که وظیفهٔ آن معرفی کاربران به سِرورها، و سِرورها به کاربران است را فراهم می‌سازد. برخلاف اغلب روش‌های اعتبارسنجی معرفی شده در این کتاب، Kerberos منحصراً بر رمزنگاری متقارن متکی بوده و از رمزنگاری کلید- عمومی استفاده نمی‌کند.

دو نسخه از Kerberos دارای استفادهٔ وسیع‌اند. نسخهٔ ۴ [MILL88,STEI88] هنوز بطور گسترده‌ای مورد استفاده است. نسخهٔ ۵ [KOHL94] بعضی از کمبودهای امنیتی نسخهٔ ۴ را جبران کرده و بعنوان یک استاندارد اینترنت (RFC 1510) پیشنهاد شده است.

این بخش را با بحث مختصری در زمینهٔ انگیزش‌های مربوط به روش Kerberos آغاز می‌کنیم. آنگاه نظر به پیچیدگی Kerberos، موضوع را با بررسی پروتکل اعتبارسنجی بکار رفته درنسخهٔ ۴ شروع می‌کنیم. این موضوع ما را قادر می‌سازد تا جوهر استراتژی Kerberos، بدون نیاز به دانستن جزئیات لازم مربوط به حملات امنیتی هوشمندانه را ملاحظه نمائیم. در نهایت، نسخهٔ ۵ را بررسی خواهیم کرد.

انگیزش

اگر مجموعه‌ای از کاربران دارای رایانه‌های شخصی مخصوص به خود که به هیچ شبکه‌ای متصل نیستند وجود داشته باشند، آنگاه منابع و فایل‌های یک کاربر را می‌توان از طریق حفاظت فیزیکی رایانهٔ او حفاظت کرد. اما وقتی این کاربران از طریق یک سیستم مرکزی با اشتراک زمانی بهم متصل می‌شوند، سیستم عامل اشتراک زمانی مسئول حفاظت مجموعه خواهد بود. سیستم عامل می‌تواند خط‌مشی‌های مربوط به کنترل دست‌یابی را، بر مبنای هویت کاربر، اعمال کرده و روش‌هائی را برای شناسائی کاربران و احراز هویت آنها در زمان اتصال به سیستم به اجرا بگذارد.

امروزه هیچکدام از این سناریوها معمول نمی‌باشند. معمول‌تر این‌است که یک معماری توزیع شده که شامل ایستگاه‌های کاری تخصیص‌یافته برای کاربران (کلاینت‌ها) و سِرورهای توزیع‌شده و یا متمرکز است، وجود داشته باشد. در چنین محیطی، سه برخورد متفاوت با مسألهٔ امنیت را می‌توان تصور کرد:

۱- اتکاء به ایستگاه‌های کاری کلاینت برای شناسائی کاربر یا کاربرانی که میخواهند به آن وصل شوند و اتکاء به هر سِرور برای اجرای خط‌مشی‌های امنیتی بر مبنای شناسائی کاربر(ID).
۲- الزام سیستم‌های کلاینت به معرفی خود در هنگام اتصال به سِرور و اتکاء به سیستم کلاینت برای شناسائی کاربری که می‌خواهد به آن وصل شود.
۳- الزام کاربر به اثبات هویت خود برای هر سرویس درخواستی و همچنین الزام سِرورها به اثبات هویت خود برای کلاینت‌ها.

در یک محیط کوچک بسته که در آن تمام سیستم‌ها متعلق به یک سازمان منفرد بوده و بتوسط همان سازمان اداره می‌شوند، استراتژی‌های اول و یا شاید دوم کفایت می‌کنند. ولی در یک محیط بازتر که در آن از اتصالات شبکه‌ای برای ارتباط ماشین‌ها با یکدیگر استفاده می‌شود، روش سوم برای حفاظت اطلاعات کاربر و منابع مستقر در سِرورها مناسب‌تر است. روش سوم همانست که Kerberos از آن پشتیبانی می‌کند. Kerberos بر مبنای یک معماری کلاینت/ سِرور عمل کرده و از یک یا چند سِرور Kerberos برای فراهم‌آوردن سرویس اعتبارسنجی، یا تشخیص هویت، استفاده می‌نماید.

اولین گزارش منتشرشده در موردKerberos [STEI88]، الزامات زیر را برای Kerberos بیان نموده است:

- **امن**: یک عامل شنود در شبکه نبایستی بتواند اطلاعات لازم برای جعل هویت یک کاربر را بدست آورد. به عبارت کلی‌تر، Kerberos بایستی آنقدر مستحکم باشد که یک دشمن قوی او را ضعیف نشمارد.
- **قابل اعتماد**: برای تمام سرویس‌هائی که برای کنترل دست‌یابی به Kerberos متکی هستند، عدم دسترسی به سرویس Kerberos به مفهوم عدم دسترسی به همه آنهاست. بنابراین Kerberos باید دارای قابلیت اعتماد بالا بوده و بایستی از یک معماری توزیع‌شده استفاده کند تا در صورت وجود مشکل، یک سیستم بتواند پشتیبان سیستم دیگر گردد.
- **شفاف**: در حالت ایده‌ال، کاربر نبایستی بجز وارد کردن کلمهٔ عبور متوجه شود که عملیات اعتبارسنجی صورت می‌پذیرد.
- **مقیاس‌پذیر**: سیستم بایستی قادر به حمایت از تعداد زیادی کلاینت و سِرور باشد. این نیاز، یک معماری توزیع‌شده و پودمانی را پیشنهاد می‌کند.

برای برآورده نمودن این نیازها، شِگرد Kerberos همان استفاده از یک سرویس اعتبارسنجی قابل اعتماد شخص ثالث است که بر مبنای پروتکلی که بتوسط Needham و Schroeder [NEED78] پیشنهاد شده است، قرار دارد. این سرویس از آن‌جهت بایستی قابل اعتماد باشد که کلاینت‌ها و سِرورها برای اعتبارسنجی و تشخیص هویت یکدیگر به میانداری Kerberos تکیه می‌کنند. با فرض اینکه پروتکل Kerberos خوب طراحی شده باشد، آنگاه سرویس اعتبارسنجی در صورتی امن است که خود سِرور Kerberos امن باشد.

نسخهٔ چهارم Kerberos

نسخهٔ چهارم Kerberos از DES که پروتکل نسبتاً پیچیده‌ای است استفاده کرده تا سرویس اعتبارسنجی را فراهم نماید. با نگاهی کلی به پروتکل، فهم نیاز به آن همه جزئیاتی که در آن منظور شده است کار آسانی نیست. بنابراین ما با استفاده از استراتژی بکار گرفته شده بتوسط Bill Bryant در پروژه آتِنه[BRYA88]، سعی می‌کنیم تا ابتدا با نگاهی به چند دیالوگ فرضی، پروتکل کامل را بنا نمائیم. هر دیالوگ جدید، برای غلبه کردن بر نقاط آسیب‌پذیر امنیتی دیالوگ قبلی، پیچیدگی‌های جدیدی را در پروتکل ایجاد می‌کند.

پس از بررسی پروتکل، به سایر جنبه‌های نسخهٔ ۴ نیز نگاهی می‌اندازیم.

یک دیالوگ سادهٔ اعتبارسنجی

در محیط حفاظت‌نشدهٔ یک شبکه، هر کلاینت میتواند برای دریافت سرویس به هر سِروری مراجعه نماید. در این حالت ریسک امنیتی آشکار، جعل هویت است. یک دشمن می‌تواند خود را بجای کلاینت دیگری جازده و امتیازات غیرقانونی از سِرورها کسب نماید. برای مقابله با این تهدید، سِرورها بایستی بتوانند هویت کلاینت‌هائی که درخواست سرویس دارند را تأیید نمایند. هر سِرور را می‌توان مجبور کرد تا این وظیفه را برای هربار مبادلهٔ کلاینت/ سِرور انجام دهد ولی در یک محیط باز، این درخواست بار سنگینی را به دوش هر سِرور قرار می‌دهد.

راه دیگر این‌است که از یک سِرور اعتبارسنج Authentication Server (AS) استفاده کرد که کلمات عبور تمام کاربران را دانسته و آنها را در یک پایگاه متمرکز داده ذخیره نماید. علاوه براین، AS با هر سِرور دیگر یک کلید سرّی یکتا را به اشتراک می‌گذارد. این کلیدها بصورت فیزیکی و یا بصورت امن دیگری توزیع شده‌اند. به دیالوگ فرضی زیر توجه کنید:

(۱) C→AS: $ID_C \| P_C \| ID_V$

(۲) AS→C: $Ticket$

(۳) C→V: $ID_C \| Ticket$

$Ticket = \mathrm{E}(K_V, [\, ID_C \| AD_C \| ID_V \,])$

که در آن

C	=	کلاینت
AS	=	سِرور اعتبارسنج
V	=	سِرور
ID_C	=	شناسهٔ کاربر روی C
ID_V	=	شناسهٔ V
P_C	=	کلمهٔ عبور کاربر روی C
AD_C	=	آدرس شبکه C
K_V	=	کلید سرّی رمزنگاری مشترک بین AS و V
||	=	جمع رشته‌ای

در این سناریو،کاربر به یک ایستگاه کاری متصل شده و درخواست دسترسی به سِرور V را می‌نماید. مدول کلاینت در ایستگاه کاری، از کاربر درخواست کلمهٔ عبور نموده و سپس پیامی را به AS میفرستدکه شامل ID کاربر، ID سِرور و کلمهٔعبور کاربر است. AS در پایگاه دادهٔ خود جستجو کرده تا ببیند آیا کاربر کلمهٔعبور صحیح برای ID خود را عرضه کرده است و آیا دست‌یابی این کاربر به سِرور V مجاز می‌باشد. اگر هر دو جواب مثبت باشد، AS کاربر را به عنوان یک کاربر معتبر شناخته و حال بایستی سِرور را متقاعد سازد که این کاربر معتبر است. برای این کار، AS بلیتی (ticket) را آماده می‌سازد که شامل ID کاربر، آدرس شبکه و ID سِرور است. این بلیت بتوسط کلید رمزی که بین AS و سِرور مشترک است، رمزنگاری می‌شود. سپس این بلیت برای C بازگردانده می‌شود. چون بلیت رمزنگاری شده است، نه می‌تواند بتوسط C و نه بتوسط یک دشمن تغییر یابد.

با این بلیت، حالا C می‌تواند برای سرویس به V مراجعه کند. C پیامی را برای V می‌فرستد که شامل ID خود C و بلیت است.V بلیت را رمزگشائی کرده و تأئید می‌نماید که ID کاربر که در بلیت وجود دارد مشابه ID رمزنشده موجود در پیام است. اگر این دو با هم تطبیق نمایند، سِرور فرض را بر این می‌گذارد که کاربر دارای هویت معتبر بوده و سرویس درخواستی را در اختیار او قرار می‌دهد.

هریک از مؤلفه‌های پیام شمارهٔ (۳) دارای اهمیت ویژه‌ای است. بلیت برای جلوگیری از تغییر و یا جعل، به رمز در می‌آید. ID سِرور (ID_V) در بلیت جای میگیرد تا سِرور بتواند تأئید کند که رمزگشائی بلیت صحیح انجام شده است. ID_C در بلیت جای دارد تا نشان دهد که این بلیت بخاطر C صادر شده است. بالاخره AD_C بمنظور مقابله با تهدید زیر مورد

استفاده قرار گرفته است. یک دشمن ممکن است بلیت ارسال شده بهمراه پیام (۲) را تصرف کرده، از نام ID_C استفاده کرده و یک پیام بشکل (۳) را از ایستگاه کاری دیگری ارسال کند. در این صورت سِرور یک بلیت معتبر که با ID کاربر تطبیق دارد را دریافت کرده و دست‌یابی را به کاربر، ولی روی ایستگاه کاری دیگر، اِعطا می‌کند. برای جلوگیری از این حمله، AS آدرس شبکه‌ای را که تقاضای اولیه از آن صادر شده بود در بلیت قرار می‌دهد. حال بلیت تنها وقتی معتبر است که از همان ایستگاه کاری ارسال شود که در بدو امر تقاضای بلیت کرده بود.

یک دیالوگ اعتبارسنجی امن‌تر

اگرچه سناریوی قبل تعدادی از مشکلات اعتبارسنجی در یک محیط شبکه‌ای باز را حل می‌کند ولی بازهم مشکلاتی باقی‌است که دوتای آنها علی‌الخصوص قابل توجه است. اولاً علاقه‌مندیم که تعداد دفعاتی که یک کاربر مجبور به وارد نمودن کلمهٔ عبور خود است را به حداقل برسانیم. فرض کنید که از هر بلیت صادر شده تنها یکبار بتوان استفاده کرد. اگر کاربرC در صبح یک روز کاری به ایستگاه متصل شده و بخواهد نامه‌های خود را در یک سِرور پستی مشاهده نماید، C بایستی کلمهٔ عبور خود را عرضه کرده تا یک بلیت برای سِرور پستی به او داده شود. اگر C بخواهد در طول روز چندین مرتبه نامه‌های خود را کنترل کند، برای هربار تلاش نیاز به عرضه نمودن مجدد کلمهٔ عبور دارد. می‌توان وضعیت را بهبود بخشید اگر بتوان کاری کرد که یک بلیت برای دفعات دیگر نیز قابل استفاده باشد. برای یک بار اتصال و گفتگو، ایستگاه کاری می‌تواند بلیت سِرور پستی را پس از دریافت ذخیره کرده و آن را برای دسترسی به سِرور پستی به دفعات از سوی کاربر مورد استفاده قرار دهد.

ولی در تحت این شرایط، بازهم کاربر برای درخواست هر سرویس جدید نیاز به یک بلیت جدید خواهد داشت. اگر کاربر بخواهد به یک سِرور چاپگر، یک سِرور پستی و یک سِرور فایل دسترسی یابد، برای اولین دسترسی به هریک از اینها نیاز به یک بلیت جدید داشته که در نتیجه برای کسب هر بلیت بایستی کلمهٔ عبور خود را ارائه دهد.

مشکل دوم این‌است که در سناریوی قبل، کلمهٔ عبور به صورت یک متن ساده و رمزنشده انتقال می‌یافت [پیام(۱)]. یک استراق‌سمع‌کننده می‌تواند کلمه عبور را گرفته و از هر سرویسی که قربانی، مجاز به دسترسی به آن بوده است استفاده نماید.

برای حل این مشکلات اضافی، روشی برای جلوگیری از انتقال کلمه عبور بصورت متن ساده، و همچنین یک سِرور جدید بنام سِرور اعطاکنندهٔ بلیت Ticket-Granting Server (TGS) را معرفی می‌کنیم. سناریوی جدید که بازهم فرضی است بقرار زیر است:

یک‌بار برای هر اتصال کاربر به سیستم

(۱) C→AS: $ID_C \parallel ID_{tgs}$

(۲) S→C: $E(K_C, Ticket_{tgs})$

یک‌باربرای تقاضای هریک از انواع سرویس

(۳) C→TGS: $ID_C \parallel ID_V \parallel Ticket_{tgs}$

(۴) TGS→C: $Ticket_V$

یک‌بار برای هر اجلاس استفاده از سرویس

$$(۵)\quad C \rightarrow V: \qquad ID_c \parallel Ticket_v$$

$$Ticket_{tgs} = E\,(K_{tgs}\,,\,[\,ID_c \parallel AD_c \parallel ID_{tgs} \parallel TS_1 \parallel Lifetime_1\,])$$

$$Ticket_v \quad = E\,(K_v\,,\,[\,ID_c \parallel AD_c \parallel ID_v \parallel TS_2 \parallel Lifetime_2\,])$$

سرویس جدید، TGS، بلیت‌هائی را برای کاربرانی که اعتبار آنها بتوسط AS تأئید شده است صادر می‌کند. بنابراین کاربر ابتدا از AS تقاضای یک بلیت اعطاکنندهٔ بلیت ($Ticket_{tgs}$) می‌نماید. این بلیت بتوسط مدول کلاینت در ایستگاه کاری کاربر ذخیره می‌شود. هربار که کاربر نیازمند دست‌یابی به سرویس جدیدی است، کلاینت آن را به TGS ارائه داده و با استفاده از بلیت ذخیره شده هویت خود را به اثبات می‌رساند. آنگاه TGS یک بلیت برای آن سرویس خاص صادر می‌کند. کلاینت هر بلیت اعطاکنندهٔ سرویس را ذخیره کرده و از آن برای ارائهٔ اعتبار کاربر خود به سِرور، در هر بار تقاضا برای سرویس خاصی، استفاده می‌کند. اجازه دهید به جزئیات امر نگاهی بیندازیم:

۱- کلاینت به نمایندگی کاربر، درخواست یک بلیت اعطاکنندهٔ بلیت را کرده و برای این امرID خود و TGS ID را برای AS می‌فرستد که بیانگر درخواست استفاده از سرویس TGS است.

۲- AS با ارسال یک بلیت،که با کلید K_c رمزنگاری شده است پاسخ می‌دهد. این کلید از کلمهٔ عبور کاربر که قبلاً در AS ذخیره شده است تهیه می‌شود. وقتی این پاسخ وارد کلاینت می‌شود، کلاینت با ارسال پیامی از کاربر تقاضای کلمهٔ عبور کرده و سپس با استفاده از آن کلید را تولید کرده و برای رمزگشائی پیام ورودی تلاش می‌کند. اگر کلمهٔ عبور صحیح باشد، بلیت بطور موفقیت‌آمیزی بازگشائی می‌شود.

نظر باینکه قاعدتاً فقط کاربر اصلی بایستی کلمهٔ عبور را بداند، تنها کاربر اصلی میتواند بلیت را بازیابی کند. بنابراین ما از کلمهٔ عبور برای کسب امتیازات از Kerberos استفاده کرده بدون اینکه نیاز باشد تا کلمهٔ عبور را بصورت متن ساده و رمز نشده ارسال کنیم. بلیت، خود شامل ID و آدرس شبکهٔ کاربر و همچنین ID سِرور TGS است. این بخش نظیر سناریوی اول است و هدف این‌است که کاربر بتواند با استفاده از این بلیت، بلیت‌های اعطاکنندهٔ سرویس متعددی را درخواست کند. بنابراین بلیت اعطاکنندهٔ بلیت بایستی قابل استفادهٔ مکرر باشد. از سوی دیگر مایل نیستیم که یک دشمن بتواند بلیت را دزدیده و از آن استفاده کند. سناریوی زیر را در نظر بگیرید: یک دشمن بلیت را دزدیده و منتظر می‌ماند تا کاربر از ایستگاه کاری خود جدا شود. سپس این دشمن یا به ایستگاه کاری کاربر دسترسی فیزیکی یافته و یا ایستگاه کاری خود را با همان آدرس شبکهٔ ایستگاه کاری قربانی پیکربندی می‌کند. دشمن قادر خواهد بود تا مجدداً از بلیت استفاده کرده و TGS را فریب دهد. برای مقابله با این امر، بلیت شامل یک برچسب زمانی(timestamp) است که دارای تاریخ و لحظهٔ صدور و یک طول عمر که مشخص‌کننده محدودهٔ زمانی معتبر آن است، می‌باشد (مثلاً ۸ ساعت). بنابراین کلاینت حالا یک بلیت قابل استفاده مکرر داشته و لازم نیست تا برای هر سرویس جدید از کاربر تقاضای کلمهٔ عبور نماید. بالاخره توجه کنید که بلیت اعطاکنندهٔ بلیت با یک کلید رمز سرّی که تنها برای AS و TGS شناخته شده است رمزنگاری می‌شود. این امر تغییر بلیت را ناممکن می‌سازد. بلیت مجدداً با یک کلید که از کلمهٔ عبور کاربر مشتق شده است، رمزنگاری می‌شود. این موضوع این اطمینان را ایجاد میکند که بلیت تنها بتوسط کاربر معتبری که هویت صحیح خود را اظهار میکند می‌تواند استخراج شود.

حال که کلاینت یک بلیت اعطاکنندهٔ بلیت دارد، دسترسی به هر سِروری با استفاده از قدم‌های ۳ و ۴ امکان‌پذیر است.

۳- کلاینت به نمایندگی کاربر یک بلیت اعطاکنندهٔ سرویس را تقاضا می‌کند. برای این مقصود، کلاینت پیامی که شامل ID کاربر، ID سرویس مورد نیاز و بلیت اعطاکنندهٔ بلیت است را برای TGS می‌فرستد.

۴- TGS بلیت ورودی را با کلیدی(K_{tgs}) که تنها بین AS و TGS به اشتراک گذاشته شده است رمزگشائی کرده و موفقیت رمزگشائی را با کشف ID خود تأئید می‌نماید. همچنین کنترل می‌کند که طول عمر بلیت منقضی نشده باشد. سپس ID کاربر و آدرس شبکه را با اطلاعات ورودی تطبیق کرده تا اعتبار کاربر را بسنجد. اگر کاربر مجاز به دسترسی به V باشد، TGS یک بلیت برای دسترسی به سرویس تقاضاشده صادر می‌نماید.

بلیت اعطاکنندهٔ سرویس دارای همان ساختار بلیت اعطاکنندهٔ بلیت است. در واقع چون TGS یک سِرور است، طبیعتاً انتظار میرود که همان عناصری که برای معرفی یک کلاینت به TGS لازم‌اند برای معرفی کلاینت به سِرور کاربردها نیز مورد نیاز باشند. بازهم بلیت شامل برچسب زمانی و طول عمر است. اگر کاربر درخواست دسترسی به همان سرویس در زمانی دیگر را داشته باشد، کلاینت میتواند بسادگی از بلیت اعطاکنندهٔ سرویس قبلی استفاده کرده و مجدداً برای اخذ کلمهٔ عبور مزاحم کاربر نشود. توجه کنید که بلیت با کلید سری K_V رمزنگاری شده که این کلید فقط برای TGS و سِرور شناخته شده بوده و بنابراین دخل تصرف در آن ممکن نیست.

بالاخره با یک بلیت اعطاکنندهٔ سرویس، کلاینت می‌تواند از طریق قدم ۵ به سرویس مورد نظر دست یابد.

۵- کلاینت به نمایندگی کاربر تقاضای دسترسی به سرویسی را می‌نماید. برای این منظور کلاینت پیامی را به سرور منتقل می‌کند که شامل ID کاربر و بلیت اعطاکنندهٔ سرویس است. سِرور با استفاده از محتویات بلیت، اعتبار آن را می‌سنجد.

این سناریوی جدید دو نیاز ذکر شده یعنی فقط یکبار درخواست کلمهٔ عبور در هر مرتبه اتصال کاربر به شبکه، و همچنین محافظت از کلمهٔ عبور کاربر را برآورده می‌سازد.

دیالوگ اعتبارسنجی نسخهٔ ۴

اگرچه سناریوی ذکرشده در بالا در مقایسه با سناریوی اول، امنیت را بهبود می‌بخشد ولی هنوز دو مشکل باقی است. جوهر مشکل اول، طول عمری است که در بلیت اعطاکنندهٔ بلیت گنجانده شده است. اگر طول عمر خیلی کوتاه باشد (مثلاً چند دقیقه)، آنگاه به دفعات از کاربر تقاضای ارائهٔ کلمهٔ عبور خواهد شد. اگر طول عمر زیاد باشد(مثلاً ساعت‌ها)، آنگاه یک دشمن فرضی فرصت زیادتری برای بازخوانی خواهد داشت. یک دشمن ممکن است روی شبکه استراق‌سمع کرده و یک کپی از بلیت اعطاکنندهٔ بلیت را بدزدد و سپس آنقدر صبر کند تا کاربر قانونی از سیستم خارج شود. در این صورت دشمن میتواند آدرس شبکهٔ کاربر قانونی را جعل کرده و پیام مرحلهٔ (۳) را برای TGS ارسال کند. این امر به دشمن اجازه خواهد داد تا دسترسی نامحدودی به منابع و فایل‌های کاربر قانونی پیدا نماید.

بهمین ترتیب اگر دشمن یک بلیت اعطاکنندهٔ سرویس را دزدیده و از آن قبل از پایان مهلت استفاده کند، می‌تواند به سرویس‌های نظیر آن دست یابد.

بنابراین نیاز دیگری چهره می‌نماید. یک سرویس شبکه (TGS یا یک سرویس کاربردی) بایستی بتواند اثبات کند که شخصی که از بلیت استفاده می‌کند همان شخصی است که بلیت برای او صادر شده است.

مسألهٔ دوم این است که ممکن است نیاز باشد تا سِرورها نیز اعتبار خود را برای کاربران به اثبات برسانند. بدون اثبات چنین اعتباری، یک دشمن ممکن است پیکربندی را طوری مورد خرابکاری قرار دهد که پیام‌های بمقصد سِرور به محل دیگری بروند. در اینصورت سِرور قلابی بجای سِرور اصلی نشسته، اطلاعات کاربر را دریافت نموده و مانع از دادن سرویس صحیح به او می‌شود.

این مشکلات را بنوبت بررسی کرده و به جدول ۱-۴ که پروتکل Kerberos واقعی را نشان میدهد ارجاع می‌دهیم.

در وهلهٔ اول به مسألهٔ ربوده شدن بلیت اعطاکنندهٔ بلیت و نیاز به اثبات اینکه عرضه‌کنندهٔ بلیت همان کلاینتی است که بلیت برای او صادر شده است می‌پردازیم. تهدیدی که در اینجا وجود دارد این‌است که دشمن بلیت را دزدیده و قبل از انقضای مهلت از آن استفاده نماید. برای غلبه بر این مشکل فرض می‌کنیم AS را مجبور سازیم تا هم کلاینت و هم TGS را با نوعی اطلاعات سرّی به‌نحو امنی تجهیز نماید. آنگاه کلاینت می‌تواند هویت خود را با آشکارنمودن همان اطلاعات سرّی، بازهم به‌نحو امنی، به اثبات برساند. یک روش مؤثر برای انجام این امر استفاده از یک کلید رمزنگاری امن است که در Kerberos کلید اجلاس(Session key) خوانده می‌شود.

جدول ۱-۴الف روش توزیع کلید اجلاس را نشان می‌دهد. همانند قبل، کلاینت با ارسال یک پیام به AS درخواست دسترسی به TGS را می‌نماید. AS با یـک پیام،که بتوسط یک کلید که از کلمهٔ عبور کاربر مشتق شده (K_C) رمـزنگاری شده است و شامل بلیت است پاسخ می‌دهد. پیام رمزنگاری شده همچنین شامل یک کپی از کلید اجلاس $K_{c,tgs}$ است که در آن اندیس‌ها نشان می‌دهند که این کلید اجلاس C و TGS است. چون کلیـد اجلاس در درون پیامی است که با K_C رمزنگاری

جدول ۱-۴ خلاصه‌ای از مبادلهٔ پیام‌ها در Kerberos Version 4

(الف) مبادلهٔ سرویس اعتبارسنجی: برای کسب بلیت اعطاکنندهٔ بلیت	
(۱)	$\mathbf{C \rightarrow AS:}\quad ID_c \| ID_{tgs} \| TS_1$
(۲)	$\mathbf{AS \rightarrow C:}\quad E(K_c, [K_{c,tgs} \| ID_{tgs} \| TS_2 \| Lifetime_2 \| Ticket_{tgs}])$ $Ticket_{tgs} = E(K_{tgs}, [K_{c,tgs} \| ID_c \| AD_c \| ID_{tgs} \| TS_2 \| Lifetime_2])$
(ب) مبادلهٔ سرویس اعطاء- بلیت: برای کسب بلیت اعطاکنندهٔ سرویس	
(۳)	$\mathbf{C \rightarrow TGS:}\quad ID_v \| Ticket_{tgs} \| Authenticator_c$
(۴)	$\mathbf{TGS \rightarrow C:}\quad E(K_{c,tgs}, [K_{c,v} \| ID_v \| TS_4 \| Ticket_v])$ $Ticket_{tgs} = E(K_{tgs}, [K_{c,tgs} \| ID_c \| AD_c \| ID_{tgs} \| TS_2 \| Lifetime_2])$ $Ticket_v = E(K_v, [K_{c,v} \| ID_c \| AD_c \| ID_v \| TS_4 \| Lifetime_4])$ $Authenticator_c = E(K_{c,tgs}, [ID_c \| AD_c \| TS_3])$
(ج) مبادلهٔ اعتبارسنجی کلاینت/ سِرور: برای کسب سرویس	
(۵)	$\mathbf{C \rightarrow V:}\quad Ticket_v \| Authenticator_c$
(۶)	$\mathbf{V \rightarrow C:}\quad E(K_{c,v}, [TS_5 + 1])$ (برای اعتبارسنجی متقابل) $Ticket_v = E(K_v, [K_{c,v} \| ID_c \| AD_c \| ID_v \| TS_4 \| Lifetime_4])$ $Authenticator_c = E(K_{c,v}, [ID_c \| AD_c \| TS_5])$

شده است، تنها کلاینت کاربر می‌تواند آن را بخواند. همین کلید اجلاس در بلیت نیز قرار دارد که تنها می‌تواند بتوسط TGS خوانده شود. بنابراین کلید اجلاس بصورت امن هم به C و هم به TGS تحویل شده است.

توجه کنید که چند بخش اطلاعات اضافی به اولین فاز دیالوگ اضافه شده است. پیام (۱) شامل یک برچسب زمانی بوده تا AS بداند که پیام دارای محدودیت زمانی است. پیام (۲) شامل چندین عنصر بلیت به فرمی است که قابل دسترس برای C باشد. باین ترتیب C قادر است تأئید کند که این بلیت برای TGS بوده و از زمان انقضای آن آگاهی می‌یابد.

با مسلح شدن به بلیت و کلید اجلاس، C آماده است که به TGS نزدیک شود. همانند قبل، C پیامی را که شامل بلیت باضافهٔ ID سرویس درخواستی [پیام (۳) در جدول ۱-۴ب] است برای TGS می‌فرستد. علاوه برآن، C یک اعتبارسنج که شامل ID و آدرس کاربرC و یک برچسب زمانی است را ارسال می‌کند. برخلاف بلیت، که دوباره قابل استفاده است، اعتبارسنج فقط یک‌بار قابل استفاده بوده و طول عمر کوتاهی دارد. TGS میتواند با کلیدی که با AS به اشتراک دارد، بلیت را رمزگشائی نماید. این بلیت نشان میدهد که کاربر C با کلید اجلاس $K_{c,tgs}$ تجهیزشده است. در واقع بلیت میگوید، «هرکه از $K_{c,tgs}$ استفاده می‌کند بایستی C باشد». TGS از کلید اجلاس برای رمزگشائی اعتبارسنج استفاده می‌کند. بدنبال آن TGS می‌تواند اسم و آدرس استخراج شده از اعتبارسنج را با همین موارد در بلیت و آدرس شبکه‌ای که پیام از آن وارد شده است مقایسه نماید. اگر همهٔ اینها با هم تطبیق داشته باشند، آنگاه TGS مطمئن می‌شود که ارسال‌کنندهٔ این بلیت واقعاً همان صاحب بلیت است. در واقع تأئیدکننده می‌گوید «در زمان $TS3$، من بدین‌وسیله از $K_{c,tgs}$ استفاده می‌کنم.» توجه شود که بلیت هویت کسی را اثبات نمی‌کند بلکه روش امنی برای توزیع کلیدهاست. این اعتبارسنج است که هویت کلاینت را به اثبات می‌رساند. چون از اعتبارسنج تنها یک‌بار می‌توان استفاده کرد و دارای طول عمر کوتاهی نیز هست، امکان اینکه یک دشمن هم بلیت و هم اعتبارسنج را دزدیده و در آینده از آن استفاده کند از بین می‌رود.

پاسخ TGS در پیام (۴) از فرم پیام (۲) تبعیت می‌کند. پیام بتوسط کلید اجلاس که مشترک بین TGS و C است رمزنگاری شده و شامل یک کلید اجلاس که بین C و سِرورV مشترک است، ID سِرورV، و برچسب زمانی بلیت می‌باشد. خود بلیت شامل همان کلید اجلاس است.

C اکنون دارای یک بلیت اعطاکنندهٔ بلیت برایV است که می‌تواند بارها مورد استفاده قرار گیرد. وقتی C این بلیت را، همانند آنچه در پیام (۵) نشان داده شده است عرضه می‌دارد، یک اعتبارسنج را نیز با آن می‌فرستد. سِرور میتواند بلیت را رمزگشائی کرده، کلید اجلاس را استخراج نموده و اعتبارسنج را نیز از رمز درآورد.

اگر اعتبارسنجی متقابل مورد نیاز باشد، سِرور میتواند همانند پیام (۶) در جدول ۱-۴ پاسخ دهد. سِرور اندازهٔ برچسب زمانی در اعتبارسنج را به اندازهٔ یک واحد اضافه کرده و پس از رمزنگاری با کلید اجلاس آن را برمی‌گرداند. C می‌تواند این پیام را از رمز درآورده و برچسب زمانی افزایش یافته را استخراج نماید. نظر به اینکه پیام بتوسط کلید اجلاس رمز شده بود، C مطمئن است که این تنها می‌توانسته بتوسط V خلق شود. محتویات پیام به C اطمینان می‌دهدکه این بازخوانی یک پاسخ قدیمی نیست.

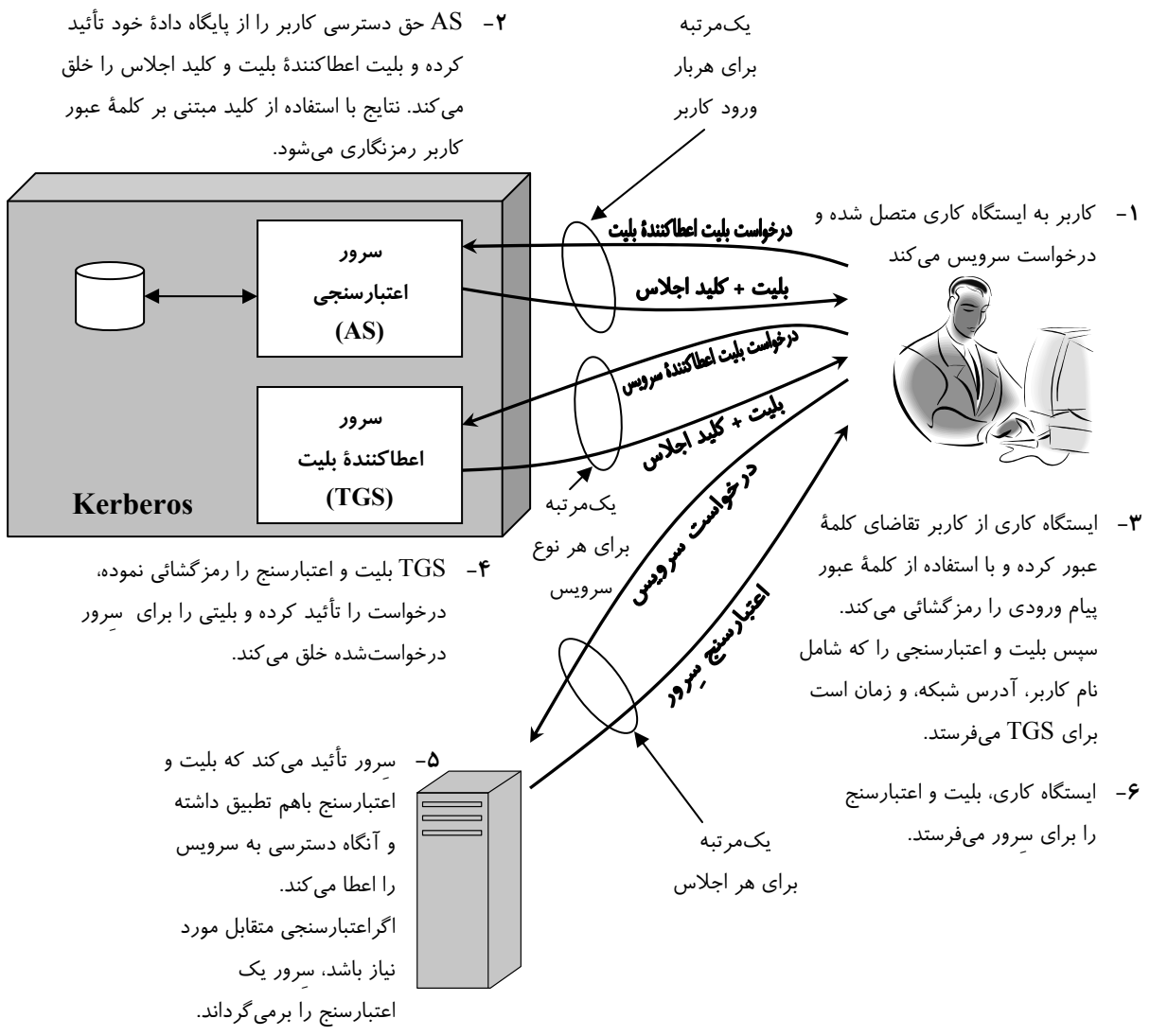
بالاخره در پایان این مرحله، کلاینت و سِرور یک کلید سرّی را به اشتراک می‌گذارند. از این کلید می‌توان برای رمزنگاری پیامهای آتی بین این دو، و یا برای مبادلهٔ یک کلید اجلاس تصادفی جدید برای این مقصود استفاده کرد.

جدول ۲-۴ وجود هریک از عناصر پروتکل Kerberos را توجیه کرده و شکل ۱-۴ یک نمای ساده از عملیات را نشان میدهد.

جدول ۲-۴ دلایل منطقی وجود مولفه‌های Kerberos Version 4

(الف) مبادلهٔ سرویس اعتبارسنجی	
پیام (۱)	کلاینت، بلیت اعطاکنندهٔ بلیت درخواست می‌کند
ID_c	به AS می‌گوید که هویت کاربری که از این کلاینت تماس می‌گیرد چیست
ID_{tgs}	به AS می‌گوید که کاربر تقاضای دست‌یابی به TGS را دارد
TS_1	AS را قادر می‌سازد تا تأئید کند که ساعت کلاینت با ساعت AS هم‌آهنگ است
پیام (۲)	AS بلیت اعطاکنندهٔ بلیت را تهیه کرده و برای کلاینت برمی‌گرداند
K_c	رمزنگاری مبتنی بر کلمهٔ عبور کاربر است، AS و کلاینت را قادر می‌سازد تا کلمهٔ عبور را تأئید نمایند و همچنین محتویات پیام (۲) را محافظت می‌نماید
$K_{c,tgs}$	کپی کلید اجلاس که به توسط کلاینت قابل دسترسی است. به توسط AS خلق شده است تا بدون این‌که نیاز به یک کلید دائمی باشد مبادلهٔ امن بین کلاینت و TGS را امکان‌پذیر نماید
ID_{tgs}	تائید می‌کند که این بلیت برای TGS است
TS_2	کلاینت را از زمان صدور این بلیت آگاه می‌سازد
$Lifetime_2$	کلاینت را از طول عمر این بلیت آگاه می‌سازد
$Ticket_{tgs}$	بلیتی است که از سوی کلاینت برای دسترسی به TGS مورد استفاده قرار می‌گیرد
(ب) مبادلهٔ سرویس اعطاء بلیت	
پیام (۳)	کلاینت، بلیت اعطاکنندهٔ سرویس درخواست می‌کند
ID_v	به TGS می‌گوید که کاربر تقاضای دسترسی به سِرور V را دارد
$Ticket_{tgs}$	به TGS اطمینان می‌دهد که این کاربر به توسط AS اعتبارسنجی شده است
$Authenticator_c$	بتوسط کلاینت تولید شده تا بلیت را معتبر نماید
پیام (۴)	TGS بلیت اعطاکنندهٔ سرویس را تهیه کرده و برای کلاینت برمی‌گرداند
$K_{c,tgs}$	کلیدی که فقط بین C و TGS مشترک است و محتویات پیام (۴) را محافظت می‌کند
$K_{c,v}$	کپی کلید اجلاس قابل دسترسی بتوسط کلاینت. بتوسط TGS خلق می‌گردد تا بدون نیاز به یک کلید دائم مشترک بین کلاینت و سِرور، مبادلهٔ امن بین آنها را امکان‌پذیر نماید
ID_v	تأئید می‌کند که این بلیت برای سِرور V است
TS_4	کلاینت را از زمان صدور این بلیت آگاه می‌سازد
$Ticket_v$	بلیتی است که از سوی کلاینت برای دسترسی به سِرور V مورد استفاده قرار می‌گیرد
$Ticket_{tgs}$	قابل استفادهٔ مکرر است تا کاربر مجبور نباشد تا هربار کلمهٔعبور خود را وارد کند
K_{tgs}	بلیت با کلیدی که فقط برای AS و TGS شناخته شده است رمزنگاری می‌شود تا از تحریف آن جلوگیری شود
$K_{c,tgs}$	کلید اجلاس قابل دسترسی بتوسط TGS. برای رمزگشائی اعتبارسنج و بنابراین اعتبارسنجی بلیت بکار می‌رود
ID_c	نمایش‌گر صاحب واقعی این بلیت است
AD_c	از استفاده از بلیت بتوسط یک ایستگاه کاری، بجز ایستگاهی که بدواً تقاضای بلیت کرده بود جلوگیری می‌کند

ID_{tgs}	به سِرور اطمینان می‌دهد که بلیت را بطور صحیح رمزگشائی کرده است
TS_2	TGS را از زمان صدور این بلیت آگاه می‌سازد
$Lifetime_2$	از بازخوانی پس از انقضای بلیت جلوگیری می‌کند
$Authenticator_c$	به TGS اطمینان می‌دهد که عرضه‌کنندهٔ بلیت همان کلاینتی است که بلیت برای او صادر شده است. دارای طول عمر کوتاهی است تا از بازخوانی جلوگیری شود
$K_{c,tgs}$	اعتبارسنج با کلیدی که تنها برای کلاینت و TGS شناخته شده است رمزنگاری می‌شود تا از تحریف جلوگیری شود
ID_C	بایستی با ID بلیت تطبیق داشته باشد تا بلیت معتبر شناخته شود
AD_C	بایستی با آدرس بلیت تطبیق داشته باشد تا بلیت معتبر شناخته شود
TS_3	TGS را از زمان تولید این اعتبارسنج آگاه می‌سازد
	(ج) مبادلهٔ اعتبارسنجی کلاینت/ سِرور
پیام (۵)	کلاینت درخواست سرویس می‌کند
$Ticket_v$	به سِرور اطمینان می‌دهد که این کاربر بتوسط AS اعتبارسنجی شده است
$Authenticator_c$	بتوسط کلاینت خلق شده تا بلیت را معتبر سازد
پیام (۶)	اعتبارسنجی اختیاری سِرور برای کلاینت
$K_{c,v}$	به C اطمینان می‌دهد که این پیام از سوی V است
$TS_5 + 1$	به C اطمینان می‌دهد که این بازخوانی یک پاسخ قدیمی نیست
$Ticket_v$	قابل استفادهٔ مجدد بوده تا لازم نباشد که کاربر برای هربار دست‌یابی به یک سِرور معین تقاضای یک بلیت جدید کند
K_v	بلیت بتوسط کلیدی که فقط برای TGS و سِرور شناخته شده است رمزنگاری شده تا از تحریف جلوگیری شود
$K_{c,v}$	کپی کلید اجلاس قابل دست‌یابی بتوسط کلاینت. برای رمزگشائی اعتبارسنج و بنابراین تأئید اعتبار بلیت استفاده می‌شود
ID_C	صاحب اصلی این بلیت را نشان می‌دهد
AD_C	از استفاده از بلیت یک ایستگاه کاری بجز آن که بدواً تقاضای بلیت کرده بود جلوگیری می‌کند
ID_V	به سِرور اطمینان می‌دهد که که بلیت را بطرز صحیح رمزگشائی کرده است
TS_4	سِرور را از زمان صدور این بلیت آگاه می‌سازد
$Lifetime_4$	از بازخوانی پس از انقضای بلیت جلوگیری می‌کند
$Authenticator_c$	به سِرور اطمینان می‌دهد که عرضه‌کنندهٔ بلیت همان کلاینتی است که بلیت برای او صادر شده است. دارای طول عمر کوتاهی است تا از بازخوانی جلوگیری شود
$K_{c,v}$	اعتبارسنج با کلیدی که تنها برای کلاینت و سِرور شناخته شده است رمزنگاری می‌شود تا از تحریف جلوگیری شود
ID_C	بایستی با ID بلیت تطبیق داشته باشد تا بلیت معتبر شناخته شود
AD_C	بایستی با آدرس بلیت تطبیق داشته باشد تا بلیت معتبر شناخته شود
TS_5	سِرور را از زمان تولید این اعتبارسنج آگاه می‌سازد



شکل ۱-۴ مروری بر Kerberos

قلمرو Kerberos و Kerberosهای متعدد

محیط خدماتی کامل یک سِرور Kerberos که شامل یک سِرور Kerberos، تعدادی کلاینت و تعدادی سِرورهای کاربردی است به موارد زیر نیاز دارد:

۱- سِرور Kerberos بایستیID کاربران(UID) و کلمهٔعبور درهم‌سازی‌شدهٔ همهٔ کاربران حوزه را در پایگاه دادهٔ خود داشته باشد. تمام کاربران بایستی در نزد Kerberos ثبت‌نام شده باشند.

۲- سِرور Kerberos بایستی با هر سِرور دیگر یک کلید سرّی مشترک داشته باشد. تمام سِرورها بایستی در نزد سِرور Kerberos ثبت‌نام شده باشند.

چنین محیطی را یک **قلمرو(Kerberos realm)** خوانند. مفهوم یک *قلمرو* را می‌توان چنین تشریح کرد: یک قلمرو Kerberos یک مجموعه از گره‌های مدیریت شده است که همگی پایگاه دادهٔ Kerberos را در اشتراک دارند. پایگاه دادهٔ Kerberos در سیستم کامپیوتری اصلی Kerberos قرار دارد که نوعاً بایستی در یک اطاق با امنیت فیزیکی خوب قرار داشته باشد. یک نسخهٔ فقط قابل خواندن از این پایگاه داده نیز قاعدتاً می‌تواند روی کامپیوترهای دیگر سیستم نصب شود. دست‌یابی و یا تغییر محتوای پایگاه دادهٔ Kerberos نیاز به کلمهٔ عبور اصلی Kerberos دارد. مفهوم دیگری که با این مسأله مرتبط است، وجود یک **رئیس (Kerberos principal)** است که سرویس و یا کاربری است که برای سیستم Kerberos شناخته شده است. هر رئیس Kerberos با نام ریاست خود شناخته می‌شود. نام‌های ریاست دارای سه جزء نام یک سرویس و یا یک کاربر، نام یک مورد، و نام یک قلمرو می‌باشند.

شبکه‌های متشکل از کلاینت‌ها و سِرورها در سازمان‌های مدیریتی مختلف، معمولاً قلمروهای متفاوتی را تشکیل می‌دهند. این که کاربران و سِرورهای یک حوزهٔ مدیریتی، در سِرور Kerberos حوزهٔ مدیریتی دیگری ثبت‌نام شده باشند نه عملی است و نه معمولاً با خط‌مشی‌های اداری منطبق است. از سوی دیگر، ممکن است کاربران یک قلمرو نیاز به دست‌یابی به سِرورهای قلمرو دیگری داشته و یا اینکه بعضی سِرورهای یک قلمرو تمایل داشته باشند تا به کاربران معتبر قلمرو دیگر ارائهٔ سرویس نمایند.

Kerberos مکانیسمی را برای حمایت از این اعتبارسنجی بین قلمروها ایجاد نموده است. برای اینکه دو قلمرو از اعتبارسنجی بین قلمروها حمایت کنند، نیاز سومی به پروتکل اضافه می‌شود:

۳- سِرور Kerberos هر قلمرو بایستی با سِرور Kerberos قلمرو دیگر یک کلید سرّی را به اشتراک بگذارد. دو سِرور Kerberos بایستی در یکدیگر ثبت‌نام شده باشند.

این روش نیازمند این‌است که سِرور Kerberos یک قلمرو به سِرور Kerberos قلمرو دیگر، نسبت به سنجش اعتبار کاربران خود اعتماد داشته باشد. علاوه بر آن سِرورهای قلمرو دوم نیز بایستی تمایل به اعتماد به سِرور Kerberos قلمرو اول داشته باشند.

با استقرار این قواعد در جای خود، مکانیسم عمل را می‌توان بصورت زیر تشریح کرد (شکل ۲-۴): کاربری که مایل به اخذ سرویس از سِروری در قلمرو دیگر است، نیاز به یک بلیت برای آن سِرور دارد. کلاینت کاربر، همان روش‌های ذکرشده برای دسترسی به TGS محلی را دنبال کرده و سپس یک بلیت اعطاکنندهٔ بلیت برای TGS دور (TGS قلمرو دیگر) تقاضا می‌کند. بدنبال آن کلاینت می‌تواند از TGS دور درخواست یک بلیت اعطاکنندهٔ سرویس جهت استفاده از سِرور مورد نیاز در قلمرو او را بنماید.

جزئیات مبادلاتی که در شکل ۲-۴ نشان داده شده است بشرح زیر است (با جدول ۱-۴ مقایسه کنید):

(۱) C→AS: $ID_c \| TS_1 \| ID_{tgs}$

(۲) S→C: $E(K_c, [K_{c,tgs} \| ID_{tgs} \| TS_2 \| Lifetime_2 \| Ticket_{tgs}])$

(۳) C→TGS: $ID_{tgsrem} \| Ticket_{tgs} \| Authenticator_c$

(۴) TGS→ C: $E(K_{c,tgs}, [K_{c,tgsrem} \| ID_{tgsrem} \| TS_4 \| Ticket_{tgsrem}])$

(۵) C→ TGS_{rem} : $ID_{vrem} \| Ticket_{tgsrem} \| Authenticator_c$

(۶) TGS_{rem} → C: $E(K_{c,tgsrem}, [K_{c,vrem} \| ID_{vrem} \| TS_6 \| Ticket_{vrem}])$

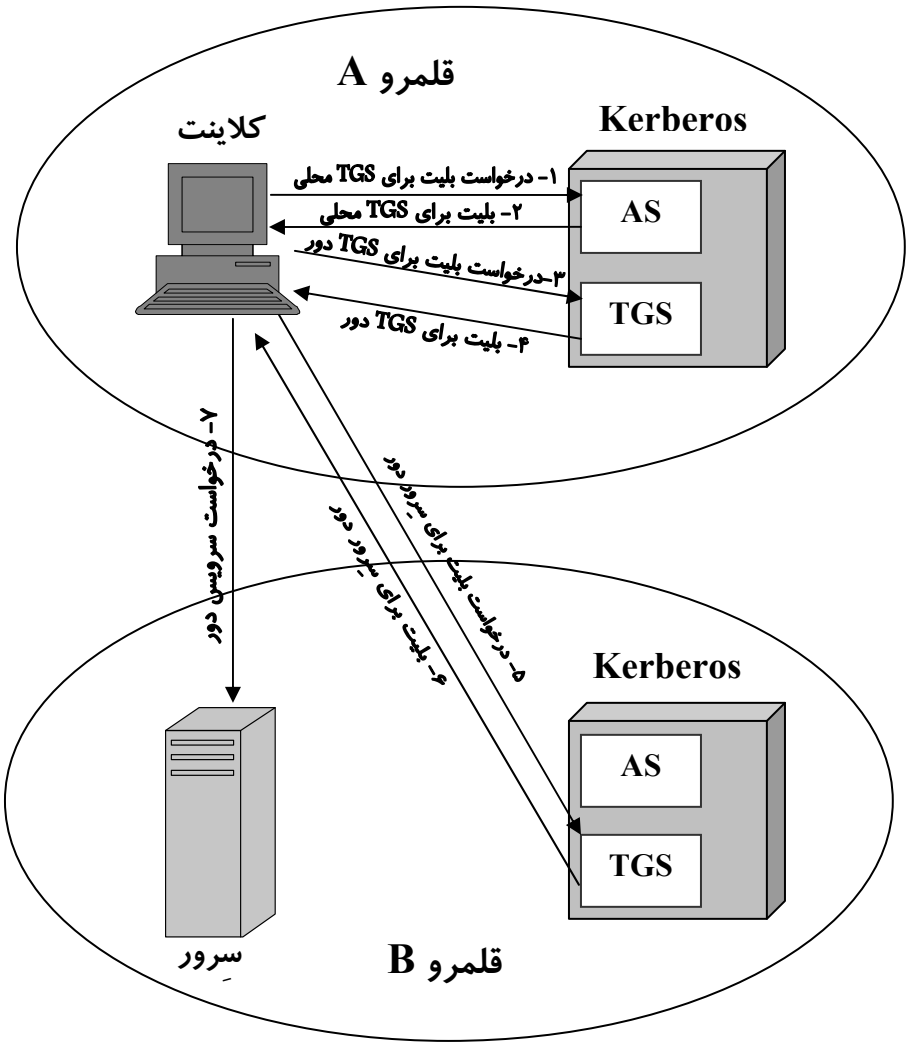
(۷) C → V_{rem} : $Ticket_{vrem} \| Authenticator_c$

بلیت ارائه شده به سِرور دور (V_{rem}) نشان‌دهندهٔ قلمروی است که درآن ابتداءً کاربر اعتبارسنجی شده بود. سِرور در پاسخ دادن به این درخواست مخیّر است.

یکی از مشکلاتی که در روش بالا خودنمائی میکند این است که در صورت افزایش زیاد قلمروها، این روش به‌نحو مناسبی مقیاس‌پذیر نیست. اگر N قلمرو وجود داشته باشد، آنگاه بایستی $N(N\text{-}1)/2$ مبادلهٔ امن کلیدها صورت پذیرد تا قلمرو هرKerberos بتواند با قلمروهای بقیهٔ Kerberosها تعامل نماید.

نسخهٔ پنجم Kerberos

نسخهٔ پنجم Kerberos در RFC 1510 تعریف شده و مزیت‌هائی نسبت به نسخهٔ چهارم آن دارد[KOHL94]. برای شروع، نگاهی به تفاوت‌های نسخهٔ پنجم نسبت به نسخهٔ چهارم نموده و سپس پروتکل نسخهٔ ۵ را بررسی می‌کنیم.



شکل ۲-۴ درخواست سرویس از قلمرو دیگر

تفاوت‌های بین نسخهٔ ۴ و نسخهٔ ۵

نسخهٔ پنجم برای رفع محدودیت‌های نسخهٔ چهارم در دو زمینه طراحی شده است: نواقص محیطی و کمبودهای تکنیکی. اجازه دهید تا بطور مختصر بهبودهای ایجادشده در هر زمینه را خلاصه کنیم.

نسخهٔ چهارم Kerberos برای استفاده در محیط پروژهٔ Athena طراحی شده بود و بنابراین نیاز در زمینهٔ اهداف کلی را نادیده گرفته است. این امر باعث ایجاد **نقائص محیطی** زیر شده است:

۱- **وابستگی به سیستم رمزنگاری**: نسخهٔ ۴ نیاز به استفاده از DES دارد. محدودیت‌های صادراتی DES و همچنین تردید در مورد توانائی‌های آن، از موارد نگران‌کننده است. در نسخهٔ ۵، متن رمزشده با یک شناسهٔ نوع رمزنگاری مجهزشده و بنابراین می‌توان از هر نوع تکنیک رمزنگاری استفاده کرد. کلیدهای رمزنگاری دارای یک دنبالهٔ مشخص‌کنندهٔ نوع و اندازه بوده و این امر اجازه میدهد تا از یک کلید در الگوریتم‌های مختلف استفاده کرده و همچنین تغییرات متفاوتی را روی یک الگوریتم داده شده انجام داد.

۲- **وابستگی به پروتکل اینترنت**: نسخهٔ ۴ نیاز به استفاده از آدرس‌های پروتکل اینترنت(IP) دارد. سایر انواع آدرس، مثل آدرس شبکهٔ ISO، را نمی‌توان بکار گرفت. در نسخهٔ ۵، آدرس‌های شبکه با دنبالهٔ نوع و طول مجهز بوده و اجازه می‌دهد تا از هر نوع آدرسی استفاده کرد.

۳- **نظم بایت‌های پیام**: در نسخهٔ ۴، ارسال‌کنندهٔ یک پیام نظم بایت‌ها را از جانب خود انتخاب کرده و با الصاق یک دنباله به پیام مشخص می‌سازد که آیا بایت دارای کمترین اهمیت در پائین‌ترین آدرس قرار دارد و یا بایت دارای بیشترین اهمیت پائین‌ترین آدرس را اشغال کرده است. این روش عملی است ولی از قراردادهای جاافتاده تبعیت نمی‌کند. در نسخهٔ ۵ تمام ساختارهای پیام با استفاده از Abstract Syntax Notation One (ASN.1) و Basic Encoding Rule(BER) تعریف شده و بنابراین نظم بایت‌ها بطور غیرقابل ابهامی مشخص می‌گردند.

۴- **طول عمر بلیت**: اندازه‌های طول عمر در نسخهٔ ۴ بصورت یک کمیت ۸- بیتی در واحدهای ۵ دقیقه‌ای کُد شده‌اند. بنابراین ماکزیمم طول عمری که می‌تواند تعریف شود برابر ۱,۲۸۰= ۵ ×$۲^{۸}$ دقیقه و یا چیزی بالاتر از ۲۱ ساعت است. این مقدار برای برخی کاربردها ممکن است کافی نباشد (مثل یک شبیه‌سازی طولانی که در طول اجرا نیاز به پشتیبانی مستمر Kerberos دارد). در نسخهٔ ۵، بلیت‌ها دارای یک زمان شروع و یک زمان خاتمه صریح بوده و بنابراین یک بلیت می‌تواند هر طول عمری را داشته باشد.

۵- **جلوراندن اعتبارسنجی**: نسخهٔ ۴ اجازه نمی‌دهد که اعتبار صادر شده برای یک کلاینت به میزبان دیگری اهدا شده و کلاینت دیگری از آن استفاده نماید. این قابلیت به یک کلاینت اجازه خواهد داد تا به یک سِرور دست یافته و از آن سِرور بخواهد که از طرف آن کلاینت به سِرور دیگری دست یابد. برای مثال، یک کلاینت تقاضائی را برای یک سِرور چاپگر می‌فرستد که با استفاده از اعتبار کلاینت به فایل کلاینت در یک سِرور فایل دسترسی یابد. نسخهٔ ۵ این قابلیت را فراهم آورده است.

۶- **اعتبارسنجی بین قلمروها**: در نسخهٔ ۴، عملیات بین N قلمرو نیاز به حدود N^2 ارتباط Kerberos به Kerberos داشته که قبلاً در مورد آن بحث شد. نسخهٔ ۵ برابر آنچه بزودی تشریح خواهد شد نیاز به روابط کمتری دارد.

جدا از این محدودیت‌های محیطی، در خود پروتکل نسخهٔ ۴ یک سری **نواقص تکنیکی** وجود دارد. بیشتر این نواقص در[BELL90] ذکر شده و نسخهٔ ۵ برای رفع آنها تلاش کرده است. نواقص به قرار زیراند:

۱- **رمزنگاری دوبل**: در جدول۱-۴ توجه کنید [پیام‌های(۲)و(۴)] که بلیت‌های صادرشده برای کلاینت‌ها دوبار رمزنگاری می‌شوند که بار اول با کلید سرّی سِرور هدف و بار دوم با کلید سرّی آشنا برای کلاینت این‌کار صورت می‌پذیرد. رمزنگاری بار دوم مورد نیاز نبوده و از نظر محاسباتی وقت‌گیر است.

۲- **رمزنگاریPCBC**: رمـزنگاری در نسخهٔ ۴ از یک مُـود غیراستانـــدارد DES به نام (PCBC) propagating cipher block chaining استفاده می‌کند. نشان داده شده است که این مُود نسبت به یک حمله که شامل تعویض بلوک‌های رمزشده است، آسیب‌پذیراست[KOHL89]. PCBC قرار بود تا بعنوان بخشی از عملیات رمزنگاری، صحت داده‌ها را نیز کنترل نماید. نسخهٔ ۵ یک مکانیسم کنترل صحت صریح ایجاد کرده که اجازه میدهد تا از مُود CBC استاندارد برای رمزنگاری استفاده شود. علی‌الخصوص، یک جمع کنترلی یا کُد hash قبل از رمزنگاری با استفاده از CBC به پیام وصل می‌گردد.

۳- **کلیدهای اجلاس**: هر بلیت شامل یک کلید اجلاس است که از طرف کاربر برای رمزکردن اعتبارسنج ارسال‌شدهٔ متناظر با آن بلیت بکار می‌رود. علاوه برآن، کلید اجلاس می‌تواند متعاقباً بتوسط کلاینت و سِرور برای محافظت پیام‌هائی که در خلال اجلاس ردوبدل می‌شوند بکار رود. ولی چون یک بلیت ممکن است مکرراً برای دست‌یابی به سرویس یک سِرور خاص مورد استفاده قرار گیرد، این خطر وجود دارد که یک دشمن پیام‌های مربوط به یک اجلاس کهنه را برای کلاینت یا سِرور بازخوانی کند. در نسخهٔ ۵ این امکان وجود دارد که یک کلاینت با سِرور در مورد یک کلید زیراجلاس به این توافق برسند که از این کلید فقط برای یک بار اتصال استفاده شود. دسترسی جدید از سوی کلاینت، نیاز به استفاده از یک کلید زیراجلاس جدید دارد.

۴- **حملات کلمهٔ عبور**: هر دو نسخه در مقابل حملهٔ کلمهٔ عبور آسیب‌پذیرند. پیام AS به کلاینت شامل مطالبی است که با یک کلید که مشتق از کلمهٔ عبور کلاینت است رمزنگاری شده است. یک دشمن ممکن است این پیام را دزدیده و با بکارگیری کلمات عبور مختلف، آن را رمزگشائی نماید. اگر نتیجهٔ یکی از این رمزگشائی‌ها موفقیت‌آمیز باشد، آنگاه دشمن کلمهٔعبور کلاینت را کشف کرده و ممکن است بعداً آن را برای کسب امتیازات اعتبارسنجی از Kerberos بکار برد. این همان نوع حملهٔ کلمهٔ عبوری است که در فصل ۹ در مورد آن بحث شده و همان پاتک‌های ذکرشده در مورد آن قابل اجراست. نسخهٔ ۵، مکانیسم جدیدی بنام پیش‌اعتبارسنجی بکاربرده که حملات کلمهٔعبور را دشوارتر نموده ولی کاملاً از آنها جلوگیری نمی‌نماید.

دیالوگ اعتبارسنجی نسخهٔ ۵

جدول ۳-۴ دیالوگ اصلی نسخهٔ ۵ را خلاصه کرده است. این دیالوگ می‌تواند به بهترین نحو با مقایسه با نسخهٔ ۴ تشریح گردد (جدول ۱-۴).

در ابتدا **مبادلهٔ سرویس اعتبارسنجی** را درنظر بگیرید. پیام(۱) درخواست کلاینت برای یک بلیت اعطاکنندهٔ بلیت است. همانند قبل این درخواست شامل ID کاربر و TGS است. اقلام جدید زیر به آن اضافه شده‌اند:

جدول ۳-۴ خلاصهٔ مبادلهٔ پیام‌ها در Kerberos Version 5

(الف) مبادلهٔ سرویس اعتبارسنجی: برای کسب بلیت اعطاکنندهٔ بلیت	
(۱)	**C→ AS:** Options || ID_C || $Realm_c$ || ID_{tgs} || $Times$ || $Nonce_1$
(۲)	**AS→ C:** $Realm_c$ || ID_C || $Ticket_{tgs}$ || E (K_c, [$K_{c,tgs}$ || $Times$ || $Nonce_1$ || $Realm_{tgs}$ || ID_{tgs}])
	$Ticket_{tgs}$ = E (K_{tgs} , [$Flags$ || $K_{c,tgs}$ || $Realm_c$ || ID_C || AD_C || $Times$])
(ب) مبادلهٔ سرویس اعطاء بلیت: برای کسب بلیت اعطاکنندهٔ سرویس	
(۳)	**C→ TGS:** Options || ID_v || $Times$ || $Nonce_2$|| $Ticket_{tgs}$ || $Authenticator_c$
(۴)	**TGS→ C:** $Realm_c$ || ID_C || $Ticket_v$ || E($K_{c,tgs}$, [$K_{c,v}$ || $Times$ || $Nonce_2$ || $Realm_v$ || ID_v])
	$Ticket_{tgs}$ = E (K_{tgs} , [$Flags$ || $K_{c,tgs}$|| $Realm_c$ || ID_C || AD_C || $Times$])
	$Ticket_v$ = E (K_v , [$Flags$ || $K_{c,v}$ || $Realm_c$ || ID_C || AD_C || $Times$])
	$Authenticator_c$ = E ($K_{c,tgs}$, [ID_C || $Realm_c$ || TS_1])
(ج) مبادلهٔ اعتبارسنجی کلاینت/ سِرور: برای کسب سرویس	
(۵)	**C→ V:** Options || $Ticket_v$ || $Authenticator_c$
(۶)	**V→ C:** E ($K_{c,v}$, [TS_2 || $Subkey$ || $Seq\#$])
	$Ticket_v$ = E (K_v , [$Flags$ || $K_{c,v}$ || $Realm_c$ || ID_C || AD_C || $Times$])
	$Authenticator_c$ = E ($K_{c,v}$, [ID_C || $Realm_c$ || TS_2 || $Subkey$ || $Seq\#$])

- **قلمرو(Realm)**: قلمرو کاربر را نشان می‌دهد.
- **موارد اختیاری(Options)**: افراشتن پرچم‌های معینی در بلیت بازگشتی را درخواست می‌نماید.
- **زمان‌ها(Times)**: تنظیم زمان‌های زیر در بلیت، با تقاضای کلاینت، را ممکن می‌سازد:
  - از: زمان آغاز برای بلیت تقاضاشده
  - تا: زمان انقضاء برای بلیت تقاضاشده
  - تمدید: زمان تمدید برای بلیت تقاضاشده
- **حال فعلی(nonce)**: یک مقدار تصادفی که بایستی در پیام(۲) تکرار گردد تا مطمئن شویم که پاسخ تازه بوده و فرم قدیمی بازخوانی شده بتوسط دشمن نمی‌باشد.

پیام (۲) یک بلیت اعطاکنندهٔ بلیت را برگردانده، اطلاعات مربوط به کلاینت را بیان نموده و یک بلوک رمزنگاری‌شده با استفاده از کلید رمزی که از کلمهٔ عبور کاربر مشتق شده است را تهیه می‌کند. این بلوک شامل کلید اجلاسی بوده که بایستی بین کلاینت و TGS مورد استفاده قرار گیرد، زمان‌هائی که در پیام (۱) مشخص شده، nonce از پیام (۱) و اطلاعات شناسائی TGS است. خود بلیت شامل کلید اجلاس، اطلاعات شناسائی کلاینت، مقادیر زمانی تقاضاشده و پرچم‌هائی است که وضعیت این بلیت و اختیارات تقاضاشده را نشان می‌دهند. این پرچم‌ها قابلیت‌های جدید قابل ملاحظه‌ای را به نسخهٔ ۵ اضافه می‌کنند. فعلاً بحث در مورد این پرچم‌ها را به تعویق انداخته و روی ساختار کلی پروتکل نسخهٔ ۵ تمرکز می‌کنیم.

حال اجازه دهید تا **مبادلهٔ سرویس اعطاکردن بلیت** در نسخه‌های ۴ و ۵ را با هم مقایسه کنیم. همانطور که دیده می‌شود، پیام (۳) برای هر دو نسخه شامل یک اعتبارسنج، یک بلیت و نام سرویس درخواستی می‌باشد. علاوه بر آن نسخهٔ ۵ شامل زمان‌های تقاضاشده و موارد اختیاری بلیت و یک nonce است که همهٔ آنها شبیه موارد پیام (۱) اند. اعتبارسنج خود ضرورتاً همانی است که در نسخهٔ (۴) بکار رفته است.

پیام (۴) دارای همان ساختار پیام (۲) بوده و یک بلیت را بهمراه اطلاعاتی که مورد نیاز کلاینت است برمی‌گرداند. اطلاعات با همان کلید اجلاسی که حالا بین کلاینت و TGS مشترک است، رمزنگاری می‌گردد.

بالاخره، برای **مبادلهٔ اعتبارسنجی کلاینت/سِرور**، چند مشخصهٔ جدید در نسخهٔ ۵ گنجانده شده است. در پیام (۵)، کلاینت بعنوان یک درخواست اختیاری، ممکن است تقاضای اعتبارسنجی متقابل نماید. برای این منظور، اعتبارسنج دارای چند میدان جدید است:

- **زیرکلید (subkey)**:کلاینت حق دارد تا یک کلید رمزنگاری برای حفاظت این اجلاس کاربردی خاص درخواست کند. اگر این میدان حذف شود، کلید اجلاس بلیت($K_{C,V}$) مورد استفاده قرار خواهد گرفت.
- **شمارهٔ ردیف (Sequence number)**: یک میدان اختیاری تعیین‌کنندهٔ شمارهٔ ردیف آغازین برای استفادهٔ سِرور در شماره‌گذاری پیام‌هایی است که در این اجلاس برای کلاینت ارسال می‌شود. پیام‌ها از این‌جهت ممکن است شماره‌گذاری شوند تا حمله‌ای از نوع بازخوانی را کشف نمایند.

اگر اعتبارسنجی متقابل مورد نیاز باشد، سِرور با پیام (۶) پاسخ می‌دهد. این پیام شامل برچسب زمانی اعتبارسنج است. توجه کنید که در نسخهٔ ۴، برچسب زمانی یک واحد افزایش می‌یافت. این امر در نسخهٔ ۵ مورد نیاز نبوده زیرا فرمت پیام بنحوی است که برای یک دشمن امکان‌پذیر نخواهد بود که پیام (۶) را بدون اطلاع از کلیدهای رمزنگاری مناسب، خلق نماید. میدان زیرکلید، اگر وجود داشته باشد، میدان زیرکلید در پیام ۵ اگر وجود داشته باشد را ملغی می‌سازد. میدان اختیاری شماره ردیف، اولین شماره‌ای که بایستی بتوسط کلاینت مورد استفاده قرار گیرد را تعیین می‌کند.

پرچم‌های بلیت (Ticket Flags)

میدان‌های پرچم قرارداده شده در نسخهٔ ۵، قابلیت‌های عملکرد را نسبت به آنچه که در نسخهٔ ۴ است توسعه بخشیده است. جدول ۴-۴ پرچم‌هائی که ممکن است در یک بلیت وجود داشته باشند را نشان داده است.

پرچم INITIAL نشان می‌دهد که این بلیت بتوسط AS و نه TGS صادر شده است. وقتی یک کلاینت از TGS درخواست یک بلیت اعطاکنندهٔ- سرویس می‌نماید، یک بلیت اعطاکنندهٔ بلیت را که از AS دریافت کرده است عرضه می‌دارد. در نسخهٔ ۴، این تنها راه اخذ یک بلیت اعطاکنندهٔ- سرویس بود. نسخهٔ ۵ این قابلیت جدید را فراهم می‌سازد که کلاینت بتواند یک بلیت اعطاکنندهٔ- سرویس را مستقیماً از AS دریافت نماید. فایدهٔ این امر چنین است: یک سِرور، مثل یک سِرور تعویض‌کنندهٔ کلمهٔ عبور، ممکن است علاقه‌مند باشد تا بداند که کلمهٔ عبور کلاینت جدیداً تست شده است.

پرچم PRE-AUTHENT، اگر افراشته باشد، نمایشگر این مطلب است که وقتی AS تقاضای اولیه را دریافت کرده است (پیام (۱))، اعتبار کلاینت را قبل از صدور یک بلیت سنجیده است. شکل دقیق این پیش‌اعتبارسنجی، تعیین‌نشده باقی مانده است. برای مثال، در نسخهٔ ۵ ساخت MIT، پیش‌اعتبارسنج برچسب زمانی را رمزنگاری کرده است که در حالت پیش‌فرض فعال خواهد بود. وقتی کاربری می‌خواهد تا یک بلیت دریافت نماید، بایستی برای AS یـک بلوک پیش اعتبارسنج

جدول ۴-۴ پرچم‌های Kerberos Version 5

INITIAL	این بلیت با استفاده از پروتکل AS صادر شده است و بر اساس بلیت اعطاکنندهٔ بلیت صادر نشده است.
PRE-AUTHENT	در هنگام اعتبارسنجی اولیه، کلاینت قبل از صدور بلیت بتوسط KDC اعتبارسنجی شده است.
HW-AUTHENT	پروتکلی که برای اعتبارسنجی اولیه بکارگرفته شده است نیاز به استفاده از سخت‌افزاری داشته است که انتظار می‌رود منحصراً در مالکیت کلاینت نام برده شده، بوده باشد.
RENEWABLE	به TGS می‌گوید که این بلیت می‌تواند برای کسب یک بلیت جایگزین که در تاریخ دیرتری منقضی می‌گردد بکار رود.
MAY-POSTDATE	به TGS می‌گوید که یک بلیت آتی می‌تواند بر اساس این بلیت اعطاکنندهٔ بلیت صادر شود.
POSTDATED	نشان می‌دهد که این بلیت تمدید شده است. سِرور انتهائی می‌تواند میدان زمان اعتبارسنجی را کنترل کرده تا از زمان اعتبارسنجی اولیه خبردار گردد.
INVALID	این کلید دارای اعتبار نبوده و بایستی قبل از استفاده بتوسط KDC اعتبار آن تأئید شود.
PROXIABLE	به TGS می‌گوید که یک بلیت اعطاکنندهٔ سرویس جدید با یک آدرس شبکهٔ متفاوت می‌تواند بر اساس بلیت عرضه‌شده صادر شود.
PROXY	نشان می‌دهد که این بلیت یک پروکسی است.
FORWARDABLE	به TGS می‌گوید که یک بلیت اعطاکنندهٔ بلیت جدید با یک آدرس شبکهٔ متفاوت می‌تواند بر اساس این بلیت اعطاکنندهٔ بلیت صادر شود.
FORWARDED	نشان می‌دهد که این بلیت یا به جلورانده شده است و یا بر اساس یک اعتبارسنجی که شامل یک بلیت اعطاکنندهٔ بلیت به جلورانده بوده است صادر شده است.

ارسال کند که شامل یک مغشوش‌کنندهٔ تصادفی، شمارهٔ نسخه و یک برچسب زمانی بوده که با کلید مبتنی بر کلمهٔ عبور کاربر رمزنگاری شده باشد. AS بلوک را رمزگشائی کرده و یک بلیت اعطاکنندهٔ بلیت را پس نمی‌فرستد مگر اینکه برچسب زمانی موجود در بلوک پیش‌اعتبارسنج، در محدودهٔ مجاز زمانی باشد (محدوده‌ای که انحراف پالس ساعت و تأخیرهای شبکه را منظورکرده باشد). امکان دیگر استفاده از کارت هوشمندی است که مرتباً کلمات عبور متغیری را که در پیام‌های پیش-اعتبارسنجی شده وجود دارد تولید می‌کند. کلمات عبور تولیدشده بتوسط کارت می‌توانند مبتنی بر کلمهٔ عبور کاربر باشند ولی بتوسط کارت طوری تغییر یابند که در عمل کلمات عبور متفاوتی بکار رود. این امر از حمله‌ای که بخواهد بر اساس حدس‌زدن کلمات عبور ساده صورت پذیرد جلوگیری می‌نماید. اگر یک کارت هوشمند و یا دستگاه مشابهی بکارگرفته شود، این موضوع بتوسط پرچم PRE-AUTHENT نشان داده خواهد شد.

وقتی یک بلیت دارای طول عمرزیادی است، خطر سرقت و استفادهٔ غیرمجاز و طولانی از آن بتوسط دشمن زیاد است. اگر از طول عمر کمتری برای کاهش این تهدید استفاده شود، آنگاه افزایش سرباره بدلیل درخواست مکرر بلیت‌های جدید اجتناب‌ناپذیر است. در مورد یک بلیت اعطاکنندهٔ بلیت، کلاینت یا بایستی کلید سرّی کاربر را ذخیره نموده، که این کار دارای ریسک بالایی است، و یا بایستی مکرراً از کاربر درخواست کلمهٔ عبور نماید. یک روش بینابینی در این مورد استفاده از بلیت‌های قابل بروزرسانی است. یک بلیت دارای یک پرچم افراشتهٔ RENEWABLE شامل دو زمان انقضاء است: یکی برای این بلیت خاص و دیگری که آخرین زمان مجاز مربوط به انقضاء را نشان می‌دهد. یک کلاینت می‌تواند بلیت را به TGS

عرضه کرده و یک زمان انقضای جدید درخواست کند. اگر زمان جدید در محدودهٔ آخرین اندازهٔ مجاز قرار داشته باشد، TGS می‌تواند یک بلیت جدید با یک زمان اجلاس جدید و یک زمان انقضای مشخص صادر نماید. مزیت این مکانیسم این‌است که TGS ممکن است در صورتی که گزارشی از سرقت بلیت داشته باشد از صدور بلیت جدید امتناع ورزد.

یک کلاینت ممکن است تقاضا کند که AS یک بلیت اعطاکنندهٔ بلیت با پرچم افراشتهٔ MAY-POSTDATE برای او صادر نماید. کلاینت آنگاه می‌تواند این بلیت را برای درخواست یک بلیت که دارای پرچم‌های افراشتهٔ POSTDATED و INVALID هستند مورد استفاده قرار دهد. متعاقب آن، کلاینت ممکن است بلیت منقضی‌شده را برای تأئید ارائه کند. این روش ممکن است برای اجرای عملیات گروهی، روی یک سِرور که متناوباً نیاز به بلیت دارد مفید باشد. کلاینت می‌تواند برای این اجلاس تعدادی بلیت با زمان‌های متنوع را یکباره بدست آورد. تمام این بلیت‌ها بجز اولی در ابتدا فاقد اعتبارند. وقتی عملیات در زمان بجائی می‌رسد که به بلیت جدیدی نیاز است، کلاینت می‌تواند بلیت مناسب امر را دارای اعتبار نماید. با بکارگرفتن این روش، کلاینت مجبور نیست تا مکرراً بلیت اعطاکنندهٔ بلیت خود را بکار گرفته تا یک بلیت اعطاکنندهٔ سرویس بدست آورد.

در نسخهٔ ۵، این امکان وجود دارد که یک سِرور از جانب کلاینت بصورت یک پروکسی عمل نماید که اثر آن استفاده از اعتبار و امتیازات کلاینت برای درخواست سرویس از سِرور دیگر است. اگر یک کلاینت بخواهد از این مکانیسم استفاده نماید، درخواست یک بلیت اعطاکنندهٔ بلیت با پرچم افراشتهٔ PROXIABLE می‌نماید. وقتی این بلیت به TGS عرضه می‌شود، TGS مجاز به صدور یک بلیت اعطاکنندهٔ بلیت با آدرس شبکهٔ دیگری خواهد بود. این بلیت آخر، دارای پرچم PROXY افراشته خواهد بود. کاربری که چنین بلیتی را دریافت میکند، ممکن است آن را پذیرفته و یا نیاز به اعتبارسنجی اضافی برای فراهم‌آوردن یک ردّپای ممیزی داشته باشد.

مقولهٔ پروکسی یک مورد محدود از روش عام‌تر و قوی‌تر به‌جلوراندن است. اگر پرچم FORWARDABLE در یک بلیت افراشته باشد، آنگاه یک TGS می‌تواند برای متقاضی، یک بلیت اعطاکنندهٔ بلیت با یک آدرس شبکهٔ متفاوت صادر کند که پرچم FORWARDED آن افراشته باشد. این بلیت می‌تواند به یک TGS دور عرضه شود. این توانائی به کلاینت اجازه می‌دهد تا به یک سِرور در قلمرو دیگر دست یابد، بدون اینکه نیاز باشد که هر Kerberos یک کلید سرّی با هر Kerberos دیگر در سایر قلمروها را به اشتراک بگذارد. بعنوان مثال، قلمروها می‌توانند دارای ساختار درختی باشند. در اینصورت یک کلاینت می‌تواند یک شاخه درخت را تا یک گرهٔ مشترک بالا رفته و سپس برای دسترسی به قلمرو هدف از شاخهٔ دیگر پائین آید. هرقدم این راه‌پیمائی شامل به‌جلوراندن یک بلیت اعطاکنندهٔ بلیت به TGS بعدی مسیر است.

۴-۲ سرویس اعتبارسنجی X.509

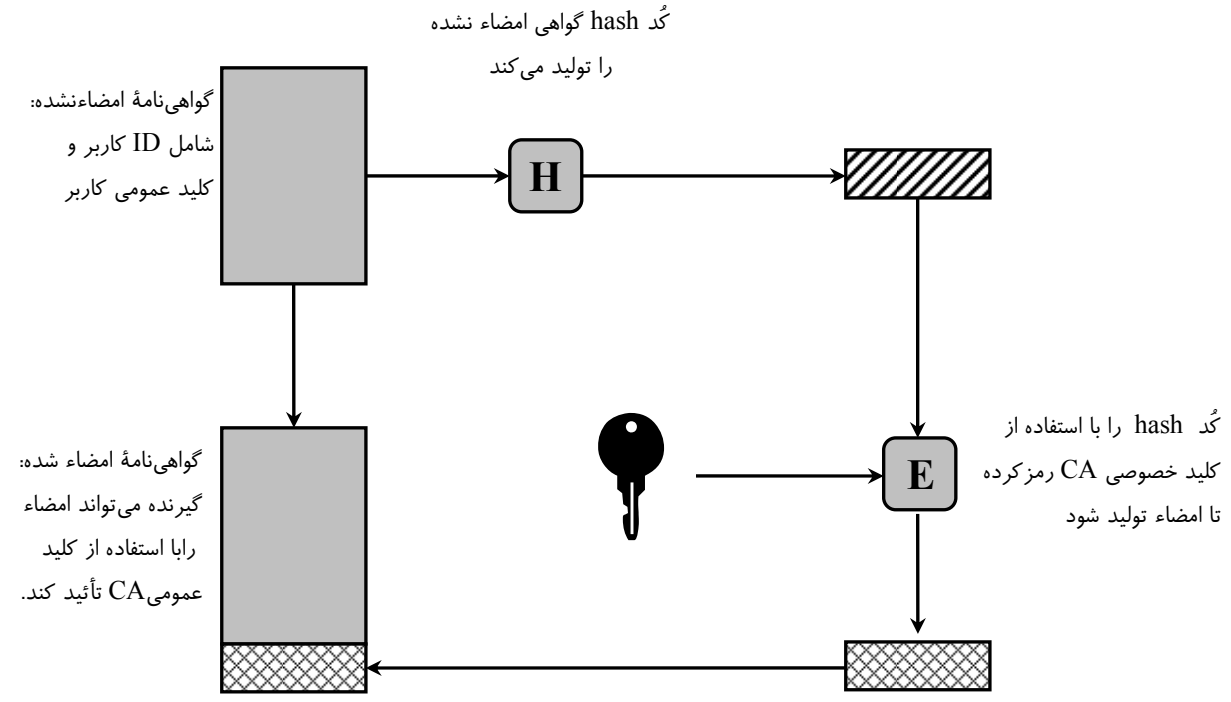
توصیه‌نامهٔ X.509 که مربوط به ITU-T است، بخشی از سری توصیه‌نامه‌های X.500 است که یک سرویس فهرست راهنما را تعریف می‌کند. فهرست راهنما در واقع یک سِرور و یا مجموعه‌ای توزیع‌شده از سِرورهاست که یک پایگاه اطلاعاتی داده در موردکاربران را نگهداری می‌کند. اطلاعات شامل یک نگاشت از نام کاربر به آدرس شبکه و همچنین سایر صفات و اطلاعات مربوط به کاربر است.

X.509 یک محدودهٔ کاری برای فراهم‌آوردن سرویس‌های اعتبارسنجی بتوسط فهرست راهنمای X.500 برای کاربران خود فراهم می‌سازد. فهرست راهنما ممکن است همانند یک صندوقچه نگهداری گواهی‌نامه‌های کلید- عمومی عمل نماید. هر گواهی شامل کلید عمومی یک کاربر بوده و بتوسط کلید خصوصی یک مسئول قابل اعتماد صدور گواهی‌نامه امضاء می‌گردد. علاوه‌براین، X.509 پروتکل‌های اعتبارسنجی دیگری که مبتنی بر استفاده از گواهی‌نامه‌های کلید- عمومی هستند را تعریف می‌نماید.

X.509 یک استاندارد مهم است زیرا ساختار گواهی‌نامه و پروتکل‌های اعتبارسنجی تعریف‌شده در X.509 در مقوله‌های متعددی مورد استفاده قرار می‌گیرند. مثلاً فرمت گواهی X.509 در S/MIME (فصل پنجم)، امنیتIP (فصل ششم)، و SSL/TLS و SET (فصل هفتم) بکار می‌روند.

X.509 بدواً در سال ۱۹۸۸ منتشر گردید. متعاقباً تغییراتی در این استاندارد داده شد تا برخی نگرانی‌های امنیتی ذکرشده در [IANS90] و [MITC90] مورد توجه قرار گیرند. توصیه‌نامهٔ اصلاح‌شده در سال ۱۹۹۳ مستند گردید. نسخهٔ سوم آن در سال ۱۹۹۵ انتشار یافت و در سال ۲۰۰۰ مورد بازنگری قرار گرفت.

X.509 بر مبنای استفاده از رمزنگاری کلید- عمومی و امضاءهای دیجیتال قرار گرفته است. استاندارد، استفاده از یک الگوریتم خاص را اجباری نمی‌سازد ولی RSA را توصیه می‌کند. فرض شده است که تکنیک امضاء دیجیتال نیاز به استفاده از یک تابع درهم‌ساز دارد. بازهم استاندارد الگوریتم درهم‌سازی خاصی را پیشنهاد نمی‌کند. توصیه‌نامهٔ ۱۹۸۸ شامل توصیف یک الگوریتم درهم‌سازی توصیه‌شده بود که بعداً مشخص گردید که ناامن بوده و بنابراین از توصیه‌نامهٔ سال ۱۹۹۳ حذف گردید. شکل ۳-۴ نحوهٔ تولید یک گواهی‌نامهٔ کلید- عمومی را نشان می‌دهد.



شکل ۳-۴ استفاده از گواهی‌نامهٔ کلید- عمومی

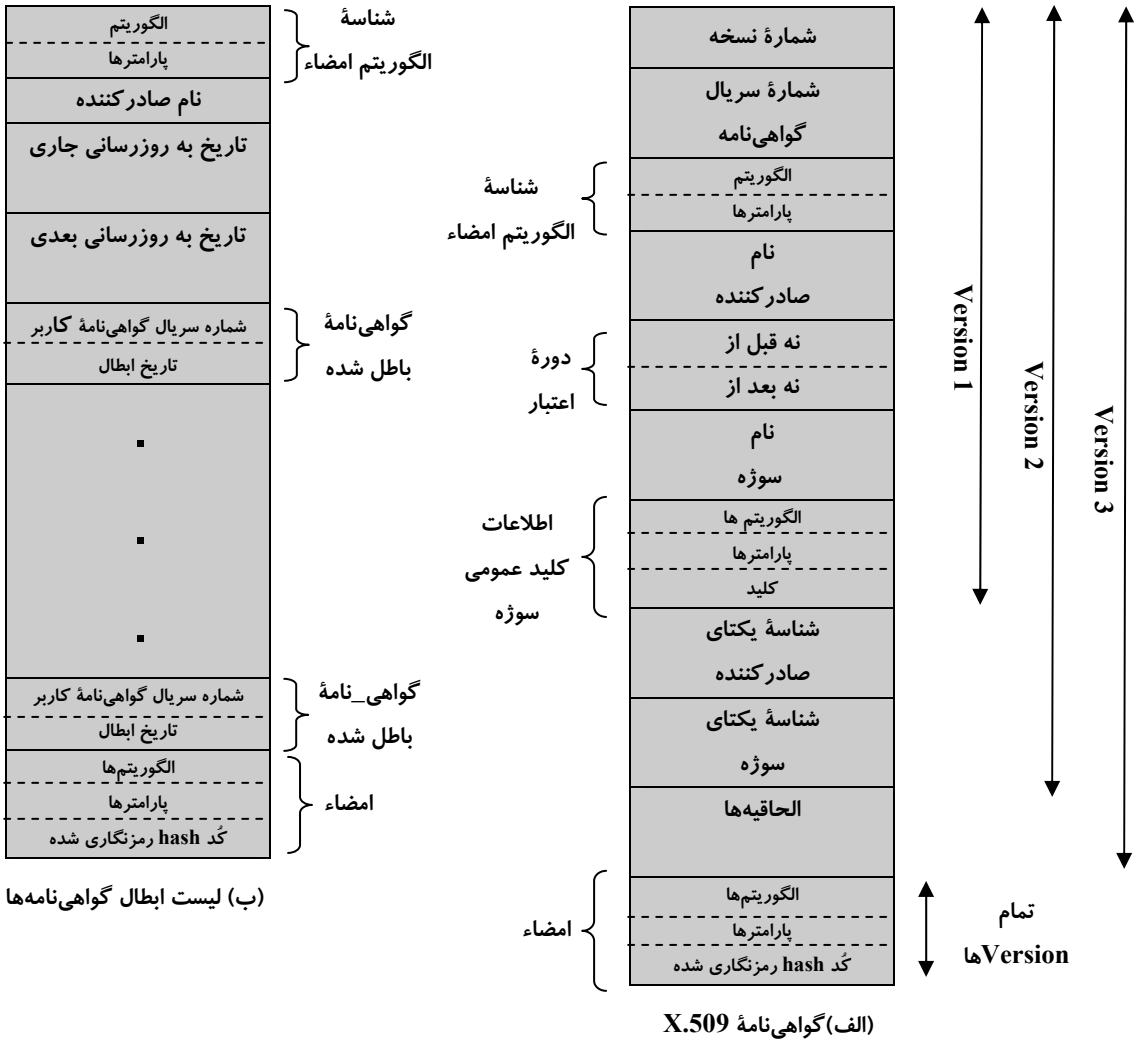
گواهی‌نامه‌ها (Certificates)

قلب روش X.509، گواهی‌نامهٔ کلید- عمومی مرتبط با هر کاربر است. فرض براین است که گواهی‌نامه‌های کاربران بتوسط یک مسئول قابل اعتماد صدور گواهی‌نامه Certification Authority (CA) صادر شده و بتوسط CA یا کاربر در فهرست راهنما درج شده است. سِرور مخصوص فهرست راهنما، خود مسئول تولید کلیدهای عمومی و یا عملیات صدور گواهی نبوده و صرفاً محل قابل دسترسی ساده‌ای است که گواهی‌نامه‌های درخواستی کاربران را در اختیار آنها می‌گذارد.

شکل ۴-۴الف فرمت عمومی یک گواهی‌نامه را نشان میدهد که شامل عناصر زیر است:

- **شمارهٔ نسخه**: بین نسخه‌های مختلف انتشاریافتهٔ فرمت گواهی‌نامه فرق می‌گذارد. پیش‌فرض آن Version 1 است. اگر شناسهٔ یکتای صادرکننده و یا شناسهٔ یکتای سوژه در فرمت حضور داشته باشند، اندازهٔ این میدان بایستی Version 2 باشد. اگر یکی یا بیشتر از الحاقیه‌ها موجود باشند، نسخه بایستی Version 3 باشد.
- **شمارهٔ سریال**: یک عدد صحیح و یکتا در نزد صادرکنندهٔ CA است که بدون هیچگونه ابهامی فقط مرتبط با این گواهی‌نامه است.
- **شناسهٔ الگوریتم امضاء**: الگوریتم و پارامترهای مربوطی است که برای امضاء این گواهی‌نامه بکارگرفته شده است. چون این اطلاعات در میدان امضاء واقع در انتهای این گواهی دوباره تکرار می‌شود، این میدان استفادهٔ کمی دارد.
- **نام صادرکننده**: نام CA صادرکننده و امضاءکنندهٔ این گواهی‌نامه در فرمت X.500 .
- **دورهٔ اعتبار**: شامل دو تاریخ است: تاریخ اول و تاریخ آخری که این گواهی‌نامه بین این دو تاریخ اعتبار دارد.
- **نام سوژه**: نام کاربری که این گواهی‌نامه مربوط به اوست. یعنی این گواهی، کلید عمومی سوژه‌ای را که کلید خصوصی مرتبط را در اختیار دارد تأئید می‌نماید.
- **اطلاعات کلید عمومی سوژه**: کلید عمومی سوژه بعلاوهٔ یک شناسهٔ مرتبط با الگوریتمی که این کلید برای آن بکار خواهد رفت، به همراه پارامترهای مربوطه.
- **شناسهٔ یکتای صادرکننده**: یک میدان از دنبالهٔ بیت‌ها که برای شناسائی یکتای CA صادرکننده بکار میرود، در صورتی‌که نام X.500 برای واحدهای مختلف مکرراً استفاده شده باشد.
- **شناسهٔ یکتای سوژه**: یک میدان از دنبالهٔ بیت‌ها که برای شناسائی یکتای سوژه بکار میرود، در صورتی‌که نام X.500 برای واحدهای مختلف مکرراً استفاده شده باشد.
- **الحاقیه‌ها**: مجموعه‌ای از یک یا چند میدان الحاق شده. الحاقیهٔ‌ها در نسخهٔ ۳ به توصیه‌نامه اضافه شده و بعداً در همین بخش توصیف خواهند شد.
- **امضاء**: همهٔ میدان‌های دیگر گواهی‌نامه را پوشش می‌دهد. این شامل کُد hash سایر میدان‌ها بوده که بتوسط کلید خصوصی CA رمزنگاری می‌شود. این میدان شامل شناسهٔ الگوریتم امضاء است.

میدان‌های یکتای شناسه‌ها در نسخهٔ ۲ اضافه شد تا مشکلات احتمالی مربوط به استفادهٔ مجدد از سوژه و یا نام‌های صادرکنندگان گواهی‌نامه در طول زمان را حل کند. این میدان‌ها بندرت مورد استفاده قرار می‌گیرند.



شکل ۴-۴ فرمت‌های X.509

استاندارد از فرم زیر برای تعریف یک گواهی استفاده می‌کند:

$$CA\langle\langle A\rangle\rangle = CA\{V, SN, AI, CA, T_A, A, Ap\}$$

که درآن

$\langle\langle X\rangle\rangle Y$ = گواهی‌نامهٔ کاربر X که بتوسط مسئول صدور گواهی Y صادر شده است.

$Y\{I\}$ = امضاء I بتوسط Y. این شامل I و کُد hash رمزنگاری شدهٔ آن است که به آن وصل شده است.

CA گواهی‌نامه را با کلید خصوصی خود امضاء می‌کند. اگر کلید عمومی مرتبط با آن برای یک کاربر شناخته شده باشد، آنگاه کاربر میتواند تأئید کند که گواهی امضاءشده بتوسط CA معتبر است. این یک روش مرسوم برای امضاء دیجیتال است که در شکل ۲-۳ب نشان داده شده است.

اخذ گواهی‌نامه یک کاربر

گواهی‌نامه‌هائیکه بتوسط CA برای کاربران صادر می‌شود، دارای مشخصات زیر است:

- هرکاربری که به کلید عمومی CA دسترسی داشته باشد می‌تواند کلید عمومی گواهی‌شدهٔ کاربر را تأئید نماید.
- هیچکس بغیر از مسئول صدور گواهی نمی‌تواند بدون اینکه مچش گیرافتد، گواهی‌نامه را دستکاری نماید.

نظر به اینکه گواهی‌نامه‌ها غیرقابل جعل‌اند، می‌توان آنها را در یک فهرست راهنما قرار داد بدون اینکه نیازی به حفاظت از آنها باشد.

اگر همهٔ کاربران، مشترکین فقط یک CA باشند، آنگاه اعتماد مشترکی نسبت به آن CA در همهٔ آنها وجود دارد. تمام گواهی‌های کاربران را می‌توان در فهرست راهنمائی قرار داد که بتوسط همهٔ آنها قابل دسترس باشد. علاوه بر آن یک کاربر می‌تواندگواهی‌نامهٔ خود را مستقیماً به کاربر دیگری منتقل نماید. در هریک از دو مورد، همین‌که B گواهی A را در اختیار خود گرفت، B اطمینان دارد که پیام‌هائی که او با کلید عمومی A رمزنگاری می‌کند نسبت به شنود امن بوده و پیام‌هائی که با کلید خصوصی A امضاءشده باشند غیرقابل جعل‌اند.

اگر تعداد کاربران خیلی زیاد باشد، عملی نخواهد بود که تمام آنها مشترک یک CA خاص باشند. چون این CA است که گواهی‌ها را امضاء می‌کند، هرکاربر مشترک CA بایستی یک کپی از کلید عمومی CA را در اختیار داشته باشد تا بتوسط آن بتواند گواهی‌ها را تأئید نماید. این کلید عمومی بایستی بصورت کاملاً امنی(از نظر اصالت و اعتبار) در اختیار هر کاربر قرار گیرد تا کاربر نسبت به گواهی‌های صادره اطمینان داشته باشد. بنابراین با تعداد کاربران زیاد، ممکن است راه حل عملی‌تر این باشد که تعدادی CA وجود داشته باشند که هریک از آنها کلید عمومی خود را بطور امنی در اختیار بخشی از کاربران قرار دهند.

حال فرض کنید که A یک گواهی‌نامه از مسئول صدور گواهی X_1 اخذ نموده و B نیز یک گواهی‌نامه از CA خود یعنی X_2 اخذ کرده باشد. اگرA بطور امنی از کلید عمومی X_2 خبر نداشته باشد، آنگاه گواهی‌نامهٔ B که بتوسط X_2 صادرشده است برای A بی‌ارزش خواهد بود. ولی اگر دو CA کلیدهای عمومی خود را بطور امنی مبادله کرده باشند، آنگاه روش زیر A را قادر خواهد ساخت تا کلید عمومی B را بدست آورد.

۱- A از فهرست راهنما، گواهی‌نامهٔ X_2 امضاء شده بتوسط X_1 را می‌گیرد. چون A بطور امنی از کلید عمومی X_1 باخبر است، A میتواند کلید عمومی X_2 را از گواهی‌نامهٔ او بدست آورده و آن را بتوسط امضاء X_1 که روی گواهی‌نامه است تأئید نماید.

۲- A سپس به فهرست راهنما برگشته و گواهی‌نامه B را که بتوسط X_2 امضاء شده است دریافت می‌کند. حال چون A یک کپی مطمئن از کلید عمومی X_2 را در اختیار دارد، A میتواند امضاء را تأئید کرده و بطور امنی کلیدعمومیB را استخراج نماید.

A برای بدست آوردن کلید عمومی B از زنجیره‌ای از گواهی‌نامه‌ها استفاده کرده است. برحسب علائم X.509 این زنجیره چنین بیان می‌شود:

$$X_1<<X_2>>X_2<<B>>$$

بهمین روش B میتواند کلید عمومی A را با زنجیرهٔ معکوس بدست آورد.

$$X_2<<X_1>>X_1<<A>>$$

این روش لازم نیست که فقط محدود به دو گواهی‌نامه باشد. یک مسیر طولانی از CAها را می‌توان برای ایجاد یک زنجیر در نظر گرفت. یک زنجیر با N عنصر چنین بیان می‌گردد:

$$X_1<<X_2>>X_2<<X_3>> \cdots X_N<<B>>$$

در این مورد، هرزوج از CAها در زنجیرهٔ (X_i , X_{i+1}) بایستی برای یکدیگر گواهی‌نامه صادر کرده باشند.

تمام این گواهی‌نامه‌های CAها بتوسط CAهای دیگر لازم است که در فهرست راهنما وجود داشته باشند و کاربر لازم است بداندکه اینها چگونه با هم مرتبط‌اند تا بتواند مسیر را تا یافتن کلید عمومی B دنبال نماید. X.509 پیشنهاد می‌کند که CAها در یک ساختار سلسله‌مراتبی طوری سازمان‌دهی شوند که ناوبری بین آنها ساده باشد.

شکل ۵-۴ که از X.509 گرفته شده است، مثالی از چنین ساختاری است. دایره‌های بهم وصل شده، ارتباط سلسله‌مراتبی CAها را نشان می‌دهد و مربع‌های مربوطه نمایشگر گواهی‌نامه‌های نگهداری شده در فهرست راهنمای هر CAاند. اقلام فهرست راهنما در هر CA شامل دو نوع گواهی‌اند:

- **گواهی‌های مستقیم**: گواهی‌های X که بتوسط سایر CAها صادر شده‌اند.
- **گواهی‌های معکوس**: گواهی‌های تولیدشده بتوسط X که گواهی‌نامهٔ سایر CAها هستند.

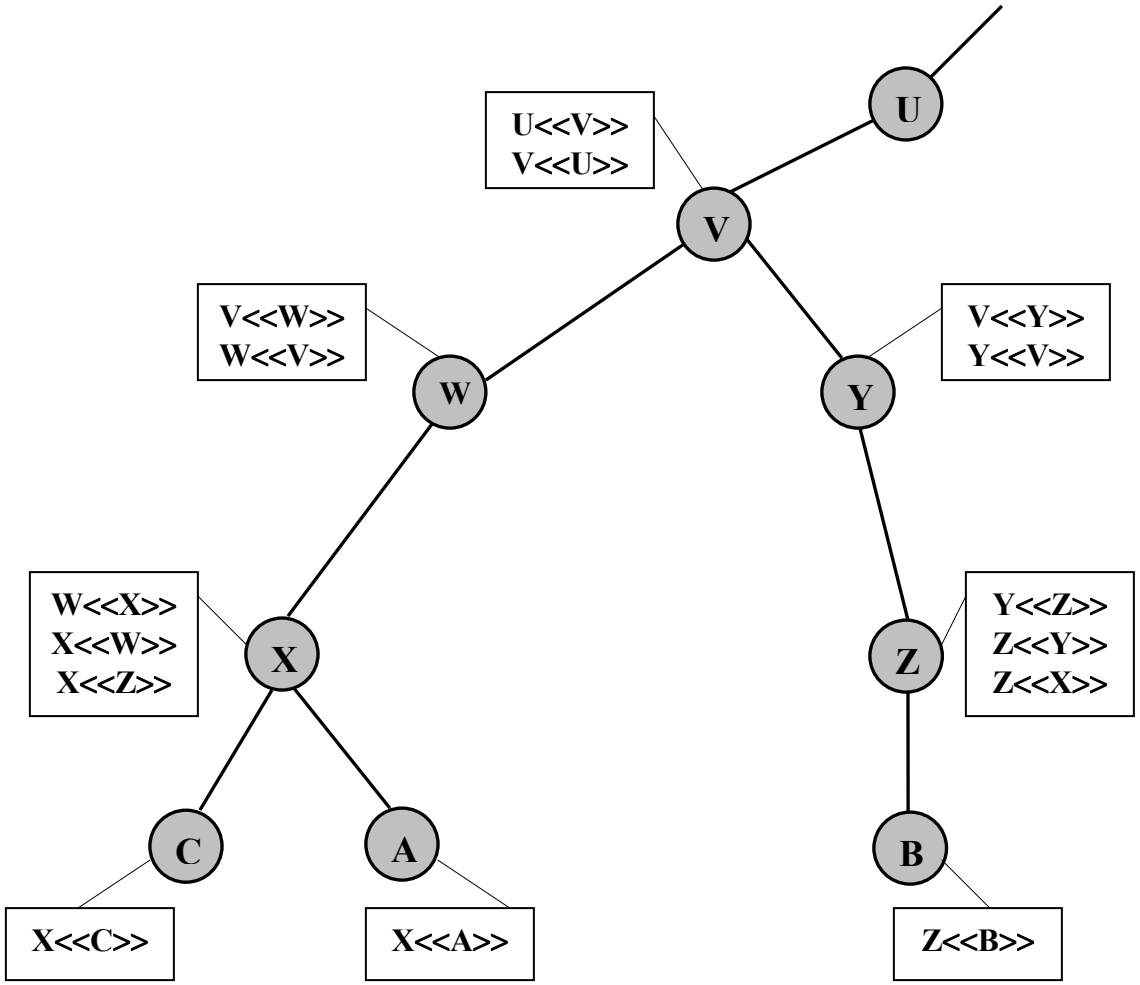
در این مثال، کاربرA میتواند گواهی‌های زیر را از فهرست راهنما بدست آورده و مسیر را تا B دنبال کند:

$$X<<W>>W<<V>>V<<Y>>Y<<Z>>Z<<B>>$$

وقتی A این گواهی‌ها را بدست آورد، می‌تواند مسیر را بترتیب دنبال کرده تا یک کپی مورد اطمینان از کلید عمومی B را بدست آورد. با استفاده از این کلید عمومی، A می‌تواند پیام‌های رمزنگاری‌شدهٔ خود را برای B بفرستد. اگر A تمایل داشته باشد تا پیام‌های رمزنگاری‌شده‌ای را از طرف B دریافت کند و یا پیام‌هائی را که برای B ارسال میشود امضاء نماید، آنگاه B نیاز به کلید عمومی A خواهد داشت که میتواند آن را از مسیر زیر بدست آورد:

$$Z<<Y>>Y<<V>>V<<W>>W<<X>>X<<A>>$$

B می‌تواند این مجموعهٔ گواهی‌ها را از فهرست راهنما استخراج کرده و یا A می‌تواند آنها را بصورت بخشی از پیام اولیه برای B ارسال دارد.



شکل ۵-۴ سلسله مراتب X.509 : یک مثال فرضی

ابطال گواهی‌نامه‌ها

از شکل ۴-۴ بخاطر آورید که هر گواهی‌نامه همانند یک کارت اعتباری دارای یک دورهٔ اعتبار است. معمولاً بایستی قبل از انقضای دورهٔ اعتبار، گواهی‌نامهٔ جدیدی صادر شود. علاوه براین ممکن است در مواردی علاقه‌مند باشیم تا به دلایل زیر یک گواهی‌نامه را قبل از انقضای مهلت آن باطل کنیم:

۱- احتمال داده شود که کلید خصوصی کاربر لو رفته است.
۲- کاربر، دیگر بتوسط این CA تأئید نمی‌شود.
۳- احتمال داده شود که گواهی‌نامهٔ CA لو رفته است.

هر CA بایستی لیستی که شامل تمام گواهی‌های باطل شده ولی منقضی نشده صادره بتوسط آن CA، اعم از گواهی‌های صادرشده برای کاربران، یا سایر CAها را نگهداری نماید. این لیست‌ها همچنین بایستی در فهرست راهنما اعلان شود.

هر لیست گواهی‌های باطل شده certificate revocation list (CRL)،که در فهرست راهنما اعلان می‌گردد بتوسط صادرکننده امضاءشده و شامل (شکل ۴-۴ب) نام صادرکننده، تاریخی که لیست تولید شده است، تاریخی که CRL بعدی قرار است منتشر شود و یک ورودی برای هر گواهی باطل شده است. هر ورودی شامل شمارهٔ سریال یک گواهی‌نامه و تاریخ ابطال آن گواهی‌نامه است. چون شمارهٔ سریال‌های مربوط به یک CA یکتا هستند، شمارهٔ سریال برای شناسائی یک گواهی‌نامه کافی است.

وقتی یک کاربر یک گواهی‌نامه در یک پیام را دریافت کرد، او بایستی تحقیق کند که گواهی‌نامه باطل نشده باشد. کاربر می‌تواند هربار که یک گواهی‌نامه دریافت می‌کند، فهرست راهنما را کنترل نماید. بـــرای جلوگیری از تأخیر زمـــانی (و احتمالاً هزینه) مرتبط با جستجوی فهرست راهنما، محتمل است که کاربر یک حافظهٔ محلی را برای نگهداری گواهی‌نامه‌ها و لیست گواهی‌نامه‌های باطل‌شده در اختیار داشته باشد.

رَویه‌های اعتبارسنجی

X.509 همچنین شامل سه رَویهٔ مختلف اعتبارسنجی است که برای استفاده در کاربردهای متنوعی طراحی شده‌اند. تمام این رَویه‌ها از امضاءهای کلید- عمومی استفاده می‌کنند. فرض براین است که دو طرف کلید عمومی یکدیگر را می‌دانند که این امر یا با کسب گواهی‌نامه‌های یکدیگر از فهرست راهنما و یا بدلیل اینکه گواهی‌نامه در پیام اولیه هرطرف وجود داشته است امکان‌پذیر خواهد بود.

شکل ۶-۴ این سه رَویه را نشان میدهد.

اعتبارسنجی یک- سویه

اعتبارسنجی یک- سویه شامل یک بار انتقال اطلاعات از سوی کاربر(A) برای کاربر(B) است و موارد زیر را روشن می‌سازد:

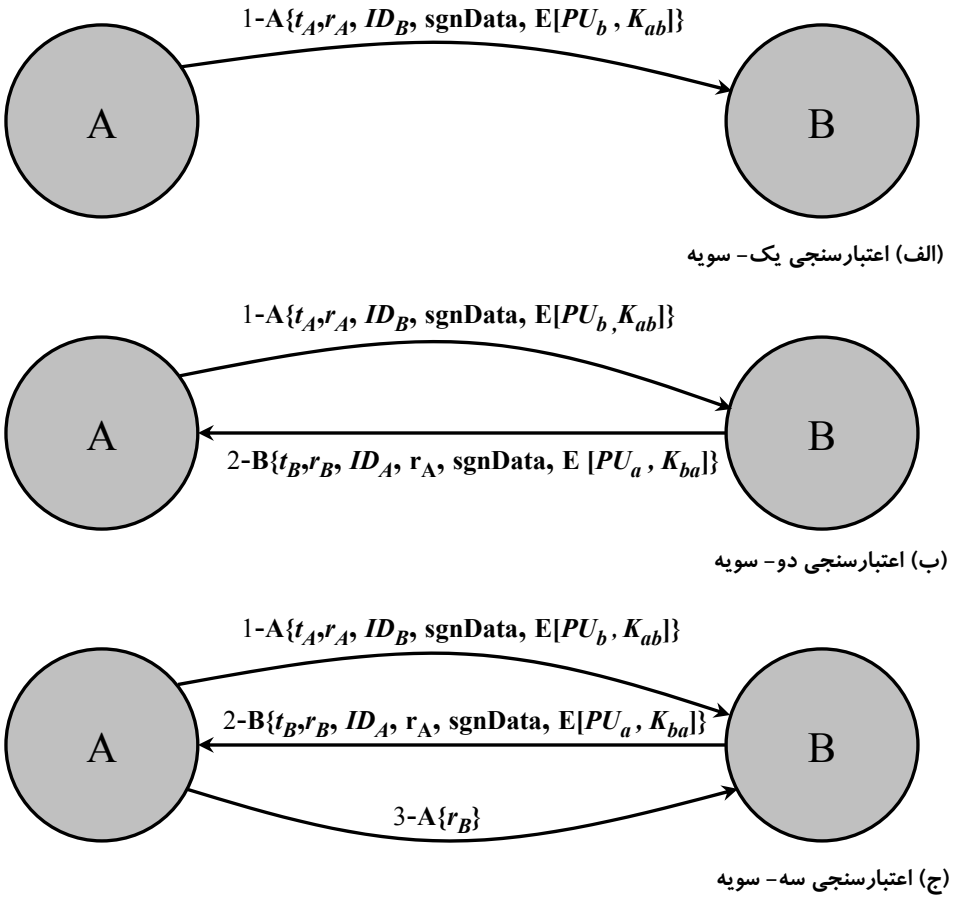
۱- هویت A و پیامی که بتوسط A تولیدشده است.

۲- اینکه پیام به مقصدB است.

۳- صحت و دست اول بودن پیام (اینکه چندبار ارسال نشده باشد).

توجه کنید که در این روش تنها هویت واحد شروع‌کنندهٔ ارتباط، و نه واحد پاسخ‌دهنده به ارتباط، تأئید می‌شود.

پیام حداقل دارای یک برچسب زمانی t_A، یک nonce (r_A) و هویت B بوده و بتوسط کلید خصوصی A امضاء می‌شود. برچسب زمانی شامل یک بخش اختیاری زمان شروع و زمان خاتمه است. این امر از تأخیر در تسلیم پیام جلوگیری می‌کند. nonce می‌تواند برای تشخیص حملات بازخوانی بکار رود. اندازهٔ nonce بایستی در طول زمان اعتبار پیام، یکتا باشد. بنابراینB می‌تواند nonce را تا زمان انقضای آن ذخیره کرده و هر پیام جدید با همان nonce را نپذیرد.



شکل ۶-۴ رَویهٔ های قدرتمند اعتبارسنجی X.509

برای اعتبارسنجی صِرف، پیام بسادگی برای ارائهٔ امتیازات به B بکار می‌رود. پیام همچنین ممکن است شامل اطلاعاتی باشد که بایستی عرضه شود. این اطلاعات، SgnData، در محدودهٔ امضاء بوده و اعتبار و اصالت آن را تضمین می‌نماید. پیام همچنین ممکن است برای رساندن یک کلید اجلاس به B بکار رود و با کلید عمومی B رمزشده باشد.

اعتبارسنجی دو- سویه

علاوه بر سه موردی که در بالا ذکرشد، اعتبارسنجی دو- سویه مقوله‌های اضافی زیر را نیز مورد توجه قرار می‌دهد:

۴- هویت B و اینکه پاسخ پیام از سوی B، واقعاً بتوسط خود B تولیدشده است.

۵- اینکه پیام به مقصد A ارسال شده است.

۶- صحت و دست اول بودن پیام

بنابراین اعتبارسنجی دو- سویه به هردو طرف درگیر اجازه می‌دهد تا هویت طرف مقابل را تأئید نمایند.

پیام پاسخ، شامل nonce از طرف A برای اعتباربخشیدن به پاسخ است. پیام همچنین شامل یک برچسب زمانی و nonce تولیدشده بتوسط B است. همانند قبل، پیام ممکن است شامل اطلاعات اضافی امضاءشده و یک کلید اجلاس باشد که بتوسط کلید عمومی A رمزنگاری شده باشد.

اعتبارسنجی سه- سویه

در اعتبارسنجی سه- سویه، یک پیام نهائی از A به B نیز وجود دارد که شامل یک کپی امضاءشدهٔ nonce (r_B) است. هدف این طراحی این است که برچسب‌های زمانی نیازی به کنترل نداشته باشند. چون هر دو nonce بتوسط طرف مقابل برگشت داده میشوند، هرطرف میتواند nonce برگشتی را کنترل کرده و حملات احتمالی بازخوانی را تشخیص دهد. این تکنیک وقتی پالس‌های ساعت همزمان در دسترس نباشند مورد لزوم است.

نسخهٔ سوم X.509

فرمت نسخهٔ دوم X.509 تمام اطلاعاتی که طراحی‌ها و تجارب پیاده‌سازی اخیر نیاز آن را ثابت کرده است، منتقل نمی‌کند. [FORD95] الزامات زیرکه در نسخهٔ دوم برآورده نشده است، را بیان کرده است:

۱- میدان سوژه برای معرفی هویت یک صاحب کلید به یک استفاده‌کننده از کلید عمومی کافی نیست. نام‌های X.509 ممکن است نسبتاً کوتاه بوده و فاقد جزئیات هویتی لازم مورد نیاز کاربر باشند.

۲- میدان سوژه همچنین برای برخی کاربردها که معمولاً واحدها را با آدرس e-mail، یک URL و یا بطریق دیگری در اینترنت شناسائی می‌کنند، ناکافی است.

۳- این نیاز وجود داردکه اطلاعات مربوط به خط‌مشی امنیتی نشان داده شود. این امر یک کاربرد امنیتی یا عمل امنیتی همچون IPSec را قادر می‌سازد تا یک گواهی X.509 را به یک خط‌مشی خاص ربط دهد.

۴- این نیاز وجود دارد تا صدماتی که ممکن است از یک CA معیوب و یا بداندیش ناشی شود را با ایجاد محدودیت‌هائی در کاربرد یک گواهی‌نامهٔ خاص کم کرد.

۵- این که بتوان کلیدهای مختلف مجزا که بتوسط صاحب آن در زمان‌های مختلف مورد استفاده قرار گرفته است را شناسائی نمود، امری مهم است. این خصیصه، مدیریت طول عمر کلید را حمایت کرده و علی‌الخصوص این توانائی که بتوان جفت کلیدها را برای کاربران و CAها در فواصل زمانی منظم و یا تحت شرایط استثنائی به روز درآورد را بوجود می‌آورد.

بجای اینکه مرتباً میدان‌های جدیدی به فرمت ثابت اضافه نمایند، تولیدکنندگان استانداردها احساس کرده‌اند که روش انعطاف‌پذیرتری مورد نیاز است. بنابراین نسخهٔ ۳ شامل تعدادی الحاقیهٔ اختیاری است که ممکن است به فرمت نسخهٔ ۲ اضافه کرد. هریک از این الحاقیه‌ها شامل یک شناسهٔ الحاقیه، یک نمایشگر اهمیت الحاقیه و یک اندازه الحاقیه است. نمایشگر اهمیت الحاقیه بیانگر این مسأله است که آیا می‌توان با خاطری آسوده از یک الحاقیه صرفنظر کرد؟ اگر نمایشگر دارای اندازهٔ true باشد و واحد اجرا آن را نشناسد، گواهی‌نامه غیرقابل قبول تلقی خواهد شد.

الحاقیه‌های گواهی‌نامه‌ها در سه گروه اصلی قرار می‌گیرند: اطلاعات کلید و خط‌مشی، مشخصات سوژه و صادرکنندهٔ گواهی، و محدودیت‌های روند گواهی کردن.

اطلاعات کلید و خط‌مشی

این الحاقیه‌ها، اطلاعات اضافی مربوط به کلید سوژه و کلید صادرکنندهٔ گواهی‌نامه را حمل کرده و بعلاوه نمایش‌دهندهٔ خط‌مشی گواهی‌نامه می‌باشند. خط‌مشی یک گواهی‌نامه شامل مجموعه‌ای از قوانین تبیین شده بوده که نمایش‌دهندهٔ قابلیت کاربرد یک گواهی‌نامه در یک جمعیت خاص و یا دسته‌ای از کاربردها با نیازهای امنیتی مشترک می‌باشند. بعنوان مثال یک نوع خط‌مشی ممکن است قابل اعمال به اعتبارسنجی اسناد مالی EDI (Electronic Data Interchange) برای تجارت اقلام در محدودهٔ قیمت مشخصی باشد.

این ناحیه شامل اقلام زیر است:

- **شناسهٔ کلید CA**: هویت کلید عمومی که بایستی برای تأئید امضاء در این گواهی‌نامه و یا CRL بکار رود را تعیین می‌کند. باعث می‌شود که بتوان بین کلیدهای متفاوت یک CA فرق گذاشت. یکی از موارد استعمال این میدان به‌روز درآوردن زوج کلید CA است.
- **شناسهٔ کلید سوژه**: هویت کلید عمومی که گواهی‌نامهٔ آن در شرف صادرشدن است را تعیین می‌کند. برای به روز رساندن کلید سوژه مفید است. همچنین یک سوژه ممکن است جفت کلیدهای متعدد و نظیرآن گواهی‌های مختلف برای اهداف متفاوت (مثل امضاء دیجیتال و موافقت در مورد کلید رمزنگاری) داشته باشد.
- **موارد استعمال کلید**: محدودیت‌های اعمال شده، نظیر اهدافی که برای آن و سیاست‌هائی که تحت آن کلید عمومی گواهی‌شده می‌تواند مورد استفاده قرار گیرد را نشان می‌دهد و ممکن است نشان‌دهندهٔ یک یا چند مورد زیر باشد: امضاء دیجیتال، عدم انکار، رمزنگاری دیتا، توافق نسبت به کلید، تأئید امضاء CA در گواهی‌نامه‌ها، تأئید امضاء CA در CRLها.
- **دورهٔ استفاده از کلید خصوصی**: مدت زمان قابل استفاده از کلید خصوصی نظیر یک کلید عمومی را تعیین می‌کند. معمولاً کلید خصوصی در طول دوره‌های مختلفی از دورهٔ اعتبار کلید عمومی استفاده می‌شود. مثلاً در رابطه با کلیدهای امضاء دیجیتال، زمان قابل استفاده از کلید خصوصی معمولاً کوتاهتر از دورهٔ تأئید کلید عمومی است.
- **خط‌مشی‌های گواهی‌نامه**: از گواهی‌نامه‌ها ممکن است در محیط‌هائی استفاده شود که خط‌مشی‌های مختلفی حاکم است. این الحاقیه، خط‌مشی‌هائی که گواهی‌نامه آنها را حمایت می‌کند را لیست می‌کند.
- **نگاشت خط‌مشی‌ها**: فقط در گواهی‌نامه‌هائی که برای یک CA از سوی CA دیگر صادر می‌شود کاربرد دارد. این الحاقیه اجازه می‌دهد تا یک CA نشان دهد که یک یا چند مورد از خط‌مشی‌های آن می‌تواند معادل خط‌مشی دیگر در حوزهٔ CA سوژه باشد..

مشخصات سوژه و صادرکنندهٔ گواهی‌نامه

این الحاقیه‌ها از نام‌های مختلف با فرمت‌های مختلف برای سوژهٔ یک گواهی‌نامه یا صادرکنندهٔ یک گواهی‌نامه حمایت کرده و می‌توانند اطلاعات بیشتری در مورد سوژهٔ گواهی‌نامه را انتقال داده تا استفاده‌کننده از گواهی‌نامه اعتماد بیشتری نسبت به یک شخص یا واحد خاص پیدا کند. نمونه‌هائی از این اطلاعات، آدرس پستی، مسئولیت سازمانی فرد و یا تصویر اوست.

میدان‌های الحاقی در این مورد چنین‌اند:

- **نام‌های دیگر سوژه**: شامل یک یا چندنام جایگزین است که می‌توانند فرم‌های مختلفی داشته باشند. این میدان برای حمایت از کاربردهای مختلفی مثل پست الکترونیک، EDI، و IPSec که ممکن است فرم نام‌هایشان متفاوت باشند اهمیت دارد.
- **نام‌های دیگر صادرکننده**: شامل یک یا چند نام جایگزین است که می‌توانند فرم‌های مختلفی داشته باشند.
- **مشخصات سوژه در فهرست راهنما**: اندازه‌های مطلوب مشخصه‌های فهرست راهنمای X.500 که مربوط به سوژهٔ این گواهی‌نامه است را نشان می‌دهد.

محدودیت‌های رَوَند گواهی کردن

این الحاقیه‌ها اجازه می‌دهند تا انواع قیود در گواهی‌نامه‌های صادرشده از طرف یک CA برای CA دیگر درج گردد. این محدودیت‌ها ممکن است نوع گواهی‌هائی را که می‌تواند از طرف CA سوژه صادر گردد و یا آنچه را که متعاقباً در زنجیرهٔ گواهی‌ها رخ میدهد مقید سازد.

میدان‌های الحاقی در این مورد چنین‌اند:

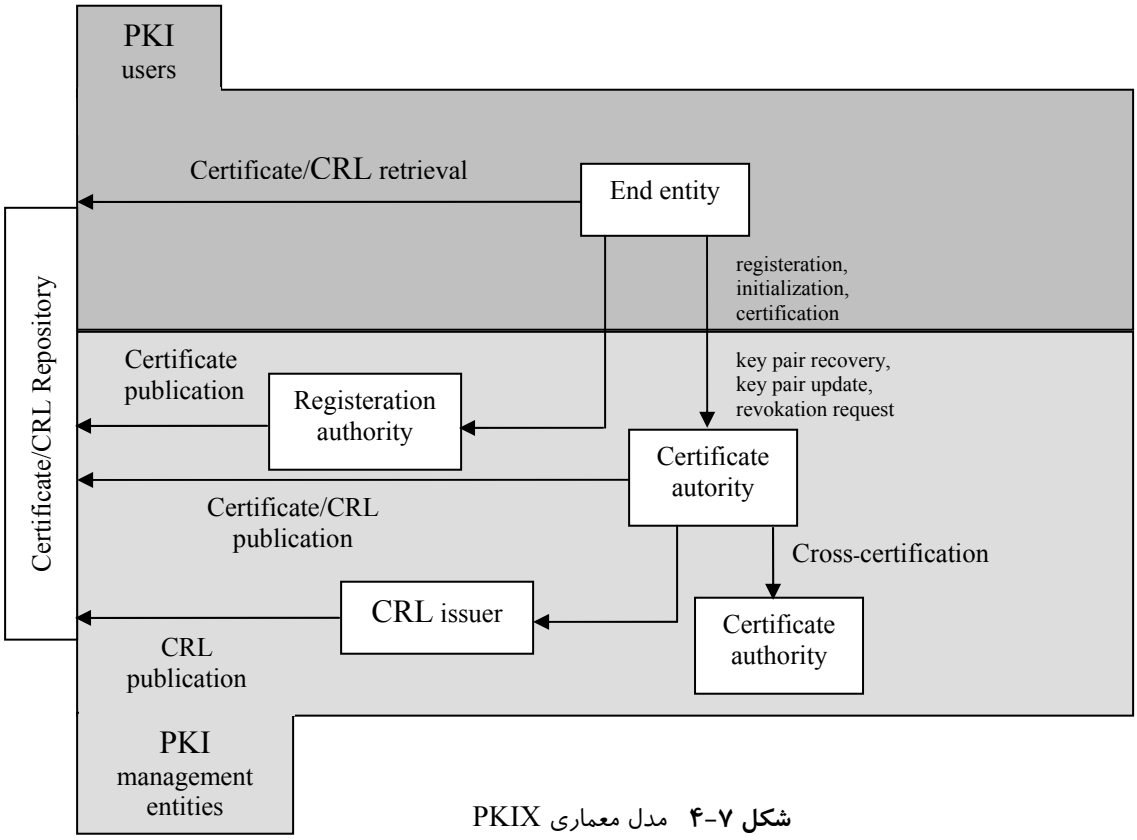
- **قیود اصلی**: نشان می‌دهد که آیا سوژه می‌تواند بصورت یک CA عمل کند. اگر چنین است، ممکن است محدودیتی نسبت به طول مسیر گواهی کردن تعیین گردد.
- **قیود نام‌گذاری**: نشان‌دهندهٔ فضای نام است که در آن نام تمام سوژه‌ها، در گواهی‌های آتی یک مسیر گواهی کردن بایستی جای گیرد.
- **قیود خط‌مشی‌ها**: مشخص‌کنندهٔ قیودی است که ممکن است نیاز به تعیین صریح خط‌مشی‌ها داشته و یا مانع نگاشت خط‌مشی در مابقی مسیر گواهی کردن شوند.

۴-۳ زیرساخت کلید- عمومی (PKI)

RFC 2822 (*فرهنگ لغات امنیتی اینترنت*)، زیرساخت کلید- عمومی (Public-Key Infrastructure) را بصورت مجموعه‌ای از سخت‌افزارها، نرم‌افزارها، آدم‌ها، سیاست‌ها و رَویه‌های لازم برای خلق، مدیریت، نگهداری، توزیع و ابطال گواهی‌نامه‌های دیجیتال بر مبنای رمزنگاری غیرمتقارن تعریف می‌کند. هدف اصلی برپائی یک PKI این است که یک روش امن، سهل و بهره‌ور برای بدست آوردن کلیدهای عمومی ایجاد گردد. گروه کاری زیرساخت کلید- عمومی X.509 در IETF، نیروی محرک ایجاد یک مدل رسمی (و عمومی) مبتنی بر گواهی‌نامهٔ X.509 بوده است تا ساختاری مبتنی بر گواهی‌نامه در اینترنت را ایجاد نماید.

شکل ۷-۴ رابطهٔ درونی عناصر کلیدی مدل PKIX را نشان داده است. این عناصر به قرار زیراند:

- **واحد انتهائی(End entity)**: یک اصطلاح رسمی برای مشخص کردن کاربران انتهائی، دستگاه‌ها (مثل سِرور و مسیریاب) و یا هر موجودیت دیگری که بتواند در مقولهٔ یک گواهی‌نامهٔ کلید- عمومی یافت شود.
- **مسئول گواهی‌نامه‌ها (CA)**: صادرکنندهٔ گواهی‌نامه‌ها و (معمولاً) لیست گواهی‌نامه‌های ابطال شده (CRL). این مقام همچنین ممکن است وظایف مدیریتی دیگری نیز انجام دهد، اگرچه اغلب این وظایف به یک یا چند مسئول ثبت‌نام (Registeration Authority) تفویض می‌شود.



شکل ۷-۴ مدل معماری PKIX

- **مسئول ثبت‌نام (RA)**: یک مؤلفهٔ اختیاری که می‌تواند مسئولیت بخشی از وظایف مدیریتی CA را بعهده گیرد. RA اغلب مسئول پردازش ثبت‌نام واحدهای انتهائی بوده اما می‌تواند در محدوده‌های دیگری نیز فعالیت نماید.
- **صادرکنندهٔ (CRL)**: یک مؤلفهٔ اختیاری که می‌تواند از جانب CA برای انتشار CRLها مأمور شود.
- **مخزن (Repository)**: یک اصطلاح عام که به هر نوع روش کاری برای ذخیره نمودن گواهی‌نامه‌ها و CRLها اشاره دارد، بطوری که آنها بتوانند از طریق واحدهای انتهائی درخواست گردند.

وظایف مدیریتی PKIX

PKIX شماری از وظایف مدیریتی که حتماً لازم است تا بتوسط پروتکل‌های مدیریتی حمایت شوند را مشخص می‌سازد. اینها در شکل ۷-۴ نشان داده شده‌اند و شامل اقلام زیراند:

- **ثبت‌نام (Registeration)**: این عملی است که در آن ابتداءً یک کاربر خود را به CA می‌شناساند (مستقیماً و یا از طریق یک RA). این عمل بایستی قبل از اینکه CA یک گواهی‌نامه و یا گواهی‌نامه‌هائی را برای آن کاربر صادر کند، انجام شود. ثبت‌نام آغاز مرحلهٔ عضوشدن در یک PKI است. ثبت‌نام معمولاً شامل یک سری رَویه‌های برخط (on-line) یا برون خط(off-line) برای احراز هویت متقابل است. معمولاً برای واحد انتهائی یک یا چند کلید سرّی مشترک که در اعتبارسنجی‌های آتی بکار خواهد رفت صادر می‌شود.

- **آغازیدن(Initialization)**: قبل از اینکه یک سیستم کلاینت بتواند بصورت امن عمل نماید، لازم است تا اقلام کلید،که رابطهٔ مناسبی با کلیدهای ذخیره شده در نواحی دیگر زیرساخت دارند، در آن نصب گردند. بعنوان مثال لازم است که کلاینت را با کلید عمومی و سایر اطلاعات مورد نیاز CA مورد اعتماد تجهیز کرد تا بتواند در تأئید مسیر گواهی‌ها از آنها استفاده کند.

- **صدورگواهی‌نامه (Certification)**: این مرحله‌ای است که در آن CA یک گواهی‌نامه برای کلید عمومی یک کاربر صادر نموده و این گواهی‌نامه را به سیستم کلاینت کاربر برگردانده و/ یا آن را در یک مخزن نگاه می‌دارد.

- **بازیابی جفت کلید (Key pair recovery)**: جفت کلیدها می‌توانند برای حمایت از خلق و یا تأئید امضاء دیجیتال و رمزنگاری/رمزگشائی بکار روند. وقتی یک جفت کلید برای رمزنگاری/رمزگشائی بکار می‌رود مهم است، تا برای زمانی که دیگر دسترسی به اقلام کلید میسر نیست، مکانیسمی برای بازیابی کلیدهای رمزگشائی لازم ایجاد کرد. در غیر اینصورت امکان نخواهد داشت که داده‌های رمزنگاری شده را بازیابی نمود. عدم امکان دسترسی به کلید رمزگشائی می‌تواند بعلت فراموش کردن کلمات عبور/PINها، خراب شدن دیسک‌ها، صدمه یافتن ژتون‌های سخت‌افزاری و غیره رخ دهد. بازیابی جفت کلید به واحدهای انتهائی اجازه می‌دهد که جفت کلیدهای رمزنگاری/رمزگشائی خود را از یک تسهیلات مسئول پشتیبانی کلید بازیابی نمایند (معمولاً CA صادرکنندهٔ گواهی‌نامهٔ واحد انتهائی مسئول این کار خواهد بود).

- **به‌روزرسانی جفت کلید (Key pair update)**: تمام جفت کلیدها لازم است تا بطور منظم به‌روزرسانی شوند (یعنی با یک جفت کلید جدید تعویض گردند) و گواهی‌نامه‌های جدیدی صادر گردد. به‌روزرسانی وقتی لازم است که طول عمر گواهی‌نامه تمام شده و یا گواهی‌نامه باطل شود.

- **درخواست ابطال (Revokation request)**: یک فرد مسئول به یک CA اطلاع می‌دهد که به دلیل شرایط غیرنرمال بوجود آمده ابطال یک گواهی‌نامه ضروری است. دلایل ابطال می‌تواند لورفتن کلید خصوصی، تغییر در روش پذیرش و یا تغییر نام باشد.

- **صدور گواهی‌نامه‌های تقاطعی (Cross certification)**: دو CA اطلاعاتی را مبادله می‌کنند که برای صدور یک گواهی‌نامهٔ تقاطعی بکار می‌رود. یک گواهی‌نامهٔ تقاطعی، گواهی‌نامه‌ای است که از سوی یک CA برای CA دیگر صادر می‌شود و شامل یک کلید امضاء CA است که برای صدور گواهی‌نامه بکار می‌رود.

پروتکل‌های مدیریتی PKIX

گروه کاری PKIX دو پروتکل مدیریتی انتخابی متفاوت بین دو واحد PKIX تعریف کرده است که وظایف مدیریتی لیست شده در بخش قبل را بعهده دارند. RFC 2510 پروتکل‌های مدیریت گواهی‌نامه certificate management protocol (CMP) را تعریف می‌کند. در CMP، هریک از وظایف مدیریتی بطور صریح بتوسط مبادله‌های پروتکلی مشخص تعریف شده است. CMP طوری طراحی شده است تا یک پروتکل انعطاف‌پذیر بوده و بتواند نیازهای مدل‌های متنوع فنی، عملیاتی و تجاری را برآورده سازد. RFC 2797 پیام‌های مدیریتی گواهی‌نامه‌ها روی CMS (CMC) را تعریف می‌کند که CMS به RFC 2797 که ساختار پیام‌های رمزی است، اشاره دارد. CMC بر مبنای کارهای ابتدائی‌تر ساخته شده و بمنظور اعمال به پیاده‌سازی‌های موجود بکار می‌رود. در CMC اگرچه تمام عملیات PKIX مورد حمایت‌اند ولی تمام وظایف به مبادلات پروتکلی مشخص نگاشت نمی‌شوند.

۴-۴ منابع مطالعاتی

یک راه بدون دردسر برای درک مفاهیم Kerberos رجوع به [BRYA88] است. یکی از بهترین مراجع Kerberos [KOHL94] می‌باشد. Kerberos [TUNG99] را از نقطهٔ نظر یک کاربر توصیف کرده است.

[PERL99] مدل‌های مختلف اعتماد که می‌توانند در PKI مورد استفاده قرار گیرند را بررسی می‌کند. [GUTM02] مشکلات استفاده از PKI را مورد توجه قرار داده و پیشنهاداتی برای ایجاد یک PKI مؤثر را ارائه می‌دهد.

BRYA88 Bryant,W. *Designing an Authentication System: A Dialogue in Four Scenes.* Project Athena document,February 1988. Available at **http://web.mit.edu/kerberos/www/dialogue.html**.

GUTM02 Gutmann, P."PKI: It`s Not Dead, Just Resting." *Computer*, August 2002.

KOHL94 Kohl,J.; Neuman, B.; and Ts'o,T." The Evolution of the Kerberos Authentication Service." in Brazier, F.,and Johansen,D. *Distributed Open Systems*. Los Alamitos,CA:IEEE Computer Society Press,1994. Available at **http://web.mit.edu/kerberos/www/papers.html**.

PERL99 Perlman, R. "An Overview of PKI Trust Models." *IEEE Network*, November/December 1999.

TUNG99 Tung,B. *Kerberos: A network Authentication System.* Reading, MA: Addison-Wesley,1999.

وِب سایت‌های مفید

- **MIT Kerberos Site**: اطلاعاتی در بارهٔ Kerberos که شامل FAQ، مقاله‌ها و اسناد و اشاره به سایت‌های محصولات تجاری است.

- **USC/ISI Kerberos Page**: یک منبع خوب دیگر از موارد مربوط به Kerberos..
- **Kerberos Working Group**: استانداردهای در حال توسعه مبتنی بر Kerberos گروه IETF..
- **Public-Key Infrastructure Working Group**: استانداردهای در حال توسعهٔ مبتنی بر X.509v3 گروه IETF..
- **Verisign**: یک فروشندهٔ تجاری پیشتاز محصولات مرتبط با X.509 . مقالات و سایر مطالب ارزنده در این سایت.
- **NIST PKI Program**: یک منبع اطلاعاتی خوب.

۴-۵ واژه‌های کلیدی، سؤالات مرورکنندهٔ بحث و مسائل

واژه‌های کلیدی

گواهی‌نامهٔ کلید- عمومی	public – key certificate	اعتبارسنجی	authentication
قلمرو	realm	سِرور اعتبارسنجی	authentication server (AS)
شمارهٔ ردیف	sequence number	یک پروتکل اعتبارسنجی معروف	Kerberos
زیرکلید	subkey	قلمرو Kerberos	Kerberos realm
بلیت	ticket	طول عمر	lifetime
سِرور اعطاکنندهٔ بلیت	ticket–granting server (TGS)	حال فعلی	nonce
گواهی‌نامهٔ X.509	X.509 certificate.	مُود زنجیره‌ای رمز قالبی انتشاریابنده	propagating cipher block chaining (PCBC) mode

سوالات مرورکننده بحث

۴-۱ Kerberos برای پاسخ‌گوئی به چه مشکلی طراحی گردید؟

۴-۲ سه تهدیدی که مرتبط با اعتبارسنجی کاربر روی یک شبکه و یا اینترنت است، کدامند؟

۴-۳ سه روش که می‌تواند اعتبارسنجی کاربر در یک محیط توزیع شده را امنیت بخشد، کدامند؟

۴-۴ چهار نیازی که برای Kerberos تعریف شده است، کدامند؟

۴-۵ کدام اقلام، یک محیط full- service Kerberos را تشکیل می‌دهند؟

۴-۶ در بستر Kerberos، یک قلمرو چیست؟

۴-۷ تفاوت‌های عمدهٔ نسخه‌های چهارم و پنجم Kerberos کدامند؟

۴-۸ هدف استاندارد X.509 چیست؟

۴-۹ یک زنجیره گواهی‌ها چیست؟

۴-۱۰ چگونه یک گواهی‌نامهٔ X.509 باطل می‌شود؟

مسائل

۴-۱ نشان دهید که در مُود PCBC، یک خطای تصادفی در یک بلوک متن رمزنگاری شده به تمام بلوک‌های بعدی متن ساده گسترش می‌یابد(شکل ۹-۴).

۴-۲ فرض کنید که در مُود PCBC، بلوک‌های C_i و C_{i+1} در هنگام انتقال باهم عوض می‌شوند. نشان دهید که این امر فقط بلوک‌های رمزگشائی‌شده P_i و P_{i+1} را تحت تأثیر قرار داده و به بلوک‌های دیگر کاری ندارد.

۴-۳ روش اعتبارسنجی اولیهٔ X.509 نشان داده شده در شکل ۶-۴ج دارای یک نقص امنیتی است. جوهر پروتکل چنین است:

$$\text{A} \rightarrow \text{B:} \quad \text{A}\ \{ t_A, r_A, ID_B \}$$
$$\text{B} \rightarrow \text{A:} \quad \text{B}\ \{ t_B, r_B, ID_A, r_A \}$$
$$\text{A} \rightarrow \text{B:} \quad \text{A}\ \{ r_B \}$$

متن X.509 چنین بیان می‌کند که برچسب‌های زمانی t_A و t_B برای اعتبارسنجی سه- سویه اختیاری هستند. اما مثال زیر را در نظر بگیرید: فرض کنید که A و B پروتکل قبل را در موردی قبلاً بکار گرفته و دشمن C سه پیام قبلی را شنود کرده است. علاوه برآن فرض کنید که از برچسب‌ها استفاده نشده و همهٔ آنها مساوی 0 قرارداده شده‌اند. بالاخره فرض کنید که C علاقه‌مند است تا نزد B خود را بجای A جا بزند. C در ابتدا اولین پیام دزدیده شده را برای B می‌فرستد:

$$\text{C} \rightarrow \text{B:} \quad \text{A}\ \{ 0, r_A, ID_B \}$$

B در حالی که فکر می‌کند با A صحبت می‌کند، جواب C را چنین می‌دهد:

$$\text{B} \rightarrow \text{C:} \quad \text{B}\{ 0, r'_B, ID_A, r_A \}$$

C در همین حال بنحوی A را وامیدارد تا به احراز هویت C بپردازد. در نتیجه A پیام زیر را برای C می‌فرستد:

$$\text{A} \rightarrow \text{C:} \quad \text{A}\ \{ 0, r'_A, ID_C \}$$

C به A پاسخ می‌دهد و از همان nonce که توسط B برای C تهیه شده بود استفاده می‌کند.

$$\text{C} \rightarrow \text{A:} \quad \text{C}\{0, r'_B, \text{A}, r'_\text{A} \}$$

A چنین پاسخ می‌دهد:

$$\text{A} \rightarrow \text{C:} \quad \text{A}\ \{ r'_B \}$$

این همان چیزی است که C لازم دارد تا B را متقاعد سازد که دارد با A صحبت می‌کند، و بنابراین C اکنون پیام ورودی را برای B پس می‌فرستد:

$$\text{C} \rightarrow \text{B:} \quad \text{A}\ \{ r'_B \}$$

بنابراین B باور خواهد کرد که دارد با A صحبت می‌کند در حالی که در واقع دارد با C صحبت می‌کند. یک راه حل ساده برای این مشکل پیدا کنید که نیازی به استفاده از برچسب‌های زمانی نداشته باشد.

۴-۴ X.509 در نسخهٔ سال ۱۹۹۸، خواصی که باید کلیدهای RSA داشته تا امن باشند، بر اساس معلومات فعلی در مورد سخت بودن به فاکتور درآوردن اعداد بزرگ، را درج می‌کند. این بحث با قرار دادن یک محدودیت در نمای عمومی و مدول n چنین پایان می‌پذیرد.:

بایستی اطمینان یافت که $e > \log_2(n)$ است تا بتوان از حملات با گرفتن ریشهٔ e ام $\bmod n$ به منظور افشاکردن متن ساده جلوگیری کرد.

اگرچه این قید صحیح است ولی دلیل ارائه شده برای نیاز به آن ناصحیح است. در این استدلال چه مشکلی نهفته است و استدلال صحیح چیست؟

ضمیمهٔ ۴- الف تکنیک‌های رمزنگاری KERBEROS

Kerberos شامل یک کتابخانهٔ رمزنگاری است که عملیات متنوعی را که به رمزنگاری مربوط می‌شود پشتیبانی می‌کند. اینها در مشخصه‌های نسخهٔ پنجم Kerberos گنجانده شده بودند و معمولاً در پیاده‌سازی‌های تجاری یافت می‌شوند. در فوریهٔ سال ۲۰۰۵، RFCهای 3961 و 3962 انتشار یافتند که موارد اختیاری تکنیک‌های رمزنگاری را توسعه می‌دهند. در این ضمیمه، تکنیک‌های RFC 1510 را شرح می‌دهیم.

تبدیل کلمهٔ عبور- به- کلید

در Kerberos، کلمات عبور محدود به کاراکترهائی هستندکه می‌توانند با فرمت ۷- بیتی ASCII نمایش داده شوند. این کلمه عبور، که دارای طول دلخواهی است، به یک کلید رمزنگاری تبدیل شده که در پایگاه دادهٔ Kerberos ذخیره می‌شود. شکل ۸-۴ روش کار را نشان می‌دهد.

ابتدا دنبالهٔ کاراکترها، s، بصورت یک دنبالهٔ بیت‌ها، b، در می‌آید بطوری که اولین کاراکتر در اولین ۷ بیت، دومین کاراکتر در دومین ۷ بیت و . . . جای داده می‌شوند. این امر را می‌توان چنین نشان داد

$$b[0] = \text{bit } 0 \text{ of } s[0]$$

. .

$$b[6] = \text{bit } 6 \text{ of } s[0]$$
$$b[7] = \text{bit } 0 \text{ of } s[1]$$

. .

$$b[7i + m] = \text{bit } m \text{ of } s[i] \quad 0 \le m \le 6$$

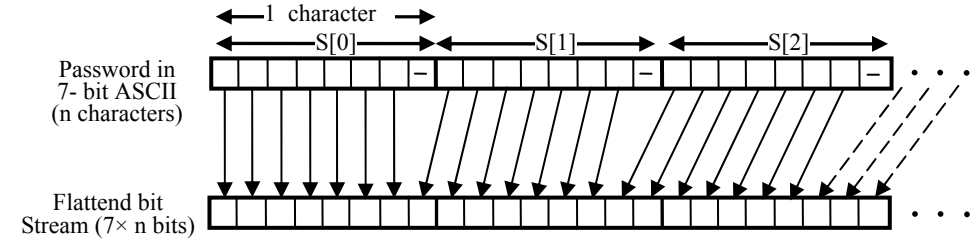
سپس دنبالهٔ بیت‌ها بصورت بادبزنی به یک دنبالهٔ ۵۶- بیتی تبدیل می‌شود. بعنوان مثال اگر دنبالهٔ بیت‌ها دارای طول ۵۹ باشد آنگاه

$$b[55] = b[55] \oplus b[56]$$
$$b[54] = b[54] \oplus b[57]$$
$$b[53] = b[53] \oplus b[58]$$

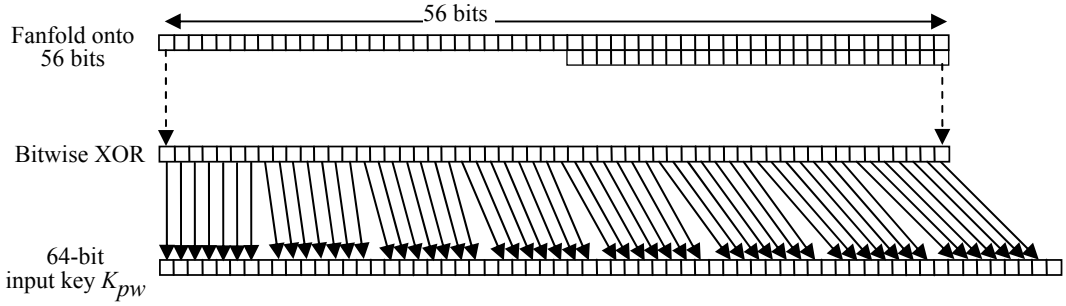
این یک کلید ۵۶- بیتی DES را ایجاد می‌کند. برای اینکه با فرمت مورد انتظار کلید ۶۴- بیتی همخوان باشد، با این دنباله بصورت یک ردیف از بلوک‌های ۷- بیتی رفتار شده که به بلوک‌های ۸ - بیتی نگاشت می‌شود تا کلید ورودی K_{pw} را ایجاد کند.

بالاخره کلمه عبور اولیه با مُود زنجیره‌ای رمز قالبی(CBC) الگوریتم DES با کلید K_{pw} رمزنگاری می‌شود. آخرین بلوک ۶۴- بیتی که از این روش حاصل می‌شود و به نام جمع کنترلی CBC خوانده می‌شود، کلید خروجی مرتبط با این کلمهٔ عبور است.

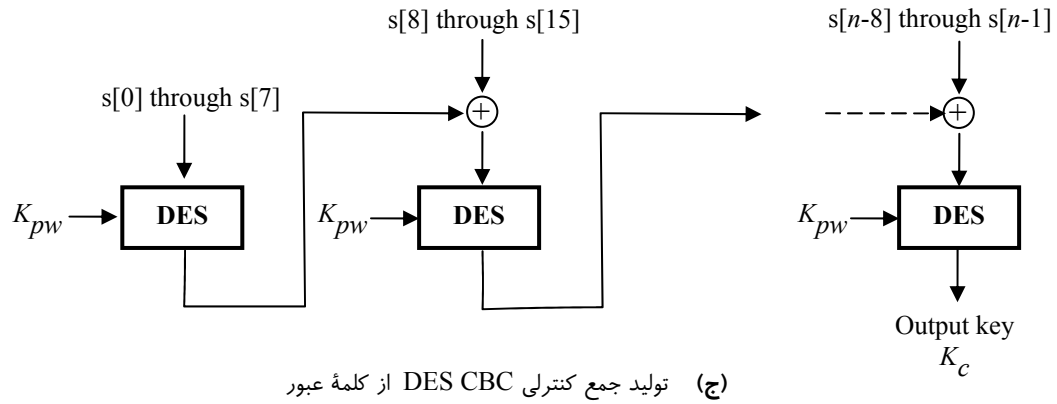
کل الگوریتم را می‌توان تابع درهم‌سازی دانست که یک کلمهٔ عبور دلخواه را به یک کُد hash ۶۴- بیتی تبدیل می‌کند.



(الف) تبدیل کلمهٔ عبور به دنبالهٔ بیت‌ها



(ب) تبدیل دنبالهٔ بیت‌ها به کلید ورودی



(ج) تولید جمع کنترلی DES CBC از کلمهٔ عبور

شکل ۸-۴ تولید کلید رمزنگاری از کلمهٔ عبور

مُود زنجیره‌ای رمزقالبی انتشاریابنده (Propagating Cipher Block Chaining Mode)

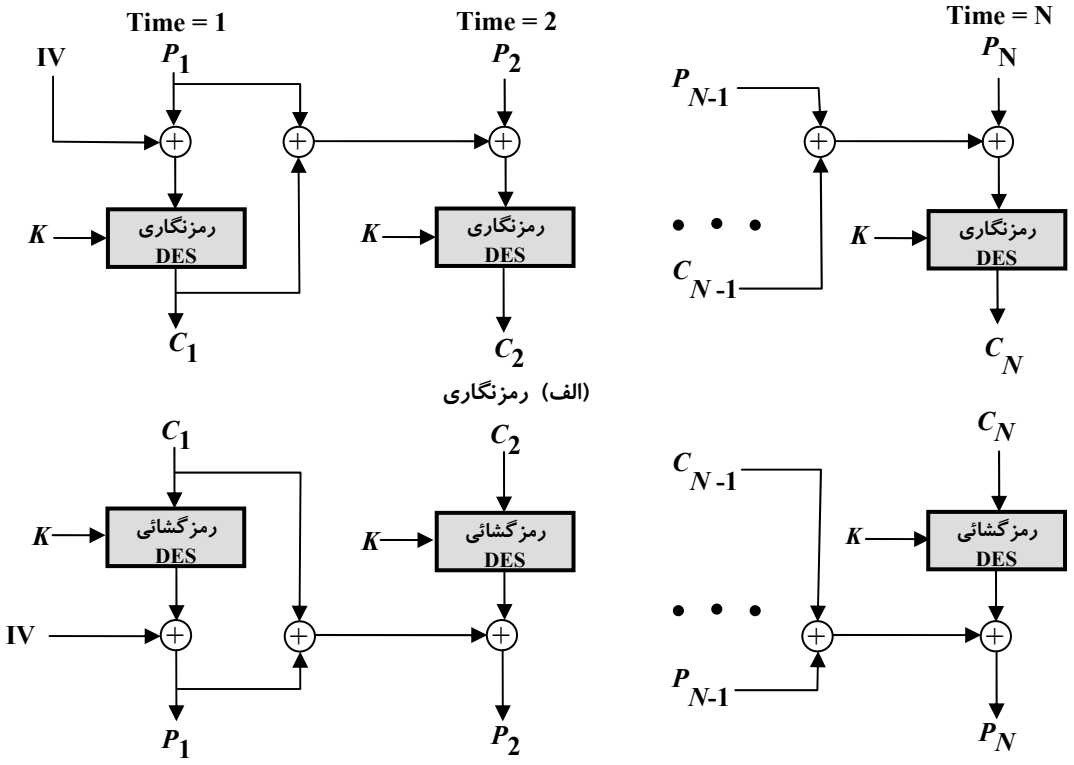
از فصل ۲ بخاطر آورید که در مُود CBC الگوریتم DES، ورودی الگوریتم در هر مرحله شامل XOR بلوک جاری متن ساده با بلوک رمزشده مرحلهٔ قبل بود که برای هر بلوک نیز از همان یک کلید استفاده می‌شد (شکل ۹-۲). مزیت این مُود نسبت به مُود کتاب لغت الکترونیکی (ECB) که در آن هر بلوک بصورت مستقل رمزنگاری می‌شود این است: در CBC اگر یک بلوک متن ساده در جای دیگری تکرار شود بلوک‌های رمزنگاری شدهٔ متفاوتی تولید خواهد شد.

CBC دارای این خاصیت است که اگر در انتقال بلوک رمزشدهٔ C_I خطائی رخ دهد این خطا به بلوک‌های رمزگشائی شدهٔ P_I و P_{I+1} گسترش می‌یابد.

نسخهٔ چهارم Kerberos فرم پیچیده‌تری از CBC را که CBC انتشاریابنده (PCBC) نامیده می‌شود مورد استفاده قرار می‌دهد. این مُود دارای این خاصیت است که هر خطا در یکی از بلوک‌های متن رمزشده به همهٔ بلوک‌های رمزگشائی شدهٔ بعدی گسترش یافته و آنها را بی‌ارزش می‌سازد. بنابراین رمزنگاری و صحت داده‌ها در یک عمل حاصل می‌گردند (برای یک حالت استثناء به مسألهٔ ۲-۴ نگاه کنید).

PCBC در شکل ۹-۴ نشان داده شده است. در این روش ورودی الگوریتم رمزنگاری، XOR بلوک جاری متن ساده، بلوک قبلی متن رمزشده، و بلوک قبلی متن ساده است:

$$C_n = \mathrm{E}\,(K, [C_{n-1} \oplus P_{n-1} \oplus P_n])$$



شکل ۹-۴ مُود زنجیره‌ای رمز قالبی انتشاریابنده (PCBC)

در موقع رمزگشائی، هر بلوک متن رمزشده از الگوریتم رمزگشائی عبور می‌کند. سپس خروجی با بلوک متن رمزشدهٔ قبلی و بلوک متن سادهٔ قبلی XOR می‌شود. با رابطهٔ زیر می‌توان نشان داد که این روش صحیح عمل می‌کند:

$$D(K, C_n) = D(K, E(K, [C_{n-1} \oplus P_{n-1} \oplus P_n]))$$

$$D(K, C_n) = C_{n-1} \oplus P_{n-1} \oplus P_n$$

$$C_{n-1} \oplus P_{n-1} \oplus D[K, C_n] = P_n$$

فصل ۵

امنیت پست الکترونیک

در بین تمام محیط‌های توزیع شده، پست الکترونیک تقریباً پراستفاده‌ترین کاربرد مبتنی بر شبکه است. این سرویس همچنین تنها کاربرد توزیع‌شده‌ای است که در تمام معماری‌ها و سیستم‌های عامل کامپیوتری بطور وسیعی مورد استفاده قرار می‌گیرد. کاربران اینترنت انتظار دارند که بتوانند به همهٔ کسان دیگری که به اینترنت متصل‌اند، صرف‌نظر از سیستم عامل و یا پروتکل ارتباطی مورد استفاده، نامه ارسال نمایند.

با وابستگی روزافزون به پست الکترونیک برای هر مقصود قابل تصور، نیـــاز به سرویس‌های اعتبـــارسنجی و محـــرمانگی مرتباً بیشتر می‌شود. در این زمینه دو روش بطور گسترده‌ای مورد استقبال قرار گرفته و از آنها استفاده می‌شود: Pretty Good Privacy (PGP) و S/MIME. هر دوی آنها در این فصل مورد بررسی قرار می‌گیرند.

۵-۱ PRETTY GOOD PRIVACY

PGP یک پدیدهٔ فوق‌العاده است. PGP با تلاش‌های نسبتاً انفرادی یک نفر بنام Phil Zimmermann، سرویسی است که محرمانگی و اعتبارسنجی را برای پست الکترونیک و همچنین کاربردهای ذخیره‌سازی فایل فراهم می‌آورد. Zimmermann کارهای زیر را انجام داده است:

۱- بهترین الگوریتم‌های موجود رمزنگاری را به عنوان پایه‌های این بنا انتخاب نمود.
۲- این الگوریتم‌ها را طوری در یک کاربرد عام تلفیق کرد که مستقل از سیستم عامل و پردازش‌گر بوده و بر مبنای مجموعهٔ کوچکی از فرامین سهل قرار دارد.
۳- بستهٔ نرم‌افزاری ایجاد شده و اسناد مربوطه که شامل کُد منبع برنامه نیز می‌باشد را از طریق اینترنت، تابلوهای اعلانات و شبکه‌های تجاری همانند AOL (American On Line) بطور مجانی در اختیار کاربران قرار داد.
۴- با یک شرکت (Viacrypt که امروز Network Associates خوانده می‌شود) قراردادی بست که یک نسخهٔ تجاری کاملاً سازگار و ارزان قیمت از PGP را تهیه نماید.

PGP بطور انفجارآمیزی رشدکرده و امروز در سطح گسترده‌ای از آن استفاده می‌شود. برخی از این دلایل این رشد چنین‌اند:

۱- نسخه‌های متعددی از آن که روی کامپیوترهای مختلفی با سیستم عامل‌های متنوع همانندWindows، UNIX، Macintosh و بسیاری دیگر کار می‌کنند بصورت جهانی و مجانی در دسترس است. علاوه براین، نسخهٔ تجاری آن کاربرانی را که به دنبال محصولی با خدمات پشتیبانی بعدی هستند راضی نموده است.
۲- بر مبنای الگوریتم‌هائی قرار دارد که بارها و بارها در آنها تجدید نظر شده و بسیار امن تلقی می‌شوند. علی‌الخصوص بستهٔ نرم‌افزاری شامل RSA، DSS و Diffie-Hellman در حوزهٔ رمزنگاری کلید- عمومی، CAST-128، IDEA و 3DES در حوزهٔ رمزنگاری متقارن و SHA-1 برای درهم‌سازی می‌باشد.

۳- دارای کاربردهای بسیار متنوعی است که از سازمان‌هائی که علاقه‌مند به انتخاب و اجرای یک روش استاندارد برای رمزکردن فایل‌ها و پیام‌ها می‌باشند شروع شده و به اشخاص حقیقی که علاقه‌مند به ارتباط امن روی اینترنت و سایر شبکه‌ها در سطح جهان می‌باشند ختم می‌گردد.
۴- نه بتوسط یک دولت و یا یک سازمان استانداردسازی تولید شده است و نه بتوسط چنین کسانی کنترل می‌شود. برای کسانی که ذاتاً اعتمادی به «تشکیلات» ندارند، این خاصیت PGP پرجاذبه است.
۵- PGP اگرچه امروز روی خط استانداردهای اینترنت قرار گرفته است(RFC 3156)، ولی با وجود این هنوز دارای فضای معطر ضدتشکیلاتی خود است.

بحث را با نگاهی کلی به عملیات PGP آغاز می‌کنیم. در قسمت بعد چگونگی خلق کلیدهای رمزنگاری و ذخیره کردن آنها را بررسی می‌نمائیم. سپس مقولهٔ بسیار مهم مدیریت کلید- عمومی را مورد توجه قرار می‌دهیم.

علائم اختصاری

بسیاری از علائم اختصاری بکار رفته در این فصل را قبلاً نیز مورد استفاده قرار داده‌ایم ولی تعدادی از آنها جدید می‌باشند. شاید بهتر باشد که این علائم اختصاری را در ابتدا خلاصه کنیم. نشانه‌های زیر بکار گرفته شده‌اند:

K_S = کلید اجلاس که در روش رمزنگاری متقارن از آن استفاده می‌شود.
PR_a = کلید خصوصی کاربر A که در روش رمزنگاری کلید- عمومی از آن استفاده می‌شود.
PU_a = کلید عمومی کاربر A که در روش رمزنگاری کلید- عمومی از آن استفاده می‌شود.
EP = رمزنگاری کلید- عمومی
DP = رمزگشائی کلید- عمومی
EC = رمزنگاری متقارن
DC = رمزگشائی متقارن
H = تابع درهم‌ساز (hash)
|| = جمع رشته‌ای
Z = فشرده‌سازی با الگوریتم ZIP
R64 = تبدیل به فرمت radix-64 ASCII

اسناد PGP اغلب از اصطلاح *کلید سرّی* (*secret key*) برای اشاره به کلیدی که در یک روش رمزنگاری کلید- عمومی در کنار کلید عمومی قرار دارد استفاده می‌کنند. همانطور که قبلاً اشاره کردیم، این عمل ممکن است باعث اشتباه شدن این کلید با کلید سرّی مورد استفاده در رمزنگاری متقارن شود. بنابراین ما بجای آن از اصطلاح *کلید خصوصی* (*private key*) استفاده می‌کنیم.

توصیف عملیاتی

عملیات واقعی PGP، صرف نظر از مدیریت کلیدها، شامل پنج سرویس است: اعتبارسنجی، محرمانگی، فشرده‌سازی، سازگاری e-mail و قطعه قطعه کردن دیتا (جدول ۱-۵). هر یک از این پنج سرویس را به نوبت بررسی می‌کنیم.

اعتبارسنجی

شکل ۱-۵الف سرویس امضاء دیجیتال که بتوسط PGP فراهم می‌شود را نشان می‌دهد. این همان امضاء دیجیتال مورد بحث در فصل ۳ و نشان داده شده در شکل ۲-۳ب است. روند کار چنین است:

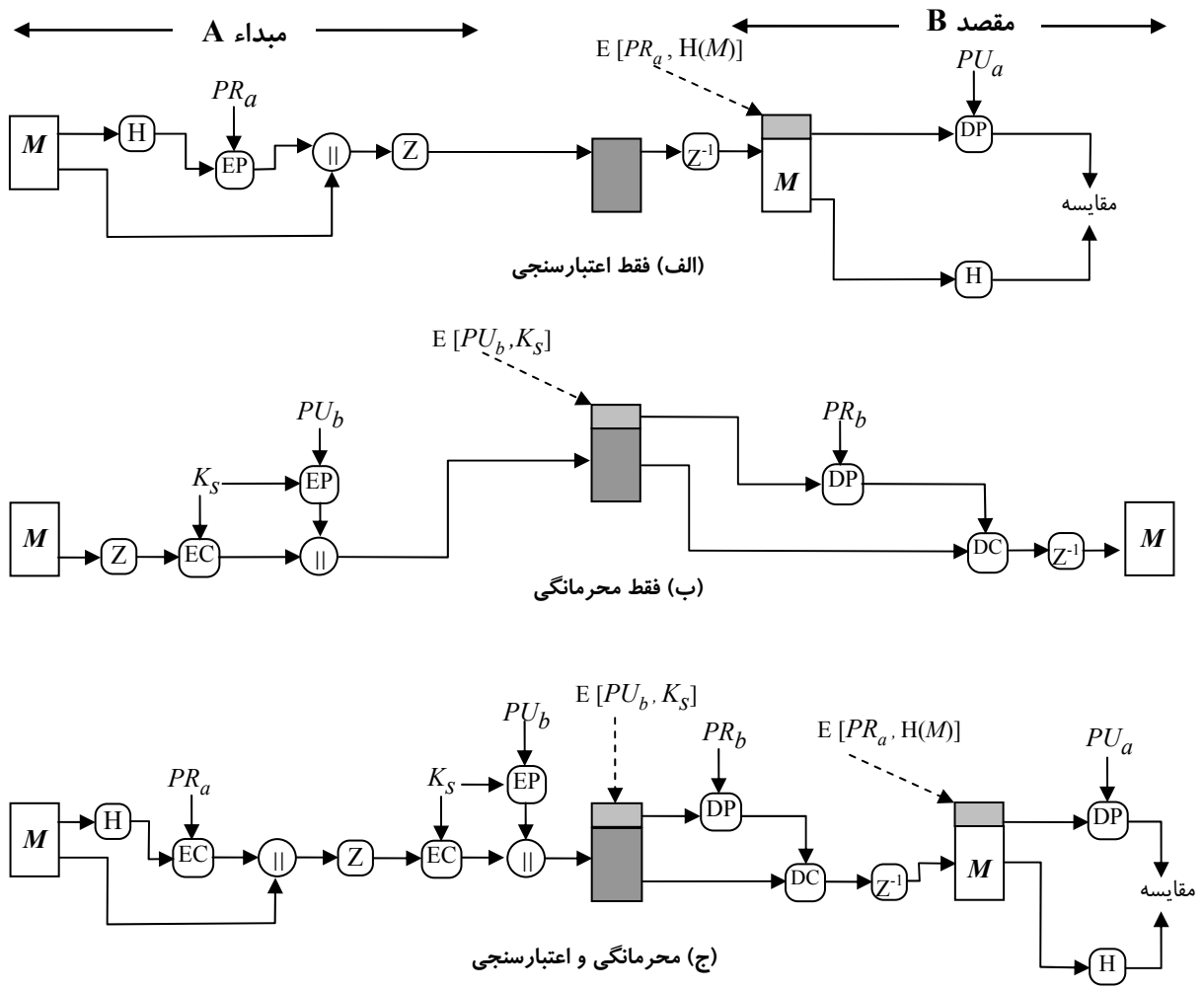
۱- فرستنده پیامی را تهیه می‌کند.
۲- با استفاده از SHA-1 یک کُد hash ۱۶۰- بیتی از پیام ایجاد می‌شود.
۳- کُد hash از طریق RSA رمزنگاری می‌شود که در آن از کلید خصوصی فرستنده استفاده شده است. نتیجه به ابتدای پیام وصل (جمع رشته‌ای) می‌شود.
۴- گیرنده از RSA و کلید عمومی فرستنده استفاده کرده تا کُد hash را رمزگشائی نموده و استخراج کند.
۵- گیرنده یک کُد hash جدید برای پیام تولید کرده و آن را با کُد hash رمزگشائی شده مقایسه می‌کند. اگر دو کُد با هم یکسان باشند، پیام پذیرفته شده و معتبر تلقی می‌گردد.

ترکیب SHA-1 و RSA یک روش مؤثر برای امضاء دیجیتال را فراهم می‌آورد. نظر به قدرت RSA، دریافت‌کننده مطمئن است که تنها صاحب یک کلید خصوصی متناظر، قادر به تولید امضاء بوده است. نظر به قدرت SHA-1، دریافت‌کننده مطمئن است که کس دیگری نمی‌توانسته است پیام جدیدی که کُد hash آن با پیام اصلی و بنابراین با امضاء پیام اصلی یکسان باشد را تولیدکند.

در انتخاب دیگر، استفاده از DSS/SHA-1 برای تولید امضاءها مجاز می‌باشد.

جدول ۱- ۵ خلاصهٔ سرویس‌های PGP

عمل	الگوریتم مورد استفاده	توصیف عملیاتی
امضاء دیجیتال	DSS/SHA یا RSA/SHA	با استفاده از SHA-1 یک کُد hash از پیام ساخته می‌شود. این چکیدهٔ پیام با استفاده از DSS یا RSA و بتوسط کلید خصوصی فرستنده رمزنگاری شده و همراه پیام قرار می‌گیرد.
رمزنگاری پیام	CAST یا IDEA یا DES سه کلیدی با Diffie-Hellman یا RSA	یک پیام با استفاده از CAST-128 یا IDEA یا 3DES و بتوسط یک کلید اجلاس یکبار- مصرف تولیدشده در فرستنده رمزنگاری می‌شود. کلید اجلاس با استفاده از Diffie-Hellman یا RSA و بتوسط کلید عمومی گیرنده رمزنگاری می‌شود و همراه پیام قرار می‌گیرد.
فشرده سازی	ZIP	یک پیام ممکن است با استفاده از ZIP، برای ذخیره‌سازی یا انتقال، فشرده گردد.
سازگاری e-mail	تبدیل Radix-64	به منظور ایجاد شفافیت برای کاربردهای e-mail، یک پیام رمزنگاری شده ممکن است با استفاده از تبدیل Radix-64 به یک دنبالهٔ ASCII تبدیل شود.
قطعه قطعه کردن	—	برای حفظ محدودیت‌های طول ماکزیمم پیام، PGP قطعه‌قطعه کردن و دوباره سرهم کردن دیتا را انجام می‌دهد.



شکل ۱-۵ عملیات رمزنگاری PGP

اگرچه امضاءها معمولاً به پیام و یا فایلی که امضاء آنها را تأئید می‌کند وصل‌اند، ولی این امر همیشه صادق نیست. امضاءهای مجزا از پیام نیز پشتیبانی می‌شوند. یک امضاء غیر متصل به پیام نیز می‌تواند بطور جدا از پیام خود در جائی ذخیره شده و انتقال یابد. این امر در زمینه‌هائی مفید واقع می‌شود. یک کاربر ممکن است علاقه‌مند باشد که برای تمام پیام‌های ارسال شده و یا دریافت شده یک کارنامهٔ امضاء جداگانه داشته باشد. یک امضاء غیرمتصل به یک برنامهٔ اجرائی می‌تواند آلودگی‌های ویروسی بعدی را کشف کند. بالاخره امضاءهای غیرمتصل می‌توانند در جائی که بیش از یک طرف امضاءکننده وجود دارد (مثل یک قرارداد قانونی)، مورد استفاده واقع شوند. امضاء هر فرد مستقل بوده و بنابراین تنها سند اصلی را تأئید می‌کند. در غیر این صورت امضاءها بایستی تودرتو باشند و معنی آن این است که امضاءکنندهٔ دوم، تأئیدکنندهٔ هم سند اصلی و هم امضاء امضاءکنندهٔ اول است و همین ترتیب ادامه می‌یابد.

محرمانگی

سرویس اصلی دیگری که بتوسط PGP فراهم می‌آید محرمانگی است که بتوسط رمزنگاری پیام‌هائی که بایستی ارسال شوند و یا بایستی بصورت فایل‌های محلی ذخیره گردند انجام می‌شود. در هر دو مورد، می‌توان از الگوریتم رمزنگاری متقارن

CAST-128 استفاده نمود. راه حل دیگر استفاده از IDEA و یا 3DES است. مُود فیدبک رمز (CFB) ۶۴- بیتی بکار می‌رود.

مثل همیشه، بایستی مشکل توزیع کلید را درنظر داشته باشیم. در PGP، هر کلید متقارن تنها یک‌بار مورد استفاده قرار می‌گیرد. یعنی برای هر پیام، یک کلید جدید بصورت یک عدد تصادفی ۱۲۸- بیتی ایجاد می‌شود. بنابراین اگرچه در سندها این کلید را بنام کلید اجلاس می‌شناسند، ولی در واقع این یک کلید یکبار مصرف است. چون این کلید تنها یک‌بار مورد استفاده قرار می‌گیرد، به پیام متصل شده و همراه آن ارسال می‌گردد. برای حفاظت از این کلید، آن را با کلید عمومی گیرنده رمزنگاری می‌کنیم. شکل ۱-۵ب رَوند عملیات را نشان می‌دهد که می‌توان آن را چنین توصیف نمود:

۱- فرستنده، پیام خود و همچنین یک عدد ۱۲۸- بیتی تصادفی که قرار است بعنوان کلید اجلاس، تنها برای این پیام، بکار رود را تولید می‌کند.
۲- پیام با استفاده از CAST-128 (یا IDEA یا 3DES) رمزنگاری می‌شود.
۳- کلید اجلاس با استفاده از کلید عمومی دریافت‌کننده پیام و RSA رمزنگاری شده و به پیام الصاق می‌گردد.
۴- گیرنده از RSA و کلید خصوصی برای رمزگشائی و بازیابی کلید اجلاس استفاده می‌کند.
۵- کلید اجلاس برای رمزگشائی بکار می‌رود.

بجای RSA برای رمزنگاری کلید، PGP حق انتخاب دیگری بنام *Diffie-Hellman* را فراهم نموده است. همانطور که در فصل ۳ توضیح داده شد، Diffie-Hellman یک الگوریتم مبادلهٔ کلید است. در حقیقت PGP از نوعی Diffie-Hellman که یک نوع رمزنگاری/رمزگشائی بنام ElGamal را فراهم می‌سازد استفاده می‌کند.

چند نکته در این مورد قابل توجه است. اولاً برای کاهش زمان رمزنگاری، ترکیبی از رمزنگاری متقارن و کلید- عمومی بجای استفادهٔ مستقیم از RSA یا ElGamal بکار می‌رود: CAST-128 و سایر الگوریتم‌های متقارن بطور چشمگیرتری سریع‌تر از RSA یا ElGamal هستند. ثانیاً استفاده از الگوریتم‌های کلید- عمومی، مشکل توزیع کلید اجلاس را حل می‌کنند زیرا تنها گیرنده قادر به بازیابی کلید اجلاسی است که به پیام مرتبط شده است. توجه کنید که در اینجا نیازی به یک پروتکل مبادلهٔ کلید اجلاس، از نوعی که در فصل ۳ مورد بحث قرار گرفت، نیست زیرا یک اجلاس جاری را دوباره آغاز نمی‌کنیم. در اینجا هر پیام با کلید مخصوص به خود، یک پیشامد مستقل است که فقط یکبار واقع می‌شود. علاوه براین، با ماهیت store-and-forward پست الکترونیک، استفاده از دستداد(handshaking) برای اطمینان یافتن از اینکه هر دو سمت دارای کلید اجلاس یکسان هستند عملی نمی‌باشد. بالاخره استفاده از کلیدهای متقارن یکبارمصرف روش محکم رمزنگاری متقارن را محکم‌تر می‌کند. با هر کلید، تنها بخش کوچکی از متن ساده رمزشده و هیچ رابطه‌ای بین کلیدها وجود ندارد. بنابراین تا مرزی که الگوریتم کلید عمومی امن است، کل روش امن خواهد بود. تا زمان حاضر، PGP محدوده‌ای از کلیدها بین ۷۶۸ تا ۳,۰۷۲ بیت را برای کاربر فراهم نموده است (کلید DSS برای امضاءهای دیجیتال محدود به ۱,۰۲۴ بیت است).

محرمانگی و اعتبارسنجی

همانطور که شکل ۱-۵ج نشان می‌دهد، هر دو سرویس را می‌توان برای یک پیام واحد بکار برد. ابتدا یک امضاء برای متن سادهٔ پیام تولید شده و به پیام الصاق می‌گردد. آنگاه متن ساده پیام بعلاوهٔ امضاء با استفاده از CAST-128 (یا IDEA یا 3DES) رمزنگاری شده و کلید اجلاس نیز با استفاده از RSA (یا ElGamal) رمز می‌گردد. این دنبالهٔ وقایع به نوع برعکس آن یعنی رمزنگاری پیام و آنگاه تولید یک امضاء برای پیام رمزشده ارجحیت دارد. معمولاً مناسب‌تر است که یک امضاء را به

همراه فرم سادهٔ پیام ذخیره کرد. علاوه بر این برای تأئید شخص ثالث، اگر عمل امضاء در ابتدا صورت پذیرد، شخص ثالث لازم نیست در هنگام تأئید امضاء، نگران کلید رمز متقارن باشد.

خلاصه اینکه وقتی هر دو سرویس مورد استفاده قرار می‌گیرند، فرستنده ابتدا پیام را با کلید خصوصی خود امضاء کرده، سپس آن را با یک کلید اجلاس رمزنگاری نموده و سپس کلید اجلاس را با کلید عمومی گیرنده به رمز درمی‌آورد.

فشرده‌سازی

بصورت پیش‌فرض، PGP پیام را پس از امضاء و قبل از رمزنگاری فشرده می‌نماید. حسن این امر صرفه‌جوئی در فضا هم برای انتقال e-mail و هم برای ذخیره‌سازی فایل است.

نحوهٔ جاسازی الگوریتم فشرده‌سازی، که در شکل ۱-۵ بصورت Z برای فشرده‌سازی و بصورت Z^{-1} برای عکس آن نشان داده شده است، امری مهم است:

۱- به دو دلیل، امضاء قبل از فشرده‌سازی انجام می‌شود:

الف- اصلح است که یک پیام را قبل از فشرده‌سازی امضاء کرد تا بتوان تنها پیام فشرده نشده به همراه امضاء را برای تأئیدهای آتی ذخیره کرد. اگر یک پیام فشرده شده امضاء شده باشد، آنگاه لازم است یا یک نسخهٔ فشرده شدهٔ پیام را ذخیره کرد و یا هروقت لازم باشد برای تأئید، پیام را از حالت فشرده خارج نمود.

ب- حتی اگر راضی باشیم که برای تأئید یک پیام آن را از حالت فشردگی درآوریم، الگوریتم فشرده‌سازی PGP مشکلی را ایجاد می‌کند. الگوریتم یک الگوریتم قطعی نیست و بکارگیری آن با مصالحه‌ای که بین سرعت اجرا و نسبت فشردگی صورت می‌پذیرد، نسخه‌های فشرده شدهٔ مختلفی از پیام را درست می‌کند. با این وجود، این الگوریتم‌های فشرده‌سازی متنوع در بین خود تراکنش داشته زیرا هر نسخهٔ الگوریتم قادر است خروجی هر نسخهٔ دیگر را بطور صحیح بازکند. اِعمال تابع درهم سازی و امضاء بعد از فشرده‌سازی، تمام پیاده‌سازی‌های PGP را به استفاده از یک الگوریتم فشرده‌سازی محدود می‌نماید.

۲- رمزنگاری پیام بعد از فشرده سازی انجام می‌شود تا امنیت رمزنگاری قدرتمندتر گردد. نظر به اینکه پیام فشرده شده دارای افزونگی کمتری نسبت به متن سادهٔ اصلی آن است، کشف رمزآن مشکل‌تر خواهد بود.

الگوریتم فشرده‌سازی مورد استفاده ZIP است که در ضمیمهٔ ۵- الف توصیف گردیده است.

سازگاری E-mail

وقتی PGP مورد استفاده قرار می‌گیرد، حداقل بخشی از بلوکی که باید انتقال یابد رمزنگاری می‌شود. اگر فقط سرویس امضاء بکار گرفته شود، آنگاه چکیدهٔ پیام (digest) رمزنگاری می‌شود (با کلید خصوصی فرستنده). اگر سرویس محرمانگی بکار گرفته شود، آنگاه پیام باضافهٔ امضاء (اگر وجود داشته باشد) رمزنگاری می‌شود (با یک کلید متقارن یکبارمصرف). بنابراین بخشی و یا تمام بلوک نتیجه شده شامل دنباله‌هائی از اُکتت‌های ۸- بیتی اختیاری خواهند بود. اما بسیاری از سیستم‌های پست الکترونیک تنها استفاده از بلوک‌هائی که شامل کُد ASCII باشند را مجاز می‌شمارند. برای همکاری در رفع این محدودیت، PGP سرویسی را فراهم آورده است که دنبالهٔ باینری ۸- بیتی خام را به یک دنبالهٔ قابل چاپ از کاراکترهای ASCII تبدیل می‌کند.

روشی که برای این مقصود بکار می‌رود، تبدیل radix-64 است. هر گروه سه اُکتتی از داده‌های باینری به چهار کاراکتر ASCII تبدیل می‌شوند. این فرمت همچنین یک CRC (Cyclic Redundancy Check) برای تشخیص خطاهای انتقال را به دیتا وصل می‌کند. توصیف این تبدیل در ضمیمهٔ ۵- ب بیان شده است.

استفاده از radix-64، طول یک پیام را بمیزان ۳۳٪ افزایش می‌دهد. خوشبختانه کلید اجلاس و بخش امضاء پیام نسبتاً کوتاه بوده و متن پیام، فشرده شده است. در واقع فشرده‌سازی این بخش بایستی آنقدر باشد که بتواند بر گسترش پیام در تبدیل به radix-64 فایق آید. بعنوان مثال [HELD96] از نسبت فشرده‌سازی متوسطی به میزان ۲ با استفاده از ZIP خبر می‌دهد. اگر از طول نسبتاً کوتاه امضاء و مؤلفه‌های مربوط به کلید صرف‌نظر شود، نتیجهٔ معمول فشرده‌سازی و گسترش یک فایل با طول X برابر ۰/۶۶۵ × X = ۰/۵ × ۱/۳۳ × X خواهد بود. بنابراین رویهم رفته هنوز یک فشردگی بمیزان ۰/۳۳ در پیام ایجاد می‌گردد.

یکی از جنبه‌های قابل توجه الگوریتم radix-64 این است که کورکورانه و بدون توجه به محتوا، دنبالهٔ ورودی را به فرمت radix-64 تبدیل می‌کند، حتی اگر خود دنبالهٔ ورودی متن ASCII باشد. بنابراین اگر پیامی امضاء شده ولی رمزنگاری نشده باشد و تبدیل به همهٔ بلوک اعمال شود، خروجی برای یک ناظر اتفاقی قابل خواندن نخواهد بود و بنابراین خود سطحی از محرمانگی را ایجاد می‌کند. بطور اختیاری، PGP را می‌توان طوری پیکربندی کرد که فرمت radix-64 را تنها به بخش امضاء پیام‌های متنی سادهٔ امضاء شده اعمال نماید. این امر گیرندهٔ انسانی را قادر می‌سازد تا بدون استفاده از PGP پیام را بخواند. در این‌حالت نیز بایستی از PGP برای تأئید امضاء کمک گرفت.

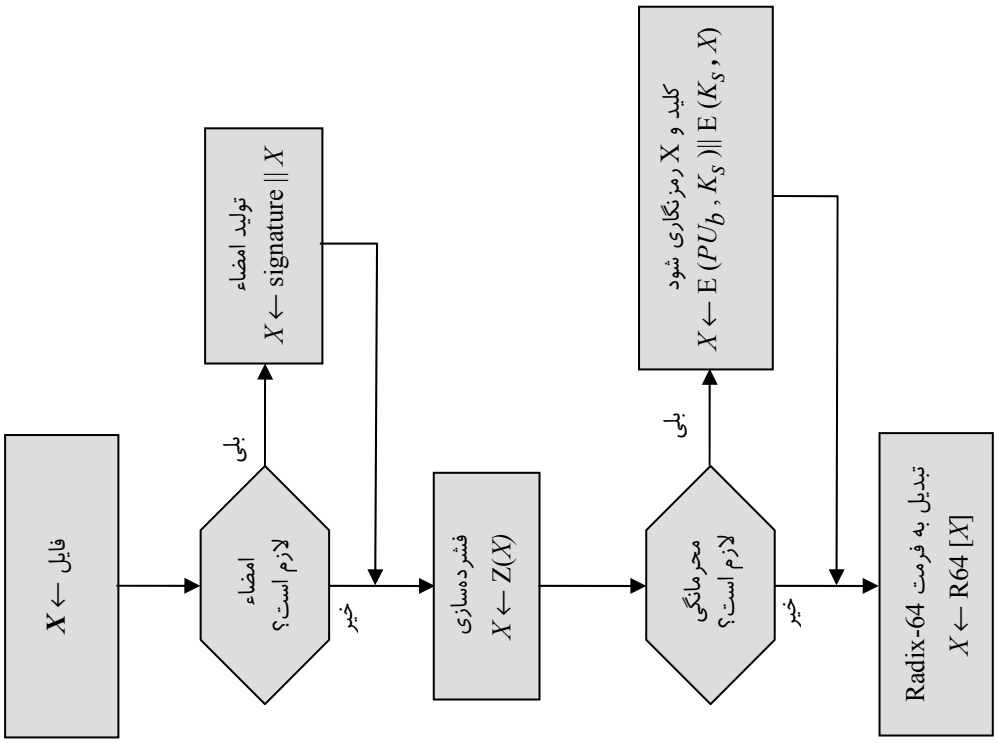
شکل ۲-۵ رابطهٔ بین چهار سرویسی که تا کنون مورد بحث واقع شد را نشان می‌دهد. در هنگام ارسال، اگر لازم باشد، با استفاده از کُد hash متن ساده و فشرده نشده پیام، یک امضاء تولید می‌شود. سپس متن سادهٔ پیام و امضاء (درصورت حضور) فشرده می‌شود. سپس اگر محرمانه‌سازی مورد نیاز باشد، بلوک (صورت فشرده شدهٔ متن ساده و یا صورت فشرده شدهٔ امضاء بعلاوهٔ متن ساده) رمزنگاری شده و در ابتدای آن کلید رمزنگاری متقارن رمزشده بتوسط کلید عمومی، قرار می‌گیرد. بالاخره کل بلوک به فرمت radix-64 در می‌آید.

در زمان دریافت، بلوک ورودی ابتدا از فرمت radix-64 بصورت باینری درمی‌آید. آنگاه اگر پیام رمزنگاری شده باشد، گیرنده کلید اجلاس را بازیابی نموده و پیام را رمزگشائی می‌کند. نتیجهٔ حاصل را باید از حالت فشردگی خارج کرد. اگر پیام امضاء شده باشد، گیرنده کُد hash انتقال یافته را استخراج کرده و آن را با کُد hash ناشی از محاسبات خود مقایسه می‌نماید.

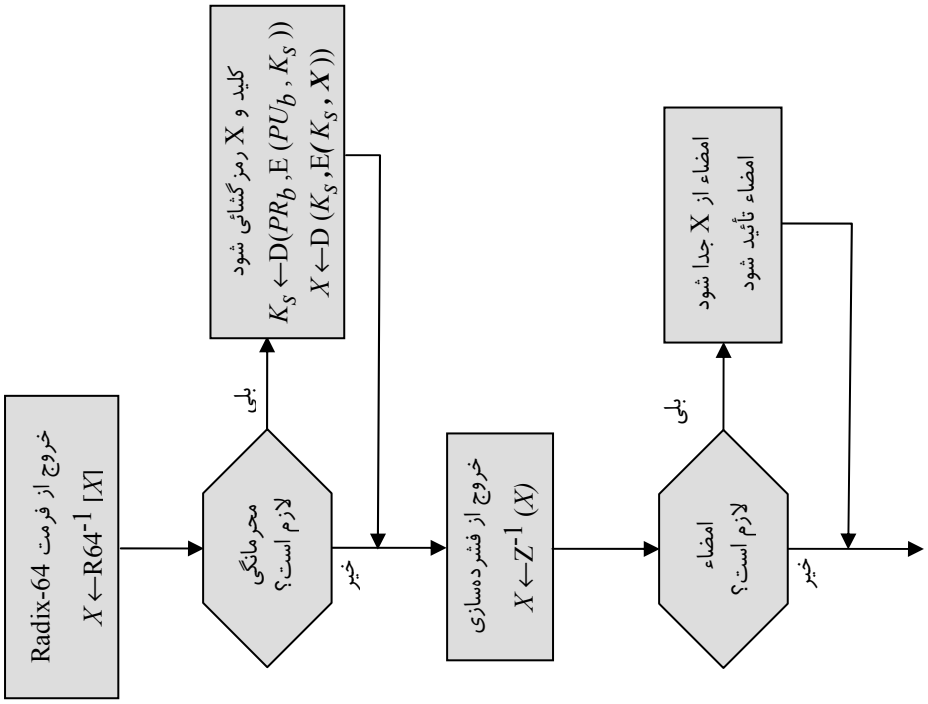
قطعه‌قطعه کردن و دوباره سرهم کردن دیتا

تسهیلات e-mail اغلب از نظر ماکزیمم طول پیام قابل ارسال دارای محدودیت‌اند. بعنوان مثال بسیاری از تسهیلات قابل دسترس از طریق اینترنت، حداکثر طول پیام را ۵۰,۰۰۰ اُکتت منظور کرده‌اند. هر پیامی که طویل‌تر از این مقدار باشد بایستی به قطعات کوچک‌تری تقسیم شده و هر قطعه بطور جداگانه ارسال شود.

برای غلبه بر این محدودیت، PGP پیامی را که بیش از حد طویل است بطور خودکار به قطعاتی که برای ارسال از طریق e-mail باندازهٔ کافی کوچک باشد تقسیم می‌کند. این قطعه‌قطعه کردن پس از همهٔ پردازش‌ها و از جمله تبدیل radix-64 انجام می‌شود. بنابراین عنصر کلید اجلاس و عنصر امضاء تنها یکبار و آنهم در ابتدای اولین قطعه حضور خواهند داشت. در سمت گیرنده، PGP بایستی تمام سرآیندهای e-mail را جدا نموده و قبل از انجام عملیات نشان داده شده در شکل۲-۵ب، تمام بلوک اولیه را سرهم نماید.



(الف) دیاگرام عمومی ارسال (از A)



(ب) دیاگرام عمومی دریافت (به B)

شکل ۲- ۵ ارسال و دریافت پیام‌های PGP

کلیدهای رمزنگاری و دسته‌کلیدها

PGP از چهار نوع کلید استفاده می‌کند: کلیدهای اجلاس که متقارن و یکبار مصرف‌اند، کلیدهای عمومی، کلیدهای خصوصی و کلیدهای متقارن مبتنی بر عبارت عبور (متعاقباً توضیح داده خواهد شد). در رابطه با این کلیدها، سه نیاز جداگانه را می‌توان تشخیص داد:

۱- برای تولید کلیدهای اجلاس غیرقابل پیش‌بینی، روشی مورد نیاز است.
۲- علاقه‌مندیم به یک کاربر اجازه دهیم تا چندین زوج کلید عمومی/کلید خصوصی را در اختیار داشته باشد. یک دلیل آن این است که شاید کاربر علاقه‌مند باشد تا هرچندگاه یکبار کلید خود را عوض کند. وقتی این اتفاق می‌افتد، هر پیامی که در مسیر وجود دارد بتوسط یک کلید خارج از رده ایجاد می‌شود. علاوه براین گیرندگان نیز تا زمانی که کلید جدید به آنها نرسد، تنها کلید قدیم را می‌شناسند. علاوه بر نیاز به تعویض کلید در طول زمان، یک کاربر ممکن است علاقه‌مند باشد تا در هر لحظه زوج کلیدهای متعددی را در اختیار داشته باشد تا با گروه‌های مختلف ارتباط داشته و یا بخواهد با تقسیم رمزنگاری بین کلیدهای مختلف، امنیت را ارتقاء بخشد. نتیجهٔ نهائی بحث این است که بالاخره یک ارتباط یک- به- یک بین کاربران و کلیدهای عمومی آنان وجود ندارد. بنابراین برای شناسائی کلیدهای مختلف، روشی مورد نیاز است.
۳- هر واحد PGP بایستی یک فایل که محتوی زوج‌های کلیدعمومی/خصوصی خود او است و فایل دیگری که محتوی کلیدهای عمومی طرف‌های مقابل است را نگهداری کند.

هر یک از این نیازها را به ترتیب بررسی می‌کنیم.

تولید کلید اجلاس

هر کلید اجلاس مربوط به فقط یک پیام بوده و تنها برای رمزنگاری و رمزگشائی آن پیام بکار می‌رود. بخاطر آوریدکه رمزنگاری/رمزگشائی پیام بتوسط یک الگوریتم رمزنگاری متقارن انجام می‌شود. CAST-128 و IDEA از کلیدهای ۱۲۸- بیتی استفاده کرده و 3DES از یک کلید ۱۶۸- بیتی استفاده می‌کند. برای این بحث، CAST-128 را در نظر می‌گیریم:

اعداد ۱۲۸- بیتی تصادفی با استفاده از خود CAST-128 تولید می‌شوند. ورودی تولیدکنندهٔ عدد تصادفی شامل یک کلید ۱۲۸- بیتی و دو بلوک ۶۴- بیتی است که دو بلوک اخیر متن ساده‌ای تلقی می‌شود که بایستی رمزنگاری گردد. با استفاده از مُود فیدبک رمز(CFB)، رمزنگار CAST-128 دو بلوک متن رمزشدهٔ ۶۴- بیتی تولید می‌کند که بهم متصل شده تا کلید اجلاس ۱۲۸- بیتی را درست کنند. الگوریتمی که برای این منظور بکار می‌رود، همان است که در ANSI X12.17 ذکر شده است.

«متن ساده» ورودی به تولیدکنندهٔ اعداد تصادفی که شامل دو بلوک ۶۴- بیتی است، خود از یک دنبالهٔ ۱۲۸- بیتی از اعداد تصادفی مشتق می‌شود. این اعداد بر مبنای حرکات کلید کاربر بوجود می‌آیند. هم از زمان استفاده از کلیدها و هم از خود کلیدهای بکار گرفته شده، برای تولید دنبالهٔ تصادفی استفاده می‌شود. بنابراین اگر کاربر کلیدهای صفحهٔ کلید را بطور اختیاری و با سرعت نرمال خود بکار بندد، یک ورودی «تصادفی» معقول تولید خواهد شد. این ورودی تصادفی همچنین با کلید اجلاس قبلی تولیدشده بتوسط CAST-128 ترکیب شده تا کلید ورودی به مولّد را ایجاد نماید. با توجه به خاصیت

مخلوط‌کنندگی مؤثر الگوریتم CAST-128، این عمل ردیفی از کلیدهای اجلاس که بطور چشمگیری غیرقابل پیش‌بینی هستند را بوجود خواهد آورد.

ضمیمهٔ ۵- ج، تکنیک تولید متغیرهای تصادفی در PGP را مورد بحث قرار داده است.

شناسه‌های کلیدها

همانطور که بحث شد، یک پیام رمزنگاری شده با یک فرم رمزشده از کلید اجلاس مورد استفاده همراهی می‌شود. خود کلید اجلاس بتوسط کلید عمومی گیرنده رمزنگاری می‌شود. بنابراین تنها گیرنده قادر به استخراج کلید اجلاس و در نتیجه استخراج پیام خواهد بود. اگر هر کاربر فقط از یک زوج کلید عمومی/خصوصی استفاده می‌کرد، آنگاه گیرنده بطور خودکار می‌دانست که باید از چه کلیدی برای رمزگشائی کلید اجلاس استفاده نماید که همان کلید خصوصی یکتای خود گیرنده است. ولی قبلاً بیان کردیم که لازم است هر کاربر دارای زوج کلیدهای عمومی/خصوصی متعددی باشد.

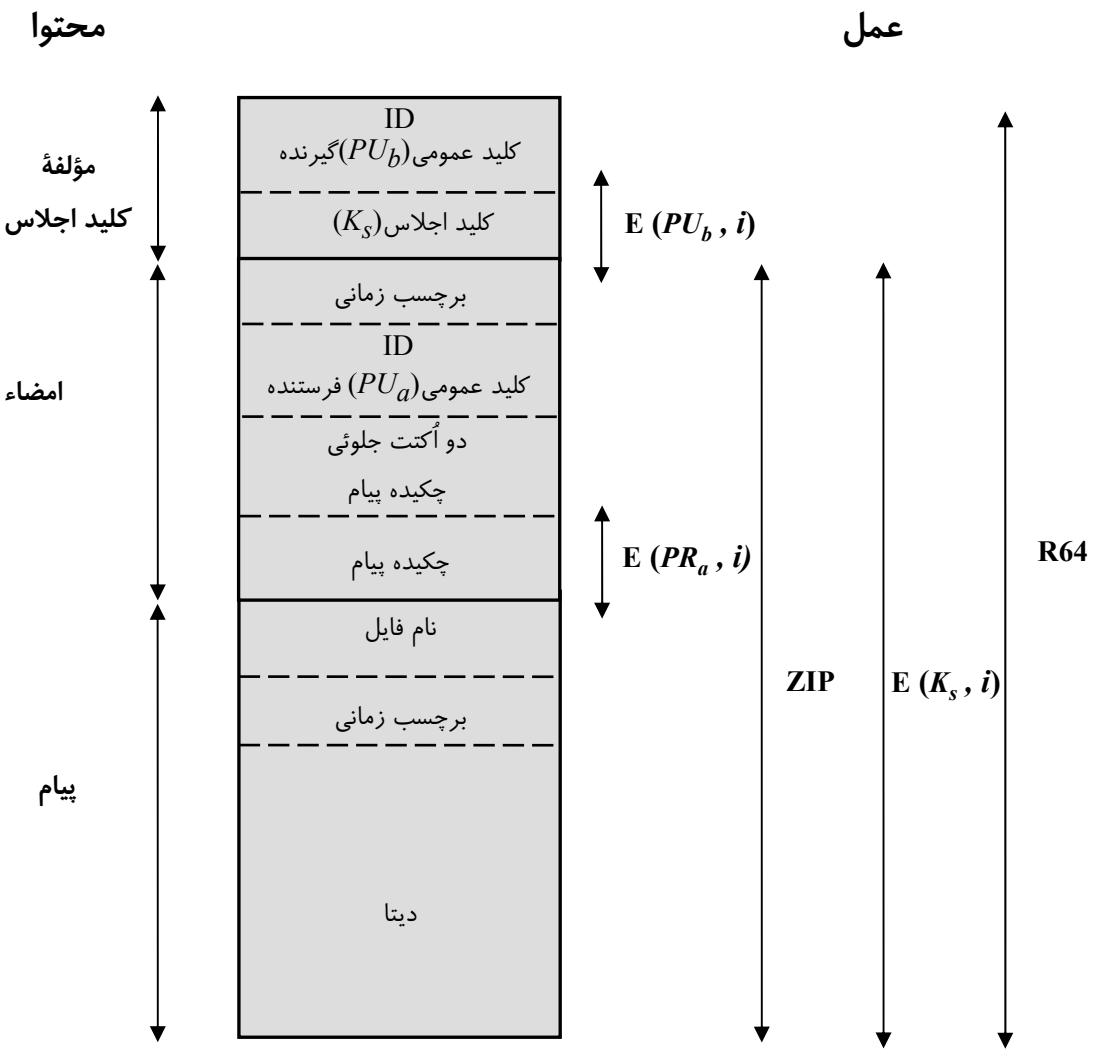
سؤال این است که گیرندهٔ پیام چگونه بداند که از کدام یک از کلیدهای عمومی او برای رمزنگاری کلید اجلاس استفاده شده است؟ یک راه حل ساده این است که کلید عمومی را بهمراه پیام ارسال کرد. دریافت‌کننده در این‌صورت می‌تواند تحقیق کند که این یکی از کلیدهای عمومی او بوده و به مراحل بعد برود. این روش قابل اجرا بوده ولی بی‌جهت فضای انتقال را اشغال می‌کند. یک کلید عمومی RSA ممکن است از صدها رقم اعشاری تشکیل شده باشد. راه حل دیگر این است که با هر کلید عمومی، شناسه‌ای را مرتبط کرد که حداقل در رابطه با یک کاربر یکتا باشد. یعنی ترکیب ID کاربر و ID کلید کافی باشد تا کلید را بصورت یکتا استخراج نمود. در این صورت تنها لازم است که ID مربوط به کلید که خیلی کوتاهتر از خود کلید است ارسال شود. اما این راه حل خود یک مشکل مدیریت و ایجاد سرباره را ایجاد می‌کند: IDهای مربوط به کلیدها بایستی تعیین شده و ذخیره شوند بطوری که هم فرستنده و هم گیرنده بتوانند از روی ID یک کلید به خود کلید عمومی دست یابند. این امر نیز پردردسر بنظر می‌رسد.

راه حلی که بتوسط PGP اتخاذ گردیده است این است که به هر کلید عمومی یک ID کلید تخصیص داده شود که، با احتمال بسیار زیاد، در محدودهٔ ID یک کاربر یکتا باشد. ID کلید مرتبط با هر کلید عمومی کم اهمیت‌ترین ۶۴- بیت آن کلید است. یعنی ID یک کلیدعمومی PU_a برابر $PU_a \bmod 2^{64}$ می‌باشد. این طول برای اینکه احتمال جعل یک ID کلید بسیار کم باشد، طولی معقول است.

یک ID کلید نیز برای امضاء دیجیتال PGP مورد نیاز است. چون یک ارسال‌کننده ممکن است یکی از چند کلید خصوصی را برای رمزنگاری چکیدهٔ پیام بکار برد، دریافت‌کننده بایستی بداند که از کدام کلید عمومی استفاده نماید. بهمین جهت مؤلفهٔ امضاء دیجیتال یک پیام شامل یک ID کلید ۶۴- بیتی مربوط به کلید عمومی مورد نیاز است. وقتی پیام دریافت شد، دریافت‌کننده ابتدا به دنبال تأئید اینکه ID کلید، مربوط به یکی از کلیدهای عمومی شناخته شدهٔ ارسال‌کننده بوده پرداخته، و سپس به دنبال تأئید امضاء می‌رود.

حال که مفهوم ID کلید را معرفی کردیم، می‌توانیم نگاه دقیق‌تری به فرمت یک پیام انتقال یافته که در شکل ۳-۵ نشان داده شده است بیندازیم. یک پیام شامل سه مؤلفه است: مؤلفهٔ پیام، مؤلفهٔ امضاء(اختیاری) و مؤلفهٔ کلید اجلاس (اختیاری).

مؤلفهٔ پیام شامل داده‌های اصلی که بایستی ذخیره و یا ارسال شود بوده و یک نام فایل و یک برچسب زمانی که زمان خلق پیام را تعیین می‌کند نیز با آن همراه است.



علائم اختصاری:

E (PUb , i) = رمزنگاری با کلید عمومی کاربر b

E (PR_a , i) = رمزنگاری با کلید خصوصی کاربر a

E (K_S , i) = رمزنگاری با کلید اجلاس

ZIP = تابع فشرده سازیZIP

R64 = تابع تبدیل Radix-64

شکل ۳- ۵ فرم عمومی پیام PGP (از A به B)

مولفهٔ امضاء شامل اقلام زیر است:

- **برچسب زمانی**: زمانی را نشان می‌دهد که پیام در آن لحظه امضاء شده است.
- **چکیدهٔ پیام**: چکیدهٔ ۱۶۰- بیتی SHA-1 پیام، که بتوسط کلید خصوصی امضاء ارسال‌کننده رمزنگاری شده است، را نشان می‌دهد. چکیده روی برچسب زمانی مؤلفهٔ امضاء که با بخش دیتای مؤلفهٔ پیام جمع رشته‌ای شده است، محاسبه می‌شود. قراردادن برچسب زمانی مؤلفهٔ امضاء در چکیدهٔ پیام، محافظت در مقابل حمله‌های از نوع بازخوانی را تضمین می‌کند. قرار ندادن نام فایل و برچسب زمانی مؤلفهٔ پیام، این اطمینان را ایجاد می‌کندکه امضاءهای

جداشده عیناً شبیه همان امضاءهای اضافه شده به اول پیام است. امضاءهای جداشده در فایلی جداگانه محاسبه می‌شوند که هیچ‌یک از میدان‌های سرآیند پیام را دارا نیستند.

- **دو اُکتت جلوئی چکیدهٔ پیام**: دریافت‌کننده را قادر می‌سازد تا بتواند تعیین کند که آیا کلید عمومی صحیح برای رمزگشائی چکیدهٔ پیام برای اعتبارسنجی بکار رفته است. این عمل با مقایسهٔ کپی متن سادهٔ اولین دو اُکتت، با اولین دو اُکتت رمزگشائی شدهٔ چکیدهٔ پیام انجام می‌شود. این اُکتت‌ها همچنین بعنوان یک FCS (Frame Check Sequence) ۱۶- بیتی برای پیام بکار می‌روند.
- **ID کلید مربوط به کلید عمومی فرستنده**: آن کلید عمومی را نشان می‌دهد که بایستی برای رمزگشائی پیام بکار رود، و بنابراین کلید خصوصی استفاده شده برای رمزنگاری پیام را نیز مشخص می‌کند.

مؤلفهٔ پیام و مؤلفهٔ اختیاری امضاء را می‌توان با استفاده از ZIP فشرده نموده و با یک کلید اجلاس نیز رمزنگاری کرد.

مؤلفهٔ کلید اجلاس شامل کلید اجلاس و شناسهٔ کلید عمومی دریافت‌کننده است که بتوسط فرستنده برای رمزنگاری کلید اجلاس از آن استفاده شده است.

تمام بلوک معمولاً بتوسط کُدینگ radix-64 کُد می‌شود.

دسته‌کلیدها

دیدیم که IDهای کلیدها در عملیات PGP نقش اساسی داشته و دو ID مربوط به دو کلید در هر پیام PGP قرار می‌گیرند تا هم محرمانگی و هم اعتبارسنجی را فراهم‌آورند. این کلیدها لازم است تا بصورت سیستماتیک سازمان‌دهی و ذخیره گردند تا بصورت بهره‌ور و مؤثر بتوسط طرف‌های درگیر مورد استفاده قرار گیرند. روشی که در PGP از آن استفاده می‌شود این است که یک زوج پایگاه داده در هر گره ایجاد شود که یکی از این پایگاه‌ها جفت کلید عمومی/خصوصی متعلق به آن گره را ذخیره کرده و پایگاه دیگر کلیدهای عمومی سایر کاربران شناخته شده برای این گره را نگهداری نماید. این ساختارها را بترتیب دسته‌کلید- خصوصی و دسته‌کلید- عمومی نامند.

شکل ۴-۵ ساختار عمومی یک **دسته‌کلید- خصوصی** را نشان می‌دهد. دسته‌کلید را می‌توان بصورت جدولی در نظر گرفت که در آن هر ردیف نمایشگر یکی از جفت کلیدهای عمومی/خصوصی متعلق به آن کاربر است. هر ردیف شامل مؤلفه‌های زیر است:

- **برچسب زمانی**: تاریخ و زمانی که این جفت کلید تولید شده است.
- **ID کلید**: کم ارزش‌ترین ۶۴- بیت کلید عمومی آن مؤلفه.
- **کلید عمومی**: بخش کلید- عمومی این جفت.
- **کلید خصوصی**: بخش کلید- خصوصی این جفت. این میدان رمزنگاری شده است.
- **ID کاربر**: معمولاً این بخش آدرس e-mail کاربر است (مثل movahed730@yahoo.com). ولی کاربر می‌تواند برای هر جفت کلید، نام متفاوتی را انتخاب نماید (مثل MOV، mov، Mmovahed و غیره) و یا همان ID کاربر را بیش از یکبار تکرار کند.

دسته‌کلید- خصوصی را می‌توان برحسب ID کاربر و یا ID کلید تنظیم کرد. بعداً اهمیت این دو تنظیم را مشاهده خواهیم کرد.

دسته‌کلید- خصوصی

Timestamp	Key ID	Public Key	Encrypted Private Key	User ID
•	•	•	•	•
•	•	•	•	•
•	•	•	•	•
T_i	$PU_i \bmod 2^{64}$	PU_i	$E(H(P_i), PR_i)$	User i
•	•	•	•	•
•	•	•	•	•
•	•	•	•	•

دسته‌کلید- عمومی

Timestamp	Key ID	Public Key	Owner Trust	User ID	Key Legitimacy	Signature(s)	Signiture Trust(s)
•	•	•	•	•	•	•	•
•	•	•	•	•	•	•	•
•	•	•	•	•	•	•	•
T_i	$PU_i \bmod 2^{64}$	PU_i	trust_flag $_i$	User i	trust_flag $_i$		
•	•	•	•	•	•	•	•
•	•	•	•	•	•	•	•
•	•	•	•	•	•	•	•

شکل ۴-۵ ساختار عمومی دسته‌کلیدهای- عمومی و خصوصی

اگرچه هدف این است که دسته‌کلید- خصوصی تنها روی ماشین کاربر که این جفت کلیدها را تولید کرده است و صاحب آن است ذخیره شود و تنها در دسترس همان کاربر قرار گیرد، ولی منطقی است که اندازهٔ کلیدهای خصوصی را تا حد ممکن مخفی نگاه داشته و حفاظت کرد. بهمین جهت خود کلید خصوصی در دسته‌کلید ذخیره نمی‌شود. بلکه این کلید با استفاده از CAST-128 (یا IDEA یا 3DES) رمزنگاری می‌گردد. روش عمل چنین است:

۱- کاربر یک عبارت عبور (passphrase) که قرار است برای رمزنگاری کلیدهای خصوصی بکار رود را انتخاب می‌کند.
۲- وقتی سیستم یک جفت کلید عمومی/خصوصی جدید با استفاده از RSA تولید می‌کند، از کاربر نسبت به عبارت عبور سؤال می‌نماید. با استفاده از SHA-1 یک کُد ۱۶۰- بیتی از این عبارت عبور درست شده و خود عبارت عبور معدوم می‌شود.
۳- سیستم، کلید خصوصی را با استفاده از CAST-128 و استفاده از ۱۲۸- بیت کُد hash بعنوان کلید، رمزنگاری می‌نماید. کُد hash متعاقباً معدوم شده و کلید خصوصی رمزشده در دسته‌کلید- خصوصی ذخیره می‌گردد.

متعاقباً وقتی یک کاربر دسته‌کلید- خصوصی را برمی‌دارد تا یک کلید خصوصی را انتخاب کند، او بایستی عبارت عبور را ارائه نماید. PGP کلید خصوصی رمزنگاری شده را بازیابی نموده، کُد hash نظیر عبارت عبور را تولید کرده و کلید خصوصی رمزنگاری شده را با استفاده از CAST-128 و کُد hash رمزگشائی می‌نماید.

این یک روش بسیار ساده و مؤثر است. همانند هر سیستم مبتنی بر کلمهٔ عبور، امنیت این سیستم وابسته به امنیت کلمهٔ عبور است. برای جلوگیری از وسوسهٔ نوشتن کلمهٔ عبور بر روی کاغذ، کاربر از یک عبارت عبور استفاده می‌کندکه بسادگی قابل حدس نبوده ولی بسادگی قابل بخاطر سپردن است.

شکل ۴-۵ همچنین ساختار کلی **دسته‌کلید- عمومی** را نشان می‌دهد. این پایگاه داده برای ذخیره کردن کلیدهای عمومی سایر کاربران که نزد این کاربر شناخته شده است بکار می‌رود. فعلاً اجازه دهید بعضی از میدان‌های موجود نشان داده شده در جدول را فراموش کرده و به توصیف میدان‌های زیر بپردازیم:

- **برچسب زمانی**: تاریخ/ زمان خلق این مورد را نشان می‌دهد.
- **ID کلید**: کم اهمیت‌ترین ۶۴- بیت کلید عمومی این مورد.
- **کلید عمومی**: کلید عمومی این مورد.
- **ID کاربر**: صاحب این کلید را مشخص می‌کند. ممکن است چندین ID کاربر مرتبط با یک کلید عمومی منفرد باشند.

دسته‌کلید- عمومی را می‌توان یا بر حسب ID کاربر و یا بر حسب ID کلید رده‌بندی نمود. نیاز به هردو روش را متعاقباً مشاهده خواهیم کرد.

حال در موقعیتی هستیم که نشان دهیم چگونه این دسته‌کلیدها در ارسال و دریافت پیام بکار می‌روند. بمنظور سهولت، از فشرده‌سازی و تبدیل radix-64 در این بحث صرفنظر می‌کنیم. ابتدا ارسال پیام (شکل ۵-۵) را در نظر گرفته و فرض کنید که پیام بایستی هم امضاء شده و هم رمزنگاری شود. PGP ارسال‌کنندهٔ پیام قدم‌های زیر را برمی‌دارد:

۱- امضاء پیام

الف- PGP کلید خصوصی ارسال‌کنندهٔ پیام را از دسته‌کلید- خصوصی او، با استفاده از اندیس your_userid، استخراج می‌کند. اگر your_userid در فرمان وجود نداشته باشد، اولین کلید خصوصی دسته‌کلید انتخاب می‌شود.

ب- PGP عبارت عبور کاربر را از او سؤال کرده تا کلید خصوصی رمزنشده را بازیابی کند.

ج- مؤلفهٔ امضاء پیام ساخته می‌شود.

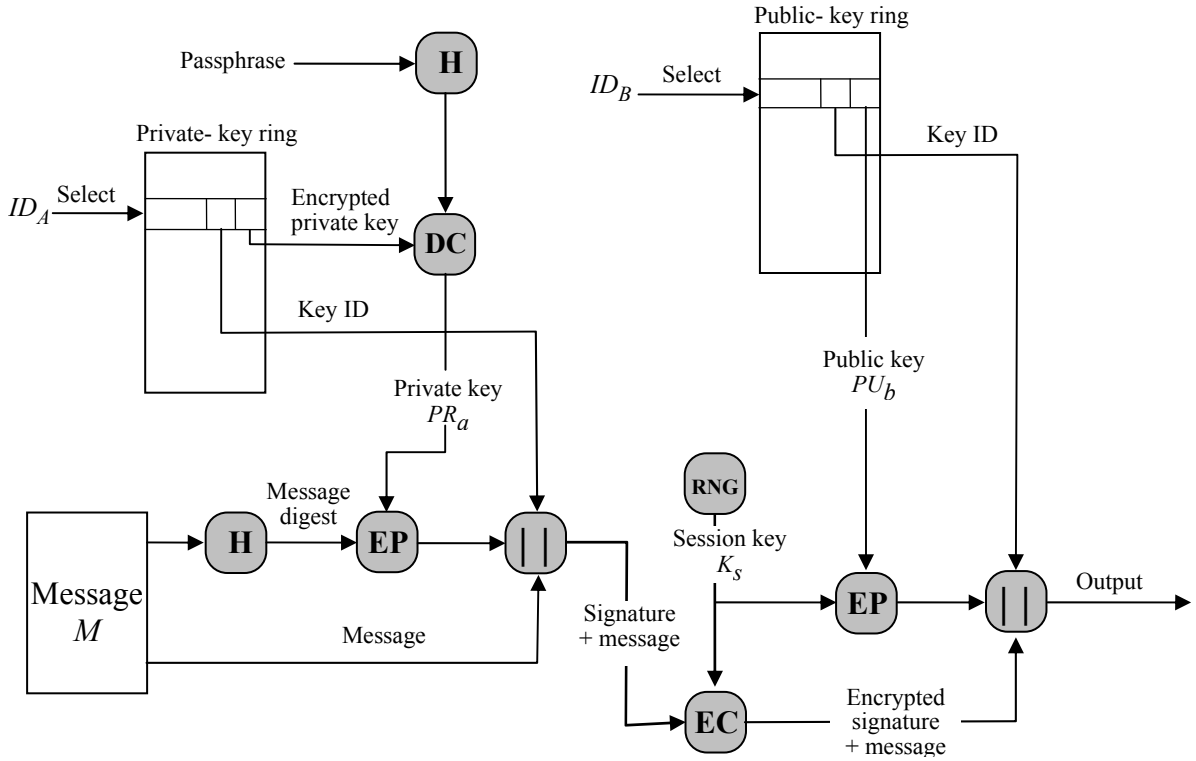
۲- رمزنگاری پیام

الف- PGP یک کلید اجلاس تولید کرده و پیام را رمزنگاری می‌کند.

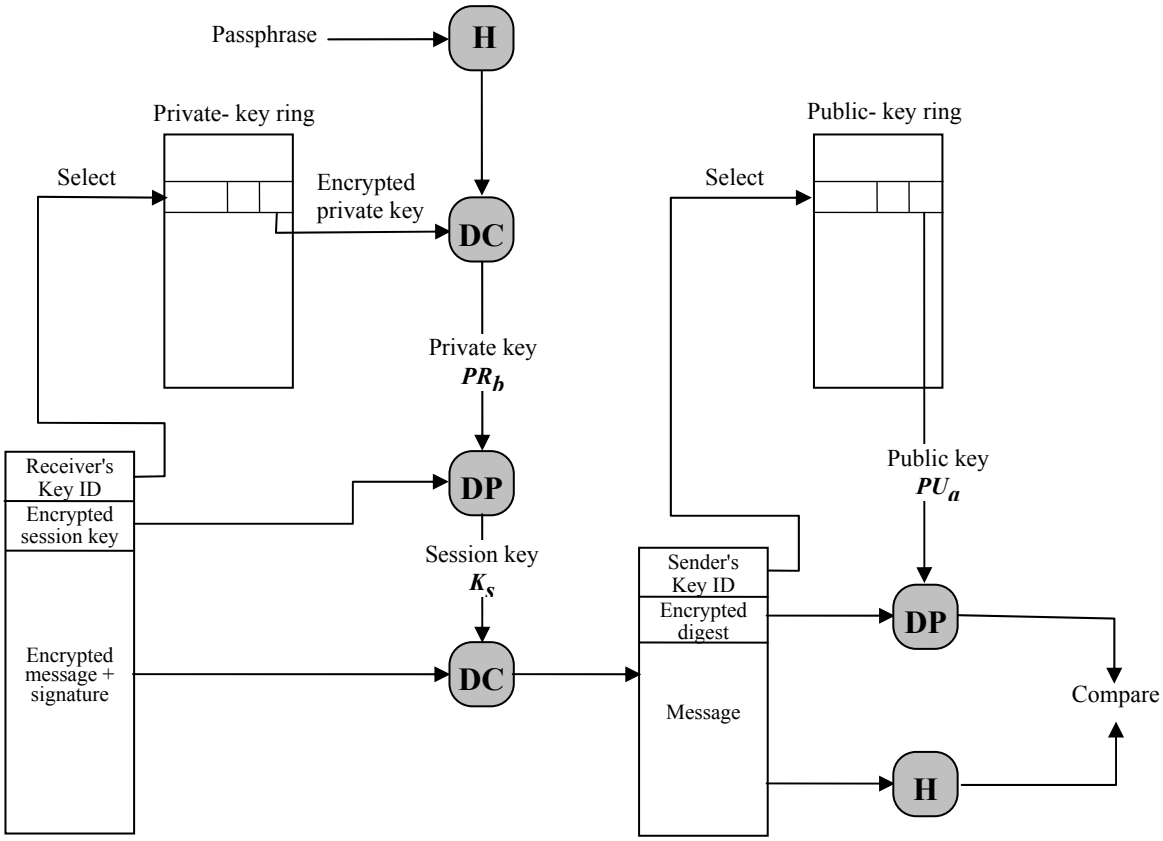
ب- PGP کلید عمومی دریافت‌کنندهٔ پیام را، با استفاده از اندیس her_userid، از دسته‌کلید- عمومی کاربر استخراج می‌کند.

ج- مؤلفهٔ کلید اجلاس پیام ساخته می‌شود.

PGP واحد دریافت‌کنندهٔ پیام قدم‌های زیر را برمی‌دارد (شکل ۶-۵):



شکل ۵-۵ تولید پیام PGP (از کاربر A به کاربر B . بدون فشرده‌سازی و تبدیل Radix-64)



شکل ۶-۵ دریافت پیامPGP (از کاربر A به کاربر B . بدون فشرده‌سازی و تبدیل Radix-64)

۱- رمزگشائی پیام

الف- PGP کلید خصوصی دریافت‌کننده را، با استفاده از میدان ID کلید در مؤلفهٔ امضاء کلید در پیام بعنوان اندیس، از دسته‌کلید- خصوصی استخراج می‌کند.

ب- PGP از کاربر عبارت عبور را سؤال کرده تا کلید خصوصی رمزنگاری نشده را استخراج نماید.

ج- PGP آنگاه کلید اجلاس را بازیابی نموده و پیام را رمزگشائی می‌نماید.

۲- اعتبارسنجی پیام

الف- PGP کلید عمومی ارسال‌کننده را، با استفاده از میدان ID کلید در مؤلفهٔ امضاء کلید در پیام بعنوان اندیس، از دسته‌کلید- عمومی کاربر استخراج می‌کند.

ب- PGP چکیدهٔ پیام انتقال یافته را بازیابی می‌کند.

ج- PGP چکیدهٔ پیام برای پیام دریافت شده را محاسبه کرده و آن را با چکیدهٔ پیام انتقال یافته مقایسه و اعتبار آن را تأئید می‌نماید.

مدیریت کلید- عمومی

همانطور که از بحث‌های انجام شده دیده می‌شود، PGP شامل یک سری عملیات و فرمت‌های هوشیار، بهره‌ور و درهم بافته‌ایست که یک سرویس مؤثر محرمانگی و اعتبارسنجی را ایجاد می‌کند. برای اینکه این سیستم کامل باشد، یک عمل نهائی دیگر نیز بایستی مورد توجه قرار گیرد که آن مدیریت کلید- عمومی است. اسناد PGP اهمیت این مقوله را چنین بیان می‌کند:

> حفاظت کلیدهای عمومی از دست کسانی که می‌خواهند آنها را به چنگ آورند، مشکل‌ترین وظیفه درکاربردهای عملی کلید-عمومی است. این مورد « پاشنهٔ آشیل » رمزنگاری کلید-عمومی بوده و پیچیدگی‌های نرم‌افزاری زیادی با حل این معضل عجین می‌شود.

PGP ساختاری را برای حل این مسأله ایجاد کرده است که انتخاب‌های متعددی را نیز شامل می‌شود. چون PGP بمنظور استفاده در محیط‌های مختلف رسمی و غیررسمی طراحی شده است، هیچ روش مدیریتی سفت و سختی برای مدیریت کلید- عمومی، همانند آنچه که در S/MIME که بعداً در همین فصل به آن خواهیم پرداخت وجود دارد، در نظر نگرفته است.

روش‌های مدیریت کلید- عمومی

اصل مشکل این است: کاربر A بایستی با استفاده از PGP، یک دسته‌کلید- عمومی درست کندکه کلیدهای عمومی سایر کاربران که با او ارتباط دارند در آن وجود داشته باشد. فرض کنید که دسته‌کلید A شامل یک کلید عمومی مربوط به B است که در حقیقت صاحب آن کلید، C است. این امر برای مثال وقتی اتفاق می‌افتد که A کلید را از یک سیستم تابلوی اعلانات (BBS) متعلق به B که کلید عمومی را در آن جا داده بوده است بردارد، ولی کلید بتوسط C تعویض شده باشد. نتیجهٔ امر این است که اکنون دو تهدید وجود دارد. اول اینکه C می‌تواند پیام‌هائی را برای A ارسال کرده و امضاء B را جعل نماید، بطوری که A تصور کند که پیام از طرف B آمده است. دوم اینکه هر پیام رمزشده از A به B می‌تواند بتوسط C خوانده شود.

برای به حداقل رساندن خطر وجود کلیدهای عمومی جعلی در دسته‌کلید یک کاربر، روش‌های متعددی تجربه شده است. فرض کنید A بخواهد یک کلید عمومی قابل اعتماد برای B بدست آورد. برخی از روش‌های پیشنهادی چنین‌اند:

۱- کلید را بصورت فیزیکی از B بگیرد. B می‌تواند کلید عمومی (PU_b) خود را روی یک دیسکت ذخیره کرده و آن را به A بدهد. A می‌تواند بعداً از روی دیسکت کلید را وارد سیستم خود نماید. این روش بسیار امن بوده ولی دارای محدودیت‌های عملی واضحی است.

۲- بتوسط تلفن کلید را تأئید نماید. اگر A بتواند از پشت تلفن B را بشناسد، می‌تواند به B زنگ زده و از وی بخواهد تا کلید را در فرمت radix-64 برای او دیکته نماید. راه حل عملی‌تر اینکه B می‌تواند کلید خود را بتوسط e-mail برای A ارسال کند. A می‌تواند با استفاده از PGP یک چکیدهٔ ۱۶۰- بیتی SHA-1 از کلید تهیه کرده و آن را با فرمت هگزادسیمال نشان دهد که این را «اثرانگشت» کلید گویند. A می‌تواند بعداً به B زنگ زده و از او بخواهد تا اثر انگشت را از پشت تلفن برای او دیکته کند. اگر دو اثر انگشت نزد A و B با هم تطبیق داشته باشند، کلید تأئید می‌شود.

۳- کلید عمومی B را از یک نفر که مورد اعتماد طرفین است، مثل D، دریافت کند. برای این منظور، معرف D یک گواهی‌نامهٔ امضاء شده را فراهم می‌آورد. گواهی‌نامه شامل کلید عمومی B، زمان تولید کلید و مدت اعتبار کلید خواهد بود. D یک چکیده SHA-1 از این گواهی‌نامه تهیه کرده، آن را با کلید خصوصی خود رمزنگاری نموده و امضاء را به گواهی الصاق می‌کند. چون فقط D می‌توانسته است امضاء را تولیدکند، هیچ کس دیگر نمی‌تواند یک کلید عمومی جعلی تهیه کرده و وانمود کند که بتوسط D امضاء شده است. گواهی‌نامهٔ امضاء شده می‌تواند بتوسط B یا D مستقیماً ارسال شده و یا در یک تابلوی اعلانات (BBS) نصب گردد.

۴- کلید عمومی B را از یک مقام مسئول صدور مجوز مورد اعتماد دریافت کند. باز هم یک گواهی‌نامه برای کلید عمومی تولید شده و بتوسط مقام مسئول امضاء می‌شود. A می‌تواند به مقام مسئول دسترسی یافته و با استفاده از ID خود یک گواهی‌نامهٔ امضاء شده را دریافت نماید.

برای موارد ۳ و ۴، A بایستی یک کپی از کلید عمومی معرف را داشته و اطمینان یابد که این کلید معتبر است. بالاخره این بر عهدهٔ A است تا سطحی از اعتماد را نسبت به کسی که بعنوان معرف عمل می‌کند، داشته باشد.

استفاده از اعتماد (Trust)

اگرچه PGP هیچ گونه دستورالعملی برای ایجاد مسئول تأئید و یا برقراری اعتماد ندارد، ولی خود روش‌های مناسبی برای استفاده از اعتماد، ارتباط دادن اعتماد با کلیدهای عمومی و بکارگیری اعتماد را فراهم آورده است.

ساختار اصلی چنین است: هر فقرهٔ موجود در دسته‌کلید- عمومی، همانطور که در بخش قبل توضیح داده شد، یک گواهی‌نامهٔ کلید- عمومی است. به همراه هر فقره گواهی‌نامهٔ کلید- عمومی یک **میدان مشروعیت کلید (key legitimacy field)** وجود دارد که نشان می‌دهد PGP تا چه حدّی نسبت به تعلق این کلید به کاربر مربوطه اعتماد دارد. هر چقدر سطح اعتماد بالاتر باشد، ارتباط کلید با ID کاربر مستحکم‌تر است. این میدان بتوسط PGP محاسبه می‌گردد. همچنین در ارتباط با هر فقره گواهی‌نامهٔ کلید- عمومی، هیچ و یا چند امضاء وجود دارد که صاحب دسته‌کلید آنها را به منظور تأئید این گواهی‌نامه جمع‌آوری کرده است. بنوبهٔ خود در ارتباط با هر امضاء یک **میدان اعتماد به امضاء (signature trust field)** وجود دارد که درجهٔ اعتماد کاربر PGP نسبت به امضاءکننده، برای تأئید کلیدهای عمومی را

نشان می‌دهد. میدان مشروعیت کلید از مجموعهٔ میدان‌های اعتماد به امضاء برای هر گواهی‌نامه مشتق شده است. بالاخره هر فقره گواهی‌نامهٔ کلید- عمومی، یک کلید عمومی مرتبط با یک دارندهٔ مشخص کلید را تعریف کرده و برای آن یک **میدان اعتماد به صاحب کلید (owner trust field)** در نظر گرفته شده است که نشان می‌دهد تا چه حد این کلید عمومی برای امضاء سایر گواهی‌نامه‌های کلید- عمومی مورد اعتماد است. این سطح اعتماد بتوسط کاربر تخصیص داده می‌شود. میدان‌های اعتماد به امضاء را می‌توان کپی‌های ذخیره شده میدان اعتماد به صاحب کلید فقره‌های دیگر دسته‌کلید دانست.

سه میدانی که در بخش قبل به آنها اشاره شد، هریک در ساختاری که به آن بایت پرچم اعتماد (trust flag byte) گویند قرار دارند. محتوای پرچم اعتماد برای هریک از این سه مورد ذکرشده در جدول ۲-۵ نشان داده شده است. فرض کنید که ما با دسته‌کلید- عمومی کاربر A سروکار داریم. عملیات اعتمادسازی را می‌توان چنین توصیف کرد:

۱- وقتی A یک کلید عمومی جدید را در دسته‌کلید- عمومی وارد می‌کند، PGP بایستی اندازه‌ای را به پرچم اعتماد مربوط به صاحب این کلید عمومی تخصیص دهد. اگر صاحب این کلید A است و بنابراین این کلید عمومی در دسته کلید- خصوصی او نیز قرار می‌گیرد، آنگاه یک اندازهٔ *اعتماد کامل* بصورت اتوماتیک به میدان اعتماد اختصاص می‌یابد. در غیر اینصورت PGP از A نسبت به سطح اعتمادی که بایستی به این صاحب کلید تخصیص یابد سؤال می‌کند و A بایستی مقدار مورد نظر خود را وارد کند. کاربر A می‌تواند مشخص نماید که این صاحب کلید ناشناخته، غیرقابل اعتماد، تا حدودی قابل اعتماد و یا کاملاً مورد اعتماد است.

۲- وقتی یک کلید عمومی جدید وارد می‌شود، یک یا چند امضاء ممکن است به آن متصل باشد. امضاءهای دیگری نیز ممکن است در آینده به آن اضافه شوند. وقتی یک امضاء برای یک فقره گواهی‌نامهٔ کلید- عمومی وارد می‌شود، PGP در دسته‌کلید- عمومی جستجو کرده تا ببیند که آیا امضاءکننده در بین صاحبان کلیدهای عمومی شناخته شده هست یا خیر. اگر جواب مثبت باشد، اندازهٔ OWNERTRUST برای این دارندهٔ کلید به میدان SIGTRUST برای این امضاء تخصیص می‌یابد. اگرجواب منفی باشد، مقدار *کاربر ناشناخته* به آن تخصیص می‌یابد.

۳- اندازهٔ میدان مشروعیت کلید بر مبنای میدان‌های اعتماد به امضاء موجود در یک فقره محاسبه می‌گردد. اگر حداقل یک امضاء دارای اندازهٔ *اعتماد کامل* باشد، آنگاه مشروعیت کلید اندازهٔ کامل می‌گیرد. در عیر اینصورتPGP یک جمع تراز داده شده از مقادیر اعتماد را محاسبه خواهد کرد. یک وزن $1/X$ به امضاءهائی که همیشه مورد اعتمادند و یک وزن $1/Y$ به امضاءهائی که معمولاً قابل اعتمادند داده می‌شود. X و Y پارامترهائی هستند که بتوسط کاربر پیکربندی می‌شوند. وقتی جمع ترازهای داده شده معرف‌های یک ترکیب کلید/UserID به ۱ برسد، این پیوند قابل اعتماد تلقی شده و مشروعیت کلید کامل فرض می‌شود. بنابراین در غیاب اعتماد کامل، حداقل X امضاء که همیشه مورد اعتماد بوده و یا Y امضاء که معمولاً قابل اعتمادند و یا ترکیبی از آنها مورد نیاز خواهد بود.

هرچندگاه یکبار، PGP دسته‌کلید- عمومی را مورد پردازش قرار داده تا اقلام آن را با هم سازگار نماید. در واقع این یک پردازش از بالا به پائین است. برای هر میدان OWNERTRUST، PGP دسته‌کلید را برای تمام امضاءهای تأئیدشده بتوسط صاحب آن جستجو کرده و میدان SIGTRUST را بروزرسانی نموده تا معادل میدان OWNERTRUST گردد. این پردازش ابتدا از کلیدهائی شروع می‌شود که برای آنها اعتماد کامل وجود دارد. آنگاه تمام میدان‌های KEYLEGIT بر اساس امضاءهای جداشده محاسبه می‌گردد.

جدول ۲-۵ محتویات بایت پرچم اعتماد (Trust Flag Byte)

(الف) اعتماد تخصیص داده‌شده به صاحب کلیدعمومی (بعد از میدان کلید ظاهر شده و بتوسط کاربر تعریف می‌شود)	(ب) اعتماد تخصیص داده‌شده به زوج USER ID/ Public Key (بعد از میدان User ID ظاهر شده و بتوسط PGP محاسبه می‌شود)	(ج) اعتماد تخصیص داده‌شده به امضاء (بعد از میدان امضاء ظاهرشده وکپی ذخیره‌شده OWNERTRUST برای این امضاءکننده است)
میدان OWNERTRUST - اعتماد تعیین نشده - کاربر ناشناخته - معمولاً برای امضاء کلیدهای دیگر مورد اعتماد نیست - معمولاً برای امضاء کلیدهای دیگر مورد اعتماد است - همیشه برای امضاء کلیدهای دیگر مورداعتماد است - این کلید در دسته کلید خصوصی وجود دارد (اعتماد کامل) بیت BUCKSTOP - این بیت در صورتی set است که این کلید در دسته کلید خصوصی ظاهر شده باشد	میدان KEYLEGIT - اعتماد نامشخص و یا تعیین نشده - مالکیت کلید مورد اعتماد نیست - اعتماد نسبی به مالکیت کلید - اعتماد کامل به مالکیت کلید بیت WARNONLY - این بیت در صورتی set است که کاربر بخواهد در صورت استفاده از یک کلید رمزنگاری که کاملاً معتبر نیست تنها به او هشدار داده شود	میدان SIGTRUST - اعتماد تعیین نشده - کاربر ناشناخته - معمولاً برای امضاء کلیدهای دیگر مورد اعتماد نیست - معمولاً برای امضاء کلیدهای دیگر مورد اعتماد است - همیشه برای امضاء کلیدهای دیگر مورد اعتماد است - این کلید در دسته کلید خصوصی وجود دارد (اعتماد کامل) بیت CONTIG - این بیت در صورتی set است که امضاء به یک مسیرپیوسته مورد اعتماد، که نهایتاً به صاحب دسته کلید کاملاً معتمد باز می‌گردد، مربوط شود

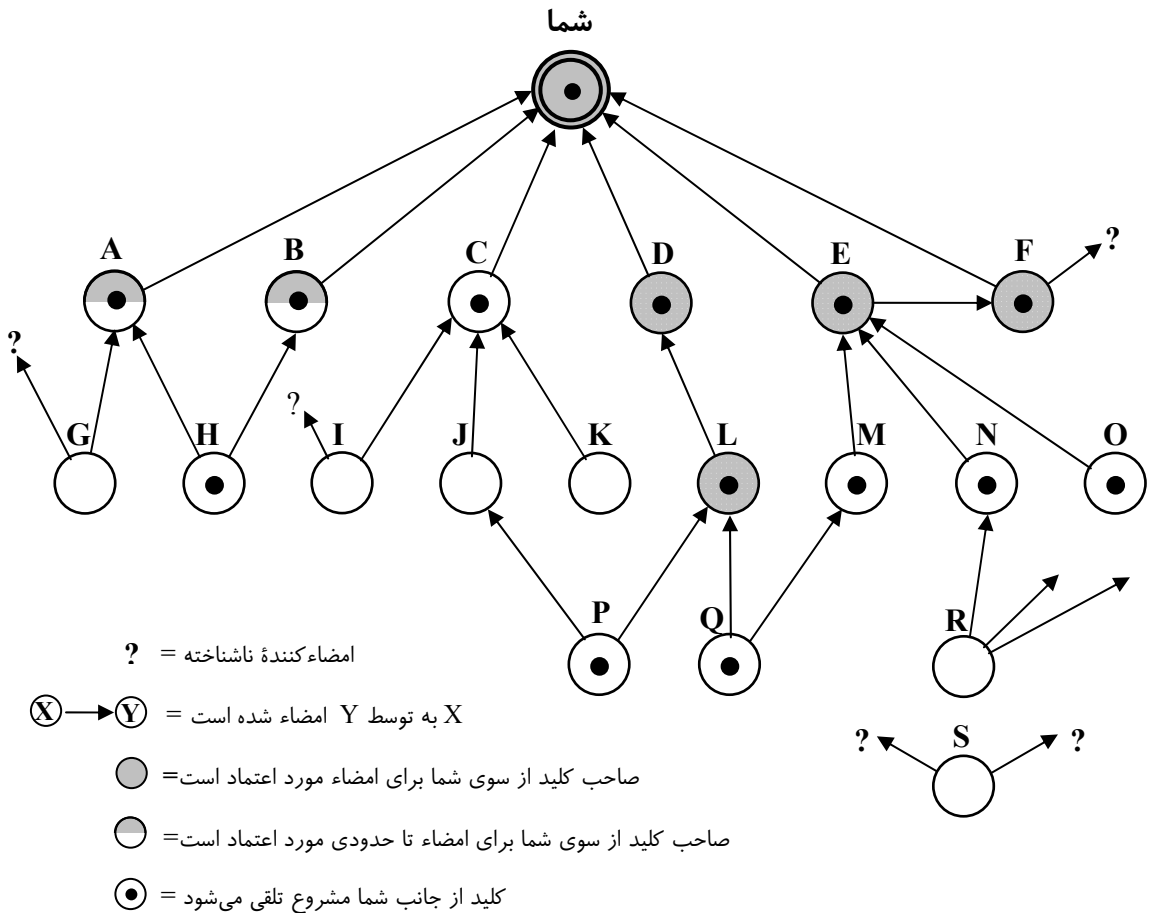
شکل ۷-۵ مثالی از نحوهٔ ارتباط اعتماد به امضاء، به مشروعیت کلید را نشان می‌دهد. در این شکل ساختار یک دسته کلید- عمومی نشان داده شده است. کاربر تعدادی از کلیدهای عمومی را جمع کرده است که برخی از آنها مستقیماً از صاحبان آنها و بعضی دیگر از شخص ثالثی که سِرور کلید است اخذ شده است.

گره‌ای که با عنوان "**شما**" نشان داده شده است به فقره‌ای در دسته‌کلید- عمومی اشاره می‌کند که نظیر این کاربر است. این کلید مشروع بوده و اندازهٔ OWNERTRUST آن اعتماد کامل است. هر گرهٔ دیگری در دسته‌کلید دارای یک اندازهٔ OWNERTRUST تعریف نشده بوده مگر اینکه اندازهٔ دیگری از طرف کاربر برای آن تعیین شده باشد. در این مثال، کاربر مشخص کرده است که همیشه به کاربران D و E و F و L برای امضاء سایر کلیدها اعتماد دارد. این کاربر به کاربران A و B برای امضاء سایر کلیدها، تا حدودی اعتماد دارد.

بنابراین میزان سایه‌دار بودن هر گره در شکل ۷-۵ نمایش دهندهٔ سطح اعتماد تخصیص داده شده بتوسط این کاربر به آن گره است. ساختار درختی نشان می‌دهد که کدام کلیدها بتوسط کدام کاربران امضاء شده اند. اگر یک کلید بتوسط کاربری امضاء شده است که کلیدش در دسته‌کلید وجود دارد، یک پیکان کلید امضاءشده را به امضاءکننده متصل کرده است. اگر کلید بتوسط کاربری امضاء شده است که کلید خود او در دسته‌کلید نیست، یک پیکان کلید امضاءشده را به یک علامت سؤال متصل کرده است که مفهوم آن این است که هویت امضاءکننده برای کاربر ناشناس است.

نکات چندی در شکل ۷-۵ نمایش داده شده است :

۱- توجه کنید که تمام کلیدهائی که صاحبان آنها کاملاً و یا بطور نسبی مورد اعتماد این کاربر بوده‌اند، بجز گرهٔ L، بتوسط این کاربر امضاء شده‌اند. همانطور که حضور گرهٔ L نشان می‌دهد، چنین امضائی از طرف کاربر همیشه ضروری نیست، اما در عمل، بیشتر کاربران محتمل است که اکثر کلیدهای کاربران مورد اعتماد خود را امضاء کنند. بعنوان مثال اگرچه کلید E قبلاً از طرف معرف مورد اعتماد F امضاء شده است، کاربر به انتخاب خود ترجیح داده است که خود هم کلید E را مستقیماً امضاء کند.



شکل ۷-۵ مثالی از مدل اعتماد PGP

۲- فرض می‌کنیم که دو امضاء نسبتاً مورد اعتماد برای تأئید یک کلید کافی باشد. در این صورت کلید کاربر H بتوسط PGP مشروع تلقی می‌گردد زیرا بتوسط A و B که هردوی آنها نسبتاً مورد اعتماد هستند، امضاء شده است.

۳- یک کلید ممکن است مشروع تلقی گردد زیرا بتوسط یک امضاءکننده کاملاً مورد اعتماد و یا دو امضاءکنندهٔ نسبتاً قابل اعتماد امضاء شده است ولی صاحب آن ممکن است برای امضاء سایر کلیدها معتمد فرض نشود. برای مثال، کلید N مشروع است زیرا بتوسط E امضاءشده و این کاربر به E اعتماد دارد، ولی N برای امضاء کلیدهای دیگر مورد اعتماد نیست زیرا این کاربر به N اندازهٔ اعتمادی را تخصیص نداده است. بنابراین اگرچه کلید R بتوسط N امضاءشده است، ولی PGP کلید R را مشروع نمی‌داند. این وضعیت کاملاً معقول است. اگر شما می‌خواهید یک پیام خصوصی برای فردی بفرستید، لازم نیست که به آن فرد از همه نظر اعتماد داشته باشید بلکه تنها کافی است مطمئن باشید که کلید عمومی صحیح آن فرد در اختیار شماست.

۴- شکل ۷-۵ همچنین مثالی از یک گرهٔ "یتیم" S با دو امضاء ناشناخته را نشان می‌دهد. چنین کلیدی ممکن است از یک سِرور کلید دریافت شده باشد. PGP نمی‌تواند صرفاً به دلیل اینکه این کلید از یک سِرور معروف دریافت شده است آن را مشروع تلقی کند. کاربر بایستی یا با امضاءکردن آن و یا با اظهار تمایل به اینکه یکی از امضاءکنندگان کلید را کاملاً معتمد می‌داند، مشروعیت کلید را به PGP اعلام دارد.

یک نکتهٔ نهائی: قبلاً خاطرنشان گردید که IDهای کاربران متعددی ممکن است با یک کلید عمومی منفرد و یا با یک دسته‌کلید- عمومی مرتبط باشند. این امر بدین خاطر است که یک فرد ممکن است از نام‌های مختلفی استفاده کرده و یا از طریق امضاء تحت نام‌های مختلف، مثلاً آدرس‌های e-mail متفاوتی را برای خودش معرفی نموده باشد. بنابراین می‌توانیم کلید عمومی را همانند ریشهٔ یک درخت بدانیم. یک کلید عمومی دارای تعدادی IDهای مرتبط با آن است که در زیر هر ID نیز تعدادی امضاء قرار دارد. پیوند یک ID کاربر به یک کلید وابسته به امضاءهای مرتبط با آن ID و کلید است، در حالی که سطح اعتماد به آن کلید (برای استفاده در مورد امضاءکردن کلیدهای دیگر) تابعی از تمام امضاءهای وابسته به آن است.

ابطال کلیدهای عمومی

یک کاربر ممکن است بخواهد کلید عمومی خود را باطل کند. این امر یا به دلیل لورفتن کلید و یا به این دلیل است که کاربر کلید را برای مدتی طولانی استفاده کرده و می‌خواهد آن را تعویض کند. توجه کنید که لازمهٔ لورفتن این است که دشمن به نحوی یک کپی از کلید خصوصی رمزنگاری نشده شما را بدست آورده باشد، یا این که دشمن هم کلید خصوصی را از دسته- کلید خصوصی شما بدست آورده و هم عبارت عبور شما را کشف کرده باشد.

قانون ابطال یک کلید- عمومی این است که صاحب آن بایستی یک گواهی‌نامهٔ ابطال که بتوسط صاحب کلید امضاء شده باشد، را تهیه کند. این گواهی‌نامه همان فرم یک گواهی‌نامهٔ امضاء نرمال را داشته اما شامل نشانگری است که نشان می‌دهد که هدف از این گواهی‌نامه لغو استفاده از یک کلید عمومی است. توجه شود که کلید خصوصی نظیر این کلید عمومی بایستی برای امضاء گواهی‌نامه‌ای که یک کلید عمومی را باطل می‌کند بکار رود. صاحب کلید سپس بایستی این گواهی‌نامه را هرچه سریع‌تر و در سطح هرچه وسیع‌تر انتشار دهد تا افراد مرتبط با او متعاقباً دسته‌کلیدهای- عمومی خود را به روز درآورند.

توجه شود که دشمنی که کلید خصوصی یک کاربر را دزدیده است، نیز می‌تواند چنین گواهی‌نامه‌ای را صادر کند. ولی چون این امر هم دشمن و هم صاحب قانونی کلید را از استفاده از کلید محروم می‌سازد، تهدید ایجادشده بسیار کمتر از استفاده بداندیشانه از یک کلید خصوصی دزدیده شده است.

۵-۲ S/MIME

S/MIME (Secure/Multipurpose Internt Mail Extension) که مبتنی بر تکنولوژی برخاسته از شرکت RSA Data Security است، یک شکوفائی امنیتی در MIME که فرمت استاندارد پست الکترونیک در اینترنت است به وجود می‌آورد. اگرچه هم PGP و هم S/MIME هر دو در خط استانداردهای IETF قرار دارند ولی بنظر می‌رسد که نهایتاً S/MIME بصورت استاندارد صنعت برای مصارف تجاری و سازمانی باقی خواهد ماند در حالی که PGP انتخاب غالب کاربران شخصی پست الکترونیک خواهد بود. S/MIME در تعدادی از اسناد تعریف شده است که مهم‌ترین آنها RFCهای 3369، 3370 و 3850 می‌باشند.

برای فهم S/MIME ابتدا لازم است تا از فرمت زیرساخت پست الکترونیکی که S/MIME از آن استفاده می‌کند، یعنی MIME اطلاعاتی داشته باشیم. اما برای درک اهمیت MIME لازم است تا به عقب برگشته و از فرم استاندارد سنّتی پست الکترونیک یعنی RFC 822 که هنوز دارای کاربرد عام است اطلاع حاصل نمائیم. بنابراین در این بخش ابتدا به معرفی این دو استاندارد قدیمی‌تر پرداخته و سپس S/MIME را مورد بحث قرار خواهیم داد.

RFC 822

RFC 822 فرمتی را برای ارسال پیام‌های متنی از طریق پست الکترونیک تعریف می‌کند. این فرمت، استاندارد ارسال پیام‌های متنی مبتنی بر اینترنت بوده و بصورت گسترده‌ای از آن استفاده می‌شود. در بستر RFC 822، چنین تصور می‌شود که هر پیام دارای یک پاکت و یک محتوا است. پاکت شامل همهٔ آن اطلاعاتی است که برای انتقال و تحویل پیام لازم است. محتوا مطلبی است که باید به گیرنده تحویل شود. استاندارد RFC 822 فقط به محتوا مربوط می‌شود. با وجود این، استاندارد محتوا شامل مجموعه‌ای از میدان‌های سرآیند است که ممکن است بتوسط سیستم پستی برای تولید پاکت بکار رود. هدف استاندارد تسهیل شناخت چنین اطلاعاتی بتوسط برنامه‌هاست.

ساختار کلی یک پیام که با RFC 822 همخوانی داشته باشد، بسیار ساده است. یک پیام شامل چند خط سرآیند (*عنوان*) بوده که به دنبال آن متن نامحدودی (*بدنه*) قرار دارد. سرآیند بتوسط یک خط خالی از بدنه جدا می‌شود. به بیان دیگر، یک پیام یک متن ASCII است و تمام خطوط آن تا اولین خط خالی، سرآیندی است که بتوسط عامل کاربر سیستم پستی مورد استفاده قرار می‌گیرد.

یک خط سرآیند معمولاً شامل یک کلمهٔ کلیدی بوده که پس از آن علامت **:** قرار گرفته و پس ازآن، آرگومان آن کلمهٔ کلیدی نوشته می‌شود. فرمت اجازه می‌دهد که یک خط طولانی به چندین خط کوتاه‌تر شکسته شود. پرکاربردترین کلمات کلیدی *FROM*، *TO*، *SUBJECT* و *DATE* می‌باشند. مثالی از یک پیام در زیر نشان داده شده است:

```
Date: Tue, 16 Jan 1998 10:37:17 (EST)
From: "William Stallings" <ws@shore.net>
Subject: The Syntax in RFC 822
To: Smith@Other-host.com
Cc: Jones@Yet-Another-host.com

Hello. This section begins the actual message body, which is delimited
from the message heading by a blank line.
```

میدان دیگری که معمولاً در سرآیندهای RFC 822 پیدا می‌شود، *Message-ID* است. این میدان شامل یک شناسهٔ یکتا در رابطه با پیام است.

الحاقیه‌های چند منظورهٔ پست الکترونیک (MIME)

MIME توسعه‌ای در چهارچوب RFC 822 ایجاد می‌کند که هدف آن رفع بعضی مشکلات و محدودیت‌های استفاده از SMTP (Simple Mail Transfer Protocol) و یا بعضی پروتکل‌های انتقال پیام دیگر و RFC 822 برای پست الکترونیک می‌باشد. [MURH98] محدودیت‌های زیر برای پروتکل SMTP/822 را ذکر کرده است:

۱- SMTP نمی‌تواند فایل‌های اجرائی یا سایر اشیاء باینری را انتقال دهد. روش‌های مختلفی برای تبدیل فایل‌های باینری به صورت متن وجود دارد که می‌تواند مورد استفادهٔ سیستم‌های پستیSMTP قرار گیرد (مثل روش مرسوم UNIX UUencode/UUdecode). ولی هیچ‌یک از اینها استاندارد نبوده و حتی استاندارد غالب هم نمی‌باشند.

۲- SMTP نمی‌تواند داده‌های متنی شامل کاراکترهای زبان‌های ملّی را انتقال دهد زیرا اینها بتوسط کُدهای ۸- بیتی با مقادیر دهدهی ۱۲۸ به بالا نمایش داده می‌شوند و SMTP محدود به کُد ۷- بیتی ASCII است.

۳- سِرورهای SMTP ممکن است پیام‌های پستی طویل‌تر از اندازهٔ معینی را نپذیرند.
۴- دروازه‌های SMTP که مترجم بین کُد ASCII و کُد EBCDIC هستند از یک مجموعه قوانین نگاشت یکسان پیروی نکرده و مشکلات ترجمه ایجاد می‌کنند.
۵- دروازه‌های SMTP به شبکه‌های پست الکترونیک X.400 نمی‌توانند از پس داده‌های غیرمتنی موجود در پیام‌های X.400 برآیند.
۶- بعضی از پیاده‌سازی‌های SMTP کاملاً به استانداردهای SMTP که در RFC 821 تعریف شده است وفادار نیستند. مشکلات معمول چنین‌اند:
- حذف، اضافه و یا بنظم درآوردن بازگشت به اول خط و خط خالی.
- قطع کردن و یا جمع کردن خطوطی که طویل‌تر از ۷۶ کاراکتر هستند.
- حذف فضای سفید در انتهای پیام (کارکترهای tab و space).
- پر کردن بین خطوط یک پیام بصورتی که همه دارای طول یکسان باشند.
- تبدیل کاراکترهای tab به چندین کاراکتر space.

MIME قصد دارد تا این مشکلات را طوری حل کند که با پیاده‌سازی‌های موجود RFC 822 سازگار باشد. مشخصه‌ها در RFCهای 2045 تا 2049 درج شده‌اند.

مروری بر MIME

مشخصه‌های MIME شامل عناصر زیر است:

۱- پنج میدان جدید برای پیام تعریف شده است که می‌توانند در سرآیند RFC 822 جای گیرند. این میدان‌ها شامل اطلاعاتی در مورد بدنه پیام است.
۲- تعدادی فرمت برای محتوا تعریف شده است که صورت ظاهر e-mailهائی که پست الکترونیک چندرسانه‌ای را پشتیبانی می‌کنند استاندارد می‌نماید.
۳- کُدینگ‌هائی برای انتقال تعریف شده‌اند که تبدیل هر نوع فرمت محتوای پیام به فرمی که در برابر تغییر به توسط سیستم پستی محافظت شده است را فراهم می‌سازد.

در این قسمت پنج میدان سرآیند پیام را معرفی می‌کنیم. سپس به فرمت‌های محتوای پیام و کُدینگ‌های انتقال می‌پردازیم.
پنج میدان سرآیند که در MIME تعریف شده است به قرار زیراند:

- **شماره نسخهٔ MIME**: اندازهٔ پارامتر این میدان بایستی 1.0 باشد. این میدان نشان می‌دهد که پیام از RFC 2045 و RFC 2046 تبعیت می‌نماید.
- **نوع محتوا**: داده‌ای که در بدنهٔ پیام قرار دارد را با جزئیات کافی توصیف می‌کند تا کاربر دریافت‌کننده بتواند عامل و یا مکانیسم مناسبی را برای نمایش این داده بکار گیرد و در غیراینصورت با روش مناسبی با دیتا برخورد نماید.
- **روش کُدینگ انتقال محتوا**: نوع تبدیل بکار گرفته شده برای نمایش بدنه پیام بصورتی که برای انتقال پستی قابل قبول باشد را مشخص می‌کند.
- **کُد شناسائی محتوا**: برای معرفی اقلام MIME در زمینه‌های چندگانه بصورت یکتا استفاده می‌شود.

- **توصیف محتوا**: یک توصیف متنی از شیئی که همراه بدنهٔ پیام است. این وقتی مفید است که این شیء قابل خواندن نباشد (مثل داده‌های صوتی).

یک و یا همهٔ میدان‌ها ممکن است در یک سرآیند نرمال RFC 822 ظاهر شوند. یک پیاده‌سازی مبتنی بر این پروتکل بایستی میدان‌های شماره نسخهٔ MIME، نوع محتوا و روش کُدینگ محتوا را پشتیبانی کرده ولی میدان‌های شناسائی محتوا و توصیف محتوا اختیاری بوده و ممکن است در سیستم گیرنده مورد توجه قرار نگیرند.

انواع محتویات MIME

بخش قالب مشخصه‌های MIME مربوط به تعریف اقلام متنوعی برای محتوای پیام است. این امر منعکس‌کنندهٔ نیاز فراهم‌آوردن روش‌های استاندارد نمایش اطلاعات، در یک محیط چند رسانه‌ای است.

جدول ۳-۵ انواع محتوا در RFC 2046 را نشان می‌دهد. ۷ نوع عمده برای محتوا و جمعاً ۱۵ زیرمجموعهٔ محتوائی در این جدول نشان داده شده است. بطور کلی یک محتوا مبین شکل عمومی دیتا بوده و یک زیرمحتوا، فرمت خاص آن محتوا را تعریف می‌کند.

جدول ۳-۵ انواع محتویات MIME

نوع (Type)	زیرنوع (Subtype)	توصیف
Text	Plain	متن فرمت نشده. ممکن است ASCII و یا ISO 8859 باشد.
	Enriched	انعطاف‌پذیری بیشتری را در فرمت فراهم می‌آورد.
Multipart	Mixed	بخش‌های مختلف مستقل از هم بوده ولی بایستی با هم منتقل شوند. آنها بایستی با همان نظمی که در پیام پستی قرار دارند به گیرنده عرضه گردند
	Parallel	فرق آن با زیرنوع Mixed در این است که نظمی برای تحویل بخش‌های مختلف به گیرنده تعریف نشده است.
	Alternative	بخش‌های مختلف، نسخه‌های متفاوت یک نوع اطلاعات می‌باشند. آنها بر حسب نزدیکی بیشتر با فرم اولیه به نظم درآمده‌اند و سیستم پستی گیرنده بایستی «بهترین» نسخه را به کاربر عرضه نماید.
	Digest	شبیه Mixed است با این تفاوت که پیش‌فرض type/subtype هر بخش message/rfc822 است.
Message	rfc822	بدنهٔ پیام خود یک پیام کپسولی شده بر اساس RFC 822 است.
	Partial	برای قطعه قطعه کردن یک واحد طولانی پستی بکار می‌رود بطوری که برای دریافت‌کننده مرئی نباشد.
	External-body	شامل یک نشانگری است که به یک عنصر در جای دیگر اشاره می‌کند.
Image	jpeg	تصویر دارای فرمت JPEG و کُدینگ JFIF است.
	gif	تصویر دارای فرمت GIF است.
Video	mpeg	فرمت MPEG
Audio	Basic	کُدینگ ۸- بیتی تک کانالهٔ ISDN با استفاده از قانون μ و نرخ 8 khz
Application	PostScript	Adobe Postscript
	Octet-stream	داده‌های باینری عمومی شامل بایت‌های ۸- بیتی

اگر بدنهٔ پیام از **نوع متن (text type)** باشد، بجز پشتیبانی از مجموعهٔ کاراکترهای مشخص شده هیچ نرم‌افزار مخصوص دیگری مورد نیاز نیست. زیر مجموعهٔ متن، یکی متن ساده (*plain*) است، که صرفاً دنباله‌ای از کاراکترهای ASCII و یا کاراکترهای مخصوص ISO 8859 است. زیرمجموعهٔ دیگر متن غنی شده (*enriched*) است که قابلیت بیشتری را در فرمت دیتا می‌پذیرد.

نوع چندبخشی (multipart type) نشان می‌دهد که بدنهٔ پیام شامل بخش‌های متعدد و مستقل است. میدان سرآیند نوع محتوا شامل یک پارامتر بنام مرز (boundry) است که فاصلهٔ بین بخش‌های بدنه را تعریف می‌کند. هر مرز از اول یک خط جدید شروع شده و شامل دو خط فاصله (hyphen) و به دنبال آن یک اندازهٔ مرز است. مرز نهائی که شامل انتهای آخرین بخش است نیز دارای یک پسوند با دو خط فاصله است. درون هر بخش ممکن است یک سرآیند اختیاری معمولی MIME وجود داشته باشد.

در زیر مثال ساده‌ای از یک پیام چندبخشی نشان داده شده است که شامل دو بخش بوده و هر بخش نیز شامل یک متن ساده است (اقتباس از RFC 2046).

```
From:  Nathaniel Borenstein  <nsb@bellcore.com>
To:  Ned Freed  <ned@innosoft.com>
Subject:  Sample message
MIME-Version:  1.0
Content-type:  multipart/mixed;  boundary ="simple boundary"

This is the preamble. It is to be ignored, though it is a handy place for
 mail composers to include an explanatory note to non-MIME conformant
readers.--simple boundary

This is implicitly typed plain ASCII text. It does NOT end with a
linebreak.--simple boundary
Content-type:  text/plain;  charset=us-ascii

This is explicitly typed plain ASCII text. It DOES end with a linebreak.

--simple boundary--
This is the epilogue. It is also to be ignored.
```

نوع چندبخشی خود دارای چهار زیـــرنـوع (subtype) است که تمـام آنهـا دارای یـک انشـای کلی هستند. **multipart/mixed subtype** وقتی مورد استفاده قرار می‌گیرد که چندین بخش مستقل در پیام وجود داشته باشد که بایستی به ترتیب مشخصی به هم گره بخورند. در **multipart/parallel subtype** نظم بخش‌های مختلف دارای اهمیت نمی‌باشند. اگر سیستم دریافت مناسب باشد، بخش‌های مختلف پیام می‌توانند بطور موازی نمایش داده شوند. برای مثال یک بخش تصویری و یا متنی می‌تواند با توضیحات صوتی همراه باشد که در حالی که تصویر و یا متن در حال نمایش است بخش صوتی نیز به همراه آن اجرا گردد.

برای **multipart/alternative subtype** بخش‌های مختلف پیام، نمایش‌های مختلفی از یک نوع اطلاعات هستند. مثال زیر نمونه‌ای از آن است:

```
From: Nathaniel Borenstein <nsb@ bellcore.com>
To: Ned Freed <ned@innosoft.com>
Subject: Formatted text mail
MIME-Version: 1.0
Content Type: multipart/alternative; boundary=boundary42
--boundary42

Content-Type: text/plain; charset=us-ascii

… plain text version of message goes here …

--boundary42
Content-Type: text/enriched

… RFC 1896 text/enriched version of some message goes here…
--boundary42--
```

در این زیرنوع، بخش‌های بدنهٔ پیام بر حسب رجحان بر یکدیگر مرتب می‌شوند. برای مثال بالا اگر سیستم گیرنده قادر به نمایش پیام با فرمت متن غنی شده باشد این عمل انجام می‌شود و در غیر اینصورت متن ساده بکار خواهد رفت.

multipart/digest subtype وقتی بکار می‌رود که هریک از بخش‌های بدنه، بصورت یک پیام RFC 822 با سرآیندهای آن تعبیر شود. این زیرنوع ما را قادر به ساخت پیامی خواهد نمود که بخش‌های مختلف آن پیام‌های انفرادی هستند. بعنوان مثال میاندار یک گروه ممکن است پیام‌های e-mail افراد گروه را جمع‌آوری کرده، این پیام‌ها را بسته‌بندی نموده و آنها را بصورت یک پیام کپسولی شدهٔ MIME بفرستد.

نوع پیام(message type) تـعدادی قـابلیت‌های مهم درMIME را فراهم می‌سازد. **message/rfc822 subtype** نشان می‌دهد که خود بدنهٔ پیام یک پیام کامل، شامل سرآیند و بدنه، است. صرفنظر از نام این زیرنوع، پیام کپسولی شده ممکن است نه تنها یک پیام سادهٔ RFC 822 بلکه هر نوع پیام دیگرMIME باشد.

message/partial subtype قطعه‌قطعه کردن یک پیام طولانی به بخش‌های مختلف را امکان‌پذیر می‌کندکه بایستی در مقصد دوباره بهم بپیوندند. برای این زیرنوع، سه پارامتردر میدان Content-Type:Message/Partial field مشخص شده است: یک *id* که برای تمام قطعات مشترک است، یک *شماره ردیف* که برای هر قطعه متفاوت است و تعداد *کل* قطعات.

message/external-body subtype نشان می‌دهد که دیتای واقعی که بایستی از طریق پیام تحویل گردد، در بدنهٔ پیام نیست. بجای آن بدنه شامل اطلاعات لازم برای دسترسی به داده است. همانند دیگر انواع پیام، message/external-body subtype دارای یک سرآیند خارجی و یک پیام کپسولی شده با سرآیند خود آن است. تنها میدان لازم در سرآیند خارجی، میدان نوع محتوا است که این پیام را بعنوان یک زیرنوع message/external-body معرفی می‌کند. سرآیند داخلی، سرآیند پیام برای پیام کپسولی شده است. میدان نوع محتوا در سرآیند خارجی بایستی شامل یک پارامتر نوع دسترسی باشد که نمایشگر روش دسترسی مثل FTP (file transfer protocol) است.

نوع کاربرد(application type) به سایر انواع دیتا اشاره می‌کند که نوعاً یا دیتای باینری ترجمه نشده و یا اطلاعاتی است که بایستی بتوسط یک برنامهٔ کاربردی مبتنی بر پست الکترونیک پردازش شود.

کُدینگ‌های انتقال MIME

یکی از مؤلفه‌های مهم دیگر در مشخصه‌های MIME، علاوه بر تعیین نوع محتوا، تعریف کُدینگ انتقال برای بدنهٔ پیام است. هدف این امر تحویل قابل اعتماد پست الکترونیک در محدودهٔ وسیعی از محیط‌های گوناگون است.

استاندارد MIME دو روش برای کُدکردن دیتا را تعریف کرده است. میدان کُدینگ انتقال محتوا در واقع می‌تواند برابر آنچه در جدول ۴-۵ لیست شده است، شش مقدار را بپذیرد. اما سه تا از این مقادیر (7bit، 8bit و binary) نمایشگر این واقعیت‌اند که هیچ کُدینگی انجام نشده است و فقط اطلاعاتی در مورد نوع دیتا را فراهم می‌سازند. برای انتقال SMTP استفاده از 7bit امن است. فرم‌های 8bit و binary ممکن است در بسترهای حمل پستی دیگر قابل استفاده باشند. اندازهٔ دیگر کُدینگ انتقال محتوا، مقدار x-token است که نشان می‌دهد روش کُدینگ دیگری بکار گرفته شده و بایستی برای آن نامی ارائه گردد. این روش ممکن است مخصوص یک سازندهٔ خاص و یا کاربرد خاص باشد. دو روشی که در این مورد تعریف شده‌اند یکی quoted-printable و دیگری base64 است. این دو روش برای این تعریف شده‌اند که یک حق انتخاب بین یک تکنیک انتقال که ضرورتاً قابل خواندن بتوسط انسان است و دیگری که برای همهٔ انواع دیتا مطمئن بوده و نسبتاً مختصر است، وجود داشته باشد.

کُدینگ انتقال quoted-printable وقتی مفید است که دیتا عمدتاً شامل اُکتت‌هائی باشد که نظیر کاراکترهای قابل چاپ ASCII اند. در واقع این روش کاراکترهای ناامن را با فرم هگزادسیمال کُد آنها نمایش داده و خطوط خالی قابل برگشت (نَرم) را برای محدودکردن خطوط پیام به ۷۶ کاراکتر معرفی می‌کند.

کُدینگ انتقال base64 که کُدینگ radix-64 نیز خوانده می‌شود، روشی معمول برای کُدکردن هر نوع دیتای باینری به نحوی است که در برابر پردازش برنامه‌های حمل پستی آسیب‌ناپذیر باشد. از این روش در PGP هم استفاده می‌شود و در ضمیمهٔ ۵ ب این فصل توصیف شده است.

یک مثال چندبخشی

شکل ۸-۵ که از RFC 2045 گرفته شده است، طرح یک پیام چندبخشی مرکب را نشان می‌دهد. پیام دارای پنج بخش است که بایستی بطور سریال نمایش داده شوند: دو متن ساده در مقدمهٔ، یک پیام چندبخشی جاسازی شده در داخل آن، یک بخش متن غنی‌شده و یک پیام متنی کپسولی شده که با کاراکترهای غیر ASCII بیان شده است. پیام چندبخشی جاسازی شده دارای دو قسمت است که بایستی نشان داده شوند، یک تصویر و یک قطعهٔ صوتی.

جدول ۴-۵ کُدینگ‌های انتقال MIME

7bit	تمام دیتا با خطوط کوتاهی از کاراکترهای ASCII نشان داده می‌شود.
8bit	خطوط کوتاه‌اند ولی ممکن است شامل کاراکترهائی غیر از ASCII باشند (اُکتت‌هائی با مجموعهٔ بیت‌هائی از درجهٔ بالا)
binary	نه تنها کاراکترهای غیر ASCII می‌توانند وجود داشته باشند بلکه خطوط پیام الزاماً برای انتقال از طریق پروتکل SMTP به اندازهٔ کافی کوتاه نیستند.
quoted-printable	داده‌ها را به ترتیبی کُد می‌کند که اگر داده‌های کُد شده بیشتر شامل متون ASCII باشند، فرم کُدشده تا حد زیادی قابل شناسائی بتوسط انسان خواهد بود.
base64	دیتا را با نگاشتی از بلوک‌های ۶- بیتی ورودی به بلوک‌های ۸- بیتی خروجی طوری کُد می‌کند که همهٔ آنها بتوسط کاراکترهای ASCII قابل چاپ‌اند.
x-token	یک نوع کُدینگ غیراستاندارد است.

MIME-Version: 1.0
From: Nathaniel Borenstein nsb@bellcore.com
To: Ned Freed ned@innosoft.com
Subject: A multipart example
Content-Type: multipart/mixed;
Boundry=unique-boundry-1

This is the preamble area of a multipart message. Mail readers that understand multipart format should ignore this preamble. If you are reading this text you might want to consider changing to a mail reader that understands how to properly display multipart messages.

--unique-boundry-1

…Some text appears here…
[Note that the preceding blank line means no header fields were given and this is text with charset US ASCII.
It could have been done wiyh explicit typing as in the next part.]

--unique-boundry-1
Content-type: text/plain;charset=US-ASCII

This could have been part of the previous part, but illustrates explicit versus implicit typing of body parts.

--unique-boundry-1
Content-Type: multipart/parallel: boundry=unique-boundry-2

--unique-boundry-2
Content-Type: audio/basic
Content-Transfer-Encoding: base64

…base64-encoded 8000 Hz single-channel mu-law-format audio data goes here….

--unique-boundry-2
Content-Type: image/jpeg
Content-Transfer-Encoding: base64

…base64-encoded image data goes here…

--unique-boundry-2--

--unique-boundry-1
Content-type: text/enriched

This is <bold><italic>richtext.</italic></bold><smaller>as defined in RFC 1896</smaller>

Isn`t it<bigger><bigger>cool?</bigger></bigger>

--unique-boundry-1
Content-Type:message/rfc822

From: (mailbox in US-ASCII)
To: (address in US-ASCII)
Subject: (subject in US-ASCII)
Content-Type: Text/plain; charset=ISO-8859-1
Content-Transfer-Encoding: Quoted-printable

…Additional text in ISO-8859-1 goes here…

--unique-boundry-1--

شکل ۸-۵ مثالی از ساختار پیام MIME

جدول ۵-۵ فرم‌های بومی و قانونی

فرم بومی (Native Form)	بدنه پیامی که باید ارسال شود با فرمت بومی سیستم ارسال‌کننده خلق می‌گردد. از مجموعهٔ کاراکترهای بومی استفاده شده و در جای مناسب از قوانین محلی پایان خط استفاده می‌شود. بدنه ممکن است یک فایل متنی مبتنی بر UNIX، یک تصویر مبتنی بر Sun، یک فایل با اندیس VMS، یک دیتای صوتی مبتنی بر سیستم که در حافظه ذخیره شده و یا هرچیز دیگری در رابطه با مدل محلی برای نمایش نوعی اطلاعات باشد. اصولاً دیتا به فرم «بومی» که مرتبط با نوع تعیین شده بتوسط نوع رسانه است خلق می‌گردد.
فرم قانونی (Canonical Form)	تمام بدنه پیام، شامل اطلاعات «خارج از باند» مثل طول رکوردها و احتمالاً اطلاعات مربوط به صفات فایل‌ها، به فرم قانونی تبدیل می‌شود. نوع رسانهٔ مخصوص بدنه و مشخصات مربوطه، فرم قانونی بکارگرفته شده را تعیین می‌کنند. تبدیل به فرم قانونی مناسب ممکن است شامل تبدیل مجموعهٔ کاراکترها، تبدیل داده‌های صوتی، فشرده‌سازی و یا سایر عملیات مختص به رسانه‌های مختلف باشند. اگر تبدیل مجموعهٔ کاراکترها مورد نظر باشد بایستی دقت کرد که دیکتهٔ لغات مورد توجه قرار گیرد زیرا ممکن است در تبدیل مجموعه‌ها به یکدیگر تناقضاتی در فرم نمایش آنها بوجود آید.

فرم قانونی

یکی از مفاهیم مهم MIME و S/MIME فرم قانونی(canonical) است. فرم قانونی یک فرمت است، که در تناسب با نوع محتوا، برای استفاده بین سیستم‌ها استاندارد شده است. این در تضاد با یک فرمت بومی است که ممکن است برای یک سیستم خاص دیگر عجیب جلوه نماید. جدول ۵-۵ که از RFC 2049 اقتباس شده است بایستی به درک مطلب کمک کند.

عملکرد S/MIME

از نظر عملکرد کلّی، S/MIME خیلی شبیه PGP است. هردوی آنها قابلیت امضاء و/ یا رمزنگاری پیام‌ها را فراهم می‌آورند. در این قسمت بطور مختصر توانمندی S/MIME را بیان می‌کنیم. سپس با بررسی فرمت‌های پیام و آماده‌سازی پیام به جزئیات این توانمندی می‌پردازیم.

عملیات

S/MIME عملیات زیر را ممکن می‌سازد:

- **Enveloped data**: این شامل محتوای رمزنگاری شده از هر نوع، و کلیدهای رمز محتوای رمزنگاری شده برای یک یا چند گیرنده است.
- **Signed data**: یک امضاء دیجیتال با محاسبهٔ چکیدهٔ پیام از محتوائی که باید امضاء شود و سپس رمزنگاری آن با کلید خصوصی امضاءکننده ایجاد می‌گردد. سپس محتوا باضافهٔ امضاء دیجیتال آن با استفاده از کُدینگ base64 کُد می‌شود. یک پیام signed data تنها بتوسط گیرنده‌ای قابل رؤیت است که قابلیت S/MIME را داشته باشد.

- **Clear-signed data**: همانند signed data، یک امضاء دیجیتال از محتوا تولید می‌شود ولی در این مورد فقط امضاء دیجیتال با استفاده از کُدینگ base64 کُد می‌شود. در نتیجه گیرنده‌هائی که به S/MIME مجهز نیستند نیز می‌توانند محتوای پیام را مشاهده نمایند ولی نمی‌توانند امضاء را تصدیق کنند.
- **Signed and enveloped data**: واحدهای signed-only و encrypted-only می‌توانند تودرتو باشند بطوری که دیتای رمزنگاری شده بتواند امضاء شده و دیتای امضاءشده بتواند رمزنگاری شود.

الگوریتم‌های رمزنگاری

جدول ۶-۵ الگوریتم‌های رمزنگاری بکار رفته در S/MIME را خلاصه کرده است. S/MIME از واژه‌های زیر که از RFC 2119 گرفته شده است استفاده کرده تا سطح نیاز را مشخص نماید:

- **بایستی (MUST)**: یک نیاز قطعی مشخصه است. یک اجراء باید شامل این ویژگی یا این تابع باشد تا با استاندارد تطبیق کند.
- **شایسته است (SHOULD)**: ممکن است در شرایط خاصی دلایل متقنی برای ملحوظ نداشتن این ویژگی یا این تابع وجود داشته باشد، ولی توصیه می‌شود که یک اجراء شامل این ویژگی یا تابع باشد.

S/MIME سه الگوریتم کلید- عمومی را بکار می‌گیرد. استاندارد امضاء دیجیتال (DSS) که در فصل ۳ از آن یاد شد، الگوریتم انتخاب شده برای امضاء دیجیتال است. S/MIME از Diffie-Hellman بعنوان الگوریتم منتخب برای رمزنگاری کلیدهای اجلاس استفاده می‌کند. در حقیقت، S/MIME از یک نوع تغییر یافتهٔ Diffie-Hellman بنام EIGamal که رمزنگاری/ رمزگشائی را فراهم می‌آورد استفاده می‌کند. در انتخاب دیگر، RSA که از آن نیز در فصل ۳ یاد گردید می‌تواند هم برای امضاءها و هم برای رمزنگاری کلید اجلاس بکار رود. این‌ها همان الگوریتم‌هائی هستند که در PGP بکار می‌روند و سطح بالائی از امنیت را فراهم می‌سازند. برای تابع hash که برای خلق امضاء دیجیتال بکار گرفته می‌شود، مشخصه، تابع ۱۶۰- بیتی SHA-1 را تعیین نموده است ولی توصیه می‌کند که گیرنده تابع ۱۲۸- بیتی MD5 را نیز به منظور سازگاری با نسخه‌های قدیمی‌تر S/MIME پشتیبانی نماید. همانطور که در فصل ۳ خاطرنشان گردید، نگرانی‌های قابل بحثی در مورد امنیت MD5 وجود دارد و بنابراین SHA-1 قطعاً انتخاب بهتری است.

برای رمزنگاری پیام، DES سه‌گانه (3DES) سه کلیدی توصیه شده است ولی اجراهای منطبق بایستی RC2 ۴۰- بیتی را پشتیبانی نمایند. مورد اخیر یک الگوریتم رمزنگاری ضعیف ولی منطبق با قوانین کنترل صادرات آمریکا است.

مشخصه‌های S/MIME شامل بحثی در مورد نحوهٔ تصمیم‌گیری نسبت به انتخاب الگوریتم رمزنگاری محتوای پیام است. در واقع یک عامل ارسال‌کنندهٔ پیام بایستی نسبت به دو مورد تصمیم‌گیری نماید. اول اینکه آیا عامل دریافت‌کننده قادر به رمزگشائی یک الگوریتم رمزنگاری هست یا نه. دوم اینکه اگر عامل دریافت‌کننده تنها قادر به پذیرش محتویات رمزنگاری ضعیف است، آیا این امر برای عامل ارسال قابل پذیرش است یا خیر. برای حمایت از این روند تصمیم‌گیری، یک عامل ارسال‌کننده می‌تواند قابلیت‌های رمزگشائی خود را بر حسب یک لیست ترجیحی برای هر پیامی که ارسال می‌شود اعلام دارد. عامل دریافت‌کننده ممکن است این اطلاعات را برای استفاده‌های آتی ذخیره کند.

قواعد زیر، برحسب ترتیب، بایستی بتوسط یک عامل ارسال رعایت شوند:

۱- اگر عامل فرستنده دارای لیستی از قابلیت‌های رمزگشائی گیرندهٔ مورد نظر بصورت ترجیحی است، او شایسته است که نخستین مورد (بالاترین اولویت) لیست،که قادر به استفاده از آن است، را انتخاب کند.

جدول ۶-۵ الگوریتم‌های رمزنگاری استفاده شده در S/MIME

عمل	نیازها
یک چکیدهٔ پیام خلق می‌گردد تا بعداً در تولید یک امضاء دیجیتال از آن استفاده شود.	**بایستی** SHA-1 را پشتیبانی کند. عامل‌های گیرنده **شایسته است** MD5 را برای سازگاری با نسخه‌های قدیمی‌تر پشتیبانی کنند.
چکیدهٔ پیام رمزنگاری می‌شود تا امضاء دیجیتال تولید شود.	عامل‌های فرستنده و گیرنده **بایستی** DSS را پشتیبانی کنند.. عامل‌های فرستنده **شایسته است** رمزنگاری RSA را پشتیبانی کنند. عامل‌های گیرنده **شایسته است** تأئید امضاءهای RSA با کلیدهائی از طول ۵۱۲ تا ۱,۰۲۴ بیت را پشتیبانی کنند.
کلید اجلاس رمزنگاری می‌گردد تا به همراه پیام ارسال شود.	عامل‌های فرستنده و گیرنده **بایستی** رمزنگاری RSA با طول کلیدهائی از ۵۱۲ تا ۱,۰۲۴ بیت را پشتیبانی کنند. عامل‌های فرستنده و گیرنده **شایسته است** Diffie-Hellman را پشتیبانی کنند.
پیام بتوسط کلید اجلاس یکبار- مصرف رمزنگاری می‌شود.	عامل‌های فرستنده و گیرنده **بایستی** رمزنگاری با 3DES را پشتیبانی کنند. عامل‌های فرستنده **شایسته است** رمزنگاری با AES را پشتیبانی کنند. عامل‌های فرستنده **شایسته است** رمزنگاری با RC2/40 را پشتیبانی کنند.
یک کُد اعتبارسنجی پیام خلق می‌شود.	عامل‌های فرستنده **بایستی** HMAC با SHA-1 را پشتیبانی کنند. عامل‌های گیرنده **شایسته است** HMAC با SHA-1 را پشتیبانی کنند.

۲- اگر عامل فرستنده چنین لیستی از یک گیرندهٔ مورد نظر را در اختیار ندارد ولی قبلاً یکی دو پیام از گیرنده دریافت کرده است، آنگاه شایسته است که در پیام خروجی از همان الگوریتم رمزنگاری استفاده کندکه در آخرین پیام امضاء و رمزنگاری شده از همان گیرنده، دریافت کرده است.

۳- اگر عامل فرستنده هیچ اطلاعاتی در مورد قابلیت‌های رمزگشائی گیرندهٔ مورد نظر نداشته ولی آمادگی این ریسک را دارد که حتی به قیمت غیرقابل رمزگشائی شدن پیام، پیام را ارسال کند شایسته است که از 3DES استفاده نماید.

۴- اگر عامل فرستنده هیچ اطلاعاتی در مورد قابلیت‌های رمزگشائی گیرندهٔ مورد نظر نداشته و نمی‌خواهد این ریسک را پذیرا شود که گیرنده نتواند پیام او را بخواند، بایستی از RC2/40 استفاده کند.

اگر قرار باشد که یک پیام به گیرنده‌های متعددی ارسال گردد و یک الگوریتم رمزنگاری مشترک نتواند برای همهٔ آنها انتخاب شود، آنگاه عامل فرستنده نیاز به ارسال دو پیام دارد. در چنین صورتی به این مهم بایستی توجه گردد که امنیت پیام بتوسط انتقال کپی با امنیت پائین‌تر آسیب‌پذیر خواهد شد.

جدول ۷-۵ انواع محتوای S/MIME

نوع	زیرنوع	پارامتر S/MIME	توصیف
Multipart	Signed		یک پیام امضاء شدهٔ صریح در دو بخش: یک بخش پیام و یک بخش امضاء.
Application	pkcs7-mime	signedData	یک موجودیت امضاء شدهٔ S/MIME.
	pkcs7-mime	envelopedData	یک موجودیت رمزنگاری شدهٔ S/MIME.
	pkcs7-mime	degenerate signedData	یک موجودیت که فقط شامل یک گواهی‌نامهٔ کلید- عمومی است.
	pkcs7-mime	compressedData	یک موجودیت فشرده‌سازی شدهٔ S/MIME.
	pkcs7-signature	signedData	نوع محتوای امضاء زیرنوع یک پیام multipart/signed است.

پیام‌های S/MIME

S/MIME از تعدادی محتوای جدید MIME استفاده می‌کند که در جدول ۷-۵ نشان داده شده است. تمام کاربردهای جدید از PKCS استفاده می‌کنند. PKCS به مجموعه‌ای از مشخصه‌های رمزنگاری کلید- عمومی اشاره می‌کند که بتوسط لابراتوارهای RSA نشر شده و در اختیار پروژه S/MIME گذاشته شده است.

در اینجا ابتدا نگاهی به رَوند عمومی آماده‌سازی پیام S/MIME انداخته و سپس محتویات جدید را بررسی می‌کنیم.

ایمن‌سازی یک واحد MIME

S/MIME یک واحد MIME را با امضاء، رمزنگاری و یا هردوی آنها ایمن می‌سازد. یک واحد MIME ممکن است تمام یک پیام (بجز سرآیندهای RFC 822) بوده و یا اگر نوع محتوا از نوع چندبخشی باشد، یک واحد MIME یک یا چند زیربخش از پیام است. یک واحد MIME بر اساس قواعد نرمال آماده‌سازی پیام MIME تهیه می‌شود. سپس واحد MIME بعلاوهٔ بعضی داده‌های مرتبط با امنیت، مثل شناسه‌های الگوریتم‌ها و گواهی‌نامه‌ها، بتوسط S/MIME مورد پردازش قرار گرفته تا آنچه بنام عنصر PKCS است تهیه شود. سپس یک عنصر PKCS بعنوان محتوای پیام در نظر گرفته شده و در MIME لفافه‌بندی می‌شود (بتوسط سرآیندهای مناسب MIME). رَوند عملیات وقتی به عناصر مشخص پرداخته و مثال‌هائی را عرضه کنیم، روشن خواهد شد.

در تمام موارد، پیامی که قرار است ارسال شود به فرم قانونی تبدیل می‌شود. علی‌الخصوص برای یک نوع و زیرنوع داده شده، فرم قانونی مناسب برای پیام انتخاب می‌گردد. برای یک پیام چندبخشی، فرم قانونی مناسب برای هر زیربخش رعایت می‌گردد.

استفاده از کُدینگ انتقال نیاز به توجه ویژه دارد. در بیشتر موارد، نتیجهٔ اعمال الگوریتم‌های امنیتی، تهیهٔ یک عنصر است که بخشی و یا همهٔ آن بصورت دیتای باینری نمایش داده شده است. این عنصر سپس در یک پیام MIME بیرونی لفافه‌بندی شده و سپس کُدینگ انتقال که معمولاً base64 است به آن اعمال می‌گردد. اما در مورد یک پیام چندبخشی امضاءشده که جزئیات آن به زودی توصیف خواهد شد، محتوای پیام در یکی از زیربخش‌ها بتوسط پروسهٔ امنیتی دست نخورده باقی خواهد ماند. بغیر از وقتی که محتوا base64 است، کُدینگ انتقال بایستی از base64 و یا quoted-printable استفاده کند تا خطر تغییر محتوا که امضاء به آن اعمال شده است وجود نداشته باشد.

حال به هریک از انواع محتوای S/MIME نگاهی می‌اندازیم.

EnvelopedData

یک زیرنوع application/pkcs7- mime برای پردازش یکی از چهار دسته S/MIME مورد استفاده قرار می‌گیرد که هریک آنها دارای یک پارامتر smime-type یکتاست. در تمام موارد، عنصر نتیجه شده که یک *object* خوانده می‌شود بصورت فرمی که بنام Basic Encoding Rules (BER) خوانده شده و در توصیه نامهٔ ITU-T X.209 تعریف شده است درمی‌آید. فرمت BER شامل دنباله‌ای از اُکتت‌هاست و بنابراین دیتای باینری است. چنین عنصری بایستی از طریق base64 در پیام بیرونی MIME کُدبندی شود. ابتدا به envelopedData نگاه می‌کنیم.

مراحل آماده‌سازی یک واحد envelopedData در MIME چنین است:

۱- یک کلید اجلاس شبه تصادفی برای یک الگوریتم رمزنگاری متقارن (RC2/40 یا 3DES) تولید شود.

۲- برای هر گیرنده، کلید اجلاس با کلید عمومی RSA گیرنده رمزنگاری شود.

۳- برای هر گیرنده بلوکی با نام RecipientInfo که شامل شناسهٔ گواهی‌نامهٔ کلید- عمومی گیرنده (این گواهی‌نامهٔ X.509 است که بعداً آن را در همین بخش تعریف خواهیم کرد)، یک شناسه برای الگوریتم رمزنگاری استفاده شده برای رمزنگاری کلید اجلاس و خود کلید اجلاس است تهیه شود.

۴- محتوای پیام با کلید اجلاس رمزنگاری شود.

بلوک‌های RecipientInfo که به دنبال آن محتوای رمزنگاری شده قرار داده شده است، envelopedData را تشکیل می‌دهند. این اطلاعات سپس بتوسط base64 کُد می‌شود. نمونه‌ای از این پیام چنین است (سرآیندهای RFC 822 نشان داده نشده‌اند):

```
Content-Type: application/pkcs7-mime; smime-type=enveloped-data;
        Name=smime.p7m
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename=smime.p7m

Rfvbnj756tbBghyHhHUujhJhjH77n8HHGT9HG4VqpfyF467GhIGfHfYT6
7n8HHGghyHhHUujhJh4VqpfyF467GhIGfHfYGTrfvbnjT6jH7756tbB9H
F8HHGTrfvhjhjH776tbB9HG4VQbnj567GhIGfHfYT6ghyHhHUujpfyF4
0GhIGfHfQbnj756YT64V
```

برای بازیابی پیام رمزنگاری شده، گیرنده ابتدا کُد base64 را باز می‌کند. سپس کلید خصوصی گیرنده برای استخراج کلید اجلاس بکار می‌رود. بالاخره محتوای پیام با استفاده از کلید اجلاس رمزگشائی می‌گردد.

SignedData

signedData smime-type در واقع می‌تواند بتوسط یک و یا چند امضاءکننده بکار رود. به منظور سهولت، توصیف خود را به مورد یک امضاء دیجیتال منفرد محدود می‌کنیم. مراحل آماده‌سازی یک واحد signedData در MIME چنین است:

۱- یک الگوریتم برای چکیدهٔ پیام انتخاب شود (SHA یا MD5).
۲- اندازهٔ چکیدهٔ پیام و یا تابع hash محتوا که باید امضاء شود تهیه گردد.
۳- چکیدهٔ پیام با کلید خصوصی امضاءکننده، رمزنگاری شود.
۴- یک بلوک بعنوان SignerInfo که شامل گواهی‌نامهٔ کلید- عمومی امضاءکننده، شناسه‌ای برای الگوریتم چکیدهٔ پیام، شناسه‌ای برای الگوریتم استفاده شده برای رمزنگاری چکیدهٔ پیام و نهایتاً چکیدهٔ رمزنگاری شدهٔ پیام است، تهیه گردد.

واحد signedData شامل یک سری بلوک‌هائی است که شامل شناسهٔ الگوریتم چکیدهٔ پیام، پیامی که بایستی امضاء شود و SignerInfo است. واحد signedData همچنین می‌تواند شامل یک سری گواهی‌نامهٔ کلید- عمومی باشد که بتواند سلسله مراتب مسئول گواهی‌نامه (CA) مرتبط با امضاءکننده را نشان دهد. این اطلاعات سپس با کُد base64 کُدبندی می‌شود. یک نمونه پیام (بجز سرآیندهای RFC 822) چنین است:

```
Content-Type: application/pkcs7-mime; smime-type=signed-data;
        Name=smime.p7m
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename=smime.p7m

567GhIGfHfYT6ghyHhHUujpfyF4f8HHGTrfvhJhjH776tbB9HG4VQbnj7
77n8HHGT9HG4VqpfyF467GhIGfHfYT6rfvbnj756tbBghyHhHUujhJhjH
HuujhJh4VqpfyF467GhIGfHfYGTrfvbnjT6jH7756tbB9H7n8HHGghyHh
6YT64V0GhIGfHfQbnj75
```

برای بازیابی پیام امضاءشده و تأئید امضاء، گیرنده ابتدا کُدینگ base64 را باز می‌کند. آنگاه کلید عمومی امضاءکننده برای رمزگشائی چکیدهٔ پیام مورد استفاده قرار می‌گیرد. گیرنده بطور مستقل چکیدهٔ پیام را محاسبه کرده و به منظور تأئید امضاء آن را با چکیدهٔ رمزگشائی شدهٔ پیام مقایسه می‌کند.

Clear Signing

Clear signing با استفاده از نوع محتوای چندبخشی با یک زیرنوع امضاءشده بدست می‌آید. همانطور که ذکر شد، عمل امضاء شامل رمزکردن پیامی که باید امضاء شود نیست و بنابراین پیام بصورت "clear" ارسال می‌شود. بنابراین گیرنده‌هائی که قابلیت MIME را داشته ولی فاقد قابلیت‌های S/MIME هستند قادر به خواندن پیام ورودی خواهند بود.

یک پیام multipart/signed دارای دو قسمت است. قسمت اول می‌تواند هریک از انواع MIME بوده باشد ولی بایستی طوری تنظیم شود که در خلال انتقال بین فرستنده و گیرنده تغییر نکند. این بدین معنی است که اگر قسمت اول بصورت 7bit نیست، لازم است که با استفاده از base64 و quoted-printable کُدبندی شود. آنگاه این قسمت به همان صورت signedData پردازش می‌شود، اما در این مورد عنصری با فرمت signedData خلق می‌شود که محتوای پیام آن خالی است. این عنصر یک امضاء جدا از پیام است. سپس کُدینگ انتقال با استفاده از base64 روی آن اعمال شده تا قسمت دوم پیام multipart/signed را درست کند. قسمت دوم دارای نوع MIME از نوع application و زیرنوع pkcs7-signature است. نمونه‌ای از این پیام چنین است:

```
Content- Type:  multipart/signed;
     Protocol="application/pkcs7-signature";
     Micalg=shal; boundary=boundary42

--boundary42
Content-Type:  text/plain

This is a clear-signed message.

--boundary42
Content-Type: application/pkcs-signature; name=smime.p7s
Content-Transfer-Encoding:  base64
Content-Disposition:  attachment; filename=smime.p7s

ghyHhHUujhJhjH77n8HHGTrfvbnj756tbB0HG4VqpfyF467GhIGfhfYT6
4VqpfyF467GhIGfHfYT6jH77n8HHGghyHhHUujhJh756tbB9HGTrfvbnj
N8HHGTrfvhJhjH776tbB9HG4VQbnj7567GhIGfHfYT6ghyHhHUujpfyF4
7GhIGfHfYT64VQbnj756
--boundary42--
```

پارامتر پروتکل نشان می‌دهد که این یک واحد two-part clear-signed است. پارامتر micalg نشان‌دهندهٔ نوع چکیدهٔ پیام است. گیرنده می‌تواند با چکیده گرفتن از قسمت اول و مقایسهٔ آن با چکیدهٔ استخراج شده از امضاء در قسمت دوم، امضاء را تأئید نماید.

تقاضای ثبت‌نام

نوعاً یک کاربرد و یا یک کاربر برای تهیهٔ یک گواهی‌نامهٔ کلید- عمومی به یک مسئول صدور گواهی‌نامهٔ (CA) متوسل می‌شود. application/pkcs10 واحد S/MIME برای انتقال درخواست گواهی‌نامه بکار می‌رود. درخواست گواهی‌نامه شامل بلوک certificationRequestInfo بعلاوهٔ یک شناسه الگوریتم رمزنگاری کلید- عمومی بعلاوهٔ امضاء بلوک certificationRequestInfo است که با استفاده از کلید خصوصی فرستنده امضاء شده است. بلوک certificationRequestInfo شامل یک نام (نام واحدی که کلید عمومی او بایستی تأئید گردد) و دنباله‌ای از بیت‌هاست که نمایشگر کلید عمومی کاربر است.

پیام Certificate-Only

یک پیام که فقط شامل گواهی‌نامه‌ها و یا لیست ابطال گواهی‌نامه‌ها (CRL) است می‌تواند در پاسخ به یک تقاضای ثبت‌نام ارسال گردد. پیام یک application/pkcs7-mime type/subtype با یک پارامتر smime-type ابطال است. مراحل اینجا همانند مراحل خلق یک پیام signedData بوده بجز اینکه در اینجا محتوای پیام وجود نداشته و میدان signerInfo خالی است.

پردازش گواهی‌نامه‌های S/MIME

S/MIME از گواهی‌نامه‌های کلید- عمومی که منطبق با نسخهٔ سوم X.509 هستند (به فصل ۴ مراجعه شود) استفاده می‌کند. روش مدیریت-کلید که S/MIME از آن استفاده می‌کند تا حدودی مخلوطی از روش سلسله مراتبی X.509 و وِب‌های معتمد PGP است. همانند مدل PGP، مدیران و یا کاربران S/MIME بایستی هر کلاینت را با لیستی از کلیدهای مورد اعتماد و زمان انقضای کلیدها پیکربندی نمایند. یعنی مسئولیت نگهداری گواهی‌نامه‌های لازم برای تأئید امضاءهای ورودی و رمزنگاری پیام‌های خروجی یک مسئولیت محلّی است. علاوه برآن گواهی‌نامه‌ها بتوسط مسئولین صدور گواهی، امضاء می‌شوند.

نقش عامل کاربر

یک کاربر S/MIME بایستی چندین عمل مدیریتی در زمینهٔ مدیریت-کلید انجام دهد:

- **تولید کلید**: کاربر یک برنامهٔ مدیریتی مرتبط (مثل کسی که مدیریت یک شبکهٔ LAN را داراست)، بایستی قادر به تولید جفت کلیدهای Diffie-Hellman و DSS بوده و شایسته است که بتواند جفت کلیدهای RSA را نیز خلق کند. هر جفت کلید بایستی از یک منبع خوب با ورودی تصادفی غیریقینی اخذ شده و به طریق امنی ذخیره گردد. یک عامل کاربر شایسته است جفت کلیدهای RSA را با طولی بین ۷۶۸ تا ۱,۰۲۴ بیت خلق کرده و نبایستی کلیدی با طول کمتر از ۵۱۲ بیت خلق کند.
- **ثبت نام**: کلید عمومی یک کاربر بایستی به منظور اخذ یک گواهی‌نامهٔ کلید- عمومی X.509 در نزد یک مسئول صدور گواهی‌نامه (CA) به ثبت برسد.
- **ذخیره‌سازی و بازیابی گواهی‌نامه‌ها**: یک کاربر، نیازمند دسترسی به یک لیست محلی از گواهی‌نامه‌هاست تا بتواند امضاءهای ورودی را تأئید کرده و پیام‌های خروجی را رمزنگاری نماید. چنین لیستی بایستی یا بتوسط کاربر، و یا بتوسط یک واحد مدیریت محلّی به نیابت تعدادی از کاربران، نگهداری گردد.

گواهی‌نامه‌های VeriSign

سازمان‌های مختلفی وجود دارند که مسئولیت صدور گواهی‌نامه‌های دیجیتال (CA) را تقبل می‌کنند. بعنوان مثال، Nortel یک بنگاه تجاری CA را فراهم نموده و می‌تواند حمایت از S/MIME در درون یک سازمان را عهـده‌دار گردد. CAهائی که مبتنی بر اینترنت هستند نیز وجود داشته که VeriSign، GTE و U.S.Portal Service از آن جمله‌اند. در بین اینها سرویس VeriSign CA بیشترین کاربرد را داشته که توصیف مختصری از آن را در اینجا می‌آوریم.

VeriSign یک سیستم CA را فراهم آورده است که هدف آن سازگاری با S/MIME و تعداد متنوع دیگری از کاربردهاست. VeriSign گواهی‌نامه‌های X.509 را با نام تجاری VeriSign Digital ID صادر می‌کند. در اوایل سال ۱۹۹۸ میلادی بیش از ۳۵,۰۰۰ وِب سایت تجاری از VeriSign Digital ID استفاده کرده و بیش از یک میلیون Digital ID برای کاربران مرورگرهای Netscape و Microsoft صادر شده بود.

اطلاعاتی که در یک Digital ID قرار دارد وابسته به نوع Digital ID و موارد استفادهٔ آن دارد. یک Digital ID حداقل شامل اقلام زیر است:

- کلید عمومی صاحب این Digital ID
- نام صاحب Digital ID و یا نام مستعار او
- تاریخ انقضاء Digital ID
- شمارهٔ سریال Digital ID
- نام مسئول صدور گواهی که این Digital ID را صادر کرده است.
- امضاء دیجیتال مسئول صدور گواهی‌نامه که Digital ID را صادر کرده است.

Digital IDها همچنین می‌توانند شامل اطلاعات دیگری باشند که کاربر آنها را عرضه کرده است مثل:

- آدرس
- آدرس e-mail
- اطلاعات عمومی ثبت ID (مثل کشور، کُد محلّی، سن و جنسیت)

VeriSign برابر جدول ۸-۵ سه سطح و یا کلاس امنیتی برای گواهی‌نامه‌های کلید- عمومی فراهم می‌آورد. یک کاربر می‌تواند بصورت برخط از سایت VeriSign و یا سایت‌های مرتبط با آن یک گواهی‌نامه درخواست کند. گواهی‌های Class 1 و Class 2 بصورت برخط پردازش شده و معمولاً ظرف چندثانیه به تأئید می‌رسند. بطور خلاصه رَویه‌های زیر بکار گرفته می‌شود:

- برای Class 1 Digital ID، VeriSign آدرس e-mail کاربر را با ارسال یک PIN و یک فرم برداشت اطلاعات Digital ID به آدرس e-mail او که در درخواست وجود دارد، تأئید می‌کند.
- برای Class 2 Digital ID، علاوه بر انجام عملیات مرتبط با Class 1، یک مقایسهٔ اتوماتیک بین اطلاعات ارائه‌شده در فرم درخواست، با پایگاه داده مشتریان نیز بعمل می‌آورد. در نهایت، تأئیدیه به آدرس پستی مشخص شده ارسال گردیده و به کاربر اطلاع داده می‌شود که یک Digital ID بنام او صادر شده است.
- برای Class 3 Digital ID، VeriSign نیاز به اطمینان سطح بالاتری از هویت درخواست‌کننده دارد. یک فرد متقاضی بایستی هویت خود را از طریق ارائهٔ مدارک ثبت شده‌ای در جای دیگر و یا با مراجعهٔ حضوری به اثبات برساند.

سرویس‌های امنیتی افزوده

تا زمان کتابت این کتاب، سه سرویس امنیتی افزوده در پیش‌نویس‌های اینترنت پیشنهاد شده‌اند. جزئیات این سرویس‌ها ممکن است تغییر کرده و سرویس‌های دیگری نیز به آنها اضافه شوند. این سه سرویس بقرار زیراند:

جدول ۸-۵ انواع گواهی‌نامه‌های کلید- عمومی VeriSign

	نحوهٔ احراز هویت	نحوهٔحفاظت از کلید- خصوصی IA	محافظت از کلید خصوصی متقاضی گواهی و مشترک	کاربردهای قابل انجام و موردنظر کاربر
Class1	جستجوی بدون ابهام و خودکار نام و آدرس e-mail	PCA : سخت‌افزار قابل اعتماد. CA : نرم‌افزار قابل اعتماد یا سخت افزار قابل اعتماد.	نرم‌افزار رمزنگاری (محافظت شده با PIN) توصیه میشود ولی لازم نیست.	مرور صفحات وِب و برخی استفاده‌ها از e-mail
Class2	همانند Class1 باضافهٔ کنترل اتوماتیک اطلاعات عضویت و کنترل اتوماتیک آدرس	PCA و CA : سخت افزار قابل اعتماد.	نرم‌افزار رمزنگاری (محافظت شده با PIN) لازم است.	e-mail شخصی و متعلق به سازمان، اشتراک برخط، تعویض کلمهٔ عبور و تأئید نرم‌افزار
Class3	همانند Class1 باضافهٔ حضور فردی با مدارک معتبر احراز هویت بعلاوهٔ کنترل خودکار Class2 ID برای افراد، و سوابق اداری برای سازمان‌ها	PCA و CA : سخت افزار قابل اعتماد.	نرم‌افزار رمزنگاری (محافظت شده با PIN) لازم است. ژتون سخت‌افزاری تـوصیه می‌شود ولی لازم نیست.	بانکداری الکترونیک، دست‌یابی به پایگاه داده، عملیات بانکی شخصی، سرویس‌های برخط ، پذیرش عضویت، سِرور تجارت الکترونیک، تأئید نرم افزار، اعتبارسنجی LRAAها و رمزنگاری مستحکم برای سِرورهای خاص

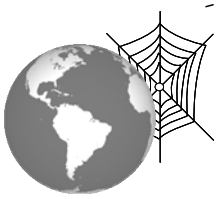
IA = Issuing Authority
CA = Certification Authority
PCA = VeriSign Public Primary Certification Authority
PIN = Personal Identification Number
LRAA = Local Registeration Authority Administrator

- **رسیدهای امضاءشده**: یک رسید امضاءشده ممکن است در یک عنصر VeriSign مورد درخواست قرار گیرد. برگرداندن یک رسید امضاءشده برای فرستندهٔ پیام، تحویل پیام را به اثبات رسانده و به ارسال‌کننده اجازه می‌دهد تا به شخص ثالثی اثبات کند که گیرنده، پیام را دریافت کرده است. در واقع گیرنده تمام پیام اولیه بعلاوهٔ امضاء اولیه (امضاءفرستنده) را امضاء کرده و امضاء جدید را به پیام وصل می‌نماید تا یک پیام S/MIME جدید تولید شود.
- **برچسب‌های امنیتی**: یک برچسب امنیتی ممکن است بهمراه مشخصات اعتبارسنجی شدهٔ یک عنصر SignedData ارسال گردد. یک برچسب امنیتی مجموعه‌ای از اطلاعات امنیتی مربوط به حساسیت محتوا است که بتوسط کپسولی کردن S/MIME فراهم آمده است. برچسب‌ها ممکن است برای کنترل دست‌یابی بکار رفته، و نشان دهند که چه کاربرانی می‌توانند به یک عنصر دست یابند. مورد استفادهٔ دیگر آنها تعیین اولویت‌ها (سرّی، محرمانه، محدود و غیره) و یا تعیین نقش فرد می‌باشند که بیانگر نوع آدم‌هائی است که می‌توانند اطلاعات را رؤیت کنند (مثل تیم پزشکی یک بیمار، بخش تعرفه‌های پزشکی و غیره).

- **لیست‌های پستی امن**: وقتی یک کاربر پیامی را برای گیرندگان متعددی می‌فرستد، برای هر گیرنده میزانی پردازش بایستی روی پیام انجام شودکه شامل استفاده از کلید عمومی هر یک از گیرندگان است. کاربر می‌تواند با استفاده از سرویس‌های S/MIME Mail List Agent (MLA) از این وظیفه رها شود. یک MLA می‌تواند یک پیام ورودی تنها را گرفته، رمزنگاری مختص گیرنده برای هر گیرنده را انجام داده و سپس پیام را به جلو راند. ارسال‌کنندهٔ اولیهٔ پیام تنها لازم است پیام را به MLA بفرستد که در این صورت رمزنگاری با کلید عمومی MLA انجام می‌شود.

۵-۳ منابع مطالعاتی

وِب سایت‌های مفید



- **PGP Home Page**: وِب سایت PGP مربوط به .PGP Corp فروشندهٔ پیشتاز محصولات PGP.
- **International PGP Home Page**: برای ارتقاء جهانی استفاده از PGP طراحی شده است. شامل اسناد و لینک‌های مرتبط است.
- **MIT Distribution Site for PGP**: توزیع‌کنندهٔ پیشتاز PGP رایگان. شامل FAQ و سایر اطلاعات بوده و لینک‌هائی نیز به سایت‌های مرتبط دارد.
- **PGP Charter**: آخرین RFCها و پیش‌نویس‌های اینترنت برای Open Specification PGP.
- **S/MIME Charter**: آخرین RFCها و پیش‌نویس‌های اینترنت در مورد S/MIME.

۵-۴ واژه‌های کلیدی، سؤالات مرورکنندهٔ بحث و مسائل

واژه‌های کلیدی

نوعی الگوریتم برای تبدیل داده‌های باینری	radix-64	امضاء جداشده	detached signature
کلید اجلاس	session key	پست الکترونیک	electronic mail
یک ساختار امنیتی برای پست الکترونیک	S/MIME	الحاقیه‌های چندمنظورهٔ پست الکترونیک	Multipurpose Internet Mail Extensions (MIME)
اعتماد	trust	یک ساختار امنیتی برای پست الکترونیک	Pretty Good Privacy (PGP)
یک نوع الگوریتم فشرده‌سازی	ZIP		

سؤالات مرورکننده بحث

۵-۱ پنج سرویس عمده‌ای که بتوسط PGP فراهم می‌آیند کدامند؟

۵-۲ فایدهٔ یک امضاء جدا شده چیست؟

۵-۳ چرا PGP یک امضاء را قبل از فشرده‌سازی تولید می‌کند؟

۵-۴ تبدیل R64 چیست؟

۵-۵ چرا تبدیل R64 برای یک کاربرد پست الکترونیک مفید است؟

۵-۶ چرا عمل قطعه‌قطعه کردن و دوباره سرهم کردن دیتا در PGP مورد نیاز است؟

۵-۷ چگونه PGP از مفهوم trust استفاده می‌کند؟

۵-۸ RFC 822 چیست؟

۵-۹ MIME چیست؟

۵-۱۰ S/MIME چیست؟

مسائل

۵-۱ PGP از مُود فیدبک رمز (CFB) الگوریتم CAST-128 استفاده می‌کند در حالی که اغلب کاربردهای رمزنگاری متقارن (بغیر از رمزنگاری کلید) از مُود زنجیره‌ای رمز قالبی (CBC) استفاده می‌کنند. داریم

$$\text{CBC:} \quad C_i = \mathrm{E}\,(K, [C_{i-1} \oplus \mathrm{P}_i]); \qquad \mathrm{P}_i = \mathrm{C}_{i-1} \oplus \mathrm{D}\,(K, C_i)$$
$$\text{CFB:} \quad \mathrm{C}_i = P_i \oplus \mathrm{E}\,(\mathrm{K}, C_{i-1}); \qquad \mathrm{P}_i = \mathrm{C}_i \oplus \mathrm{E}\,(K, \mathrm{C}_{i-1})$$

بنظر می‌رسد که هـــر دو روش امنیت یکسانی را فراهـم می‌سازند. دلیلی ارائه کنید که چرا PGP از مُود CFB اسـتفاده می‌کند.

۵-۲ در روش PGP، تعداد مورد انتظار کلیدهای اجلاس تولید شده، قبل از تکرار یک کلید اجلاس قبلاً خلق شده، چقدر است؟

۵-۳ در PGP، احتمال اینکه کاربری با N کلید عمومی، حداقل یک ID کلید تکراری داشته باشد چقدر است؟

۵-۴ اولین ۱۶ بیت چکیدهٔ پیام در یک امضاء PGP بصورت clear تفسیر می‌گردد.

الف- این امر تا چه حد امنیت الگوریتم hash را زیر سؤال می‌برد؟

ب- این امر واقعاً تا چه حد مقصود را که همانا کمک به درک این مطلب است که آیا کلید صحیح RSA برای رمزگشائی چکیده بکار رفته است، برآورده می‌نماید؟

۵-۵ در شکل ۴-۵ هر قلم در دسته‌کلید- عمومی شامل یک میدان trust است که میزان اعتماد مرتبط با صاحب این کلید- عمومی را نشان می‌دهد. چرا این کافی نیست؟ یعنی اگر این صاحب کلید مورد اعتماد است و این همان کلید عمومی اوست، چرا این اعتماد برای PGP کافی نیست تا این کلید عمومی را بکار برد.

۵-۶ تبدیل radix-64 را بعنوان نوعی رمزنگاری در نظر بگیرید. در این صورت کلیدی وجود ندارد. اما فرض کنید که یک دشمن تنها می‌داند که نوعی الگوریتم جایگذاری برای رمزکردن متن انگلیسی بکار رفته است. این الگوریتم در برابر شکستن رمز تا چه حد امن است؟

۵-۷ Phil Zimmermann، IDEA، 3DES سه کلیدی و CAST-128 را بعنوان الگوریتم‌های رمزنگاری متقارن برای PGP برگزید. دلایلی ذکر کنید که چرا الگوریتم‌های رمزنگاری متقارن زیر که در این کتاب مورد بحث قرار گرفته‌اند برای PGP مناسب و یا نامناسب‌اند: DES، 3DES دو کلیدی و AES.

ضمیمهٔ ۵- الف فشرده‌سازی دیتا با استفاده از ZIP

PGP از یک بستهٔ نرم‌افزاری مخصوص فشرده‌سازی بنام ZIP استفاده می‌کندکه بتوسط Jean-lup Gailly، Mark Adler و Richard Wales نوشته شده است. ZIP یک نرم‌افزار رایگان بوده که به زبان C نوشته شده است و بعنوان یک برنامهٔ سودمند روی UNIX و بعضی سیستم‌های دیگر اجرا می‌شود. ZIP از نظر عملیاتی معادل PKZIP است که یک اشتراک‌افزار پراستفاده در سیستم‌های Windows بوده و به توسط PKWARE,Inc. تهیه شده است. الگوریتم zip شاید معمول‌ترین تکنیک فشرده‌سازی در سیستم عامل‌های متفاوت بوده و نسخه‌های رایگان و اشتراکی آن برای Macintosh و سایر سیستم‌ها ازجمله Windows و UNIX موجود است.

Zip و الگوریتم‌های مشابه آن از تحقیقات Jacob Ziv و Abraham Lempel سرچشمه می‌گیرند. در سال ۱۹۷۷ میلادی، آنها روشی را که بر پایهٔ یک حافظهٔ موقت از نوع پنجرهٔ لغزان قرار داشته و آخرین متن پردازش شده را نگاه می‌داشت، توصیف نمودند[ZIV77]. از این الگوریتم معمولاً با نام LZ77 یاد می‌شود. نسخه‌ای از این الگوریتم در روش فشرده‌سازی zip بکار گرفته شده است (PKZIP، gzip، zipit و غیره).

LZ77 و انواع دیگر آن از این واقعیت استفاده می‌کنند که کلمات و جملات یک متن (صور تصویری در مورد GIF) دارای تکرارهای احتمالی هستند. وقتی تکرار واقع می‌شود، ردیف تکرارشده را می‌توان با یک کُد کوتاه جایگزین نمود. برنامهٔ فشرده‌سازی به دنبال چنین تکرارهائی گشته و کُدهائی را برای جایگزینی دنباله‌های تکرارشده تولید می‌نماید. در طول زمان از کُدها برای پیداکردن دنباله‌های جدید استفاده می‌شود. الگوریتم بایستی بنحوی تعریف شود که برنامهٔ بازکننده قادر به کُدگشائی و بازیابی متن اصلی داده‌ها باشد.

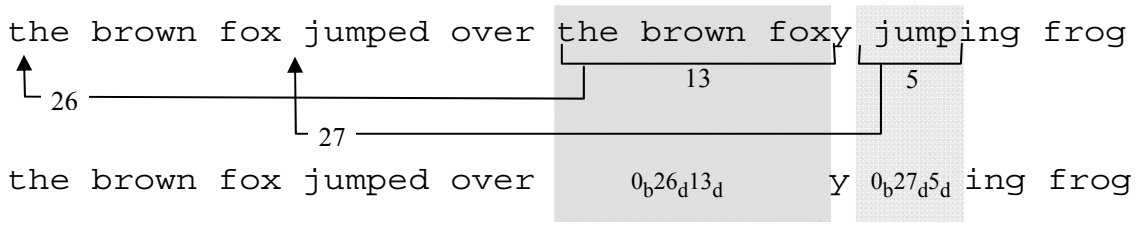
قبل از مطالعهٔ جزئیات LZ77 اجازه دهید تا به یک مثال ساده بپردازیم. جملهٔ بی‌معنی زیر را

the brown fox jumped over the brown foxy jumping frog

که دارای طول ۵۳ اُکتت = ۴۲۴ بیت است را در نظر بگیرید (این مثال از[WEIS93] اقتباس شده است). الگوریتم این متن را، از چپ به راست پردازش می‌کند. در ابتدا هر کاراکتر بصورت یک پترن ۹- بیتی که شامل یک بیت 1 و به دنبال آن نمایش ۸- بیتی کُد ASCII آن کاراکتر است در می‌آید. همین‌طور که پردازش ادامه می‌یابد، الگوریتم به دنبال دنباله‌های تکراری می‌گردد. وقتی به یک تکرار برخورد می‌کند، الگوریتم به اسکن خود ادامه داده تا تکرار خاتمه یابد. بعبارت دیگر هربار تکراری واقع می‌شود، الگوریتم هر تعداد کاراکتر را که ممکن است جایگزین می‌کند. اولین دنبالهٔ تکراری در جملهٔ بالا، **the brown fox** است. این دنباله بتوسط یک نشانگر به دنبالهٔ قبلی و همچنین طول دنباله جایگزین می‌شود. در این مورد، دنبالهٔ قبلی **the brown fox** در ۲۶ کاراکتر قبل واقع شده و طول دنبالهٔ تکرارشده ۱۳ کاراکتر است. برای این مثال، دو راه حل برای کُدینگ تصور کنید: یک نشانگر ۸- بیتی و یک طول ۴- بیتی، یا یک نشانگر ۱۲- بیتی و یک طول ۶- بیتی. یک سرآیند۲- بیتی نشان می‌دهد که کدام روش انتخاب شده است، 00 نمایش‌دهندهٔ روش اول و 01 نمایش‌دهندهٔ روش دوم است. بنابراین دومین وقوع **the brown fox** بصورت <00_b><26_d><13_d> و یا 1101 00011010 00 کُد می‌شود.

بخش‌های باقیماندهٔ پیام فشرده شده، حرف **y**، دنبالهٔ $<0_b><27_d><5_d>$ که جایگزین دنبالهٔ شامل کاراکتر space و به دنبال آن **jump** می‌شود و دنبالهٔ کاراکترهای **ing frog** است می‌گردد.

شکل ۹-۵ نگاشت فشرده‌سازی را نشان می‌دهد. پیام فشرده شده شامل ۳۵ کاراکتر ۹- بیتی و دو کُد است که عملاً ۳۴۳= ۱۴ × ۲ + ۹ × ۳۵ بیت می‌شود. اگر این پیام فشرده شده را با پیام فشرده نشدهٔ اصلی که شامل ۴۲۴ بیت است مقایسه کنیم، نسبت فشرده‌سازی برابر ۱٫۲۴ بدست می‌آید.



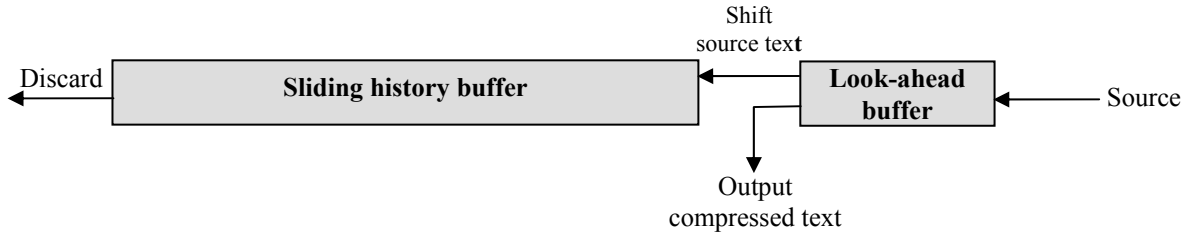
شکل۹-۵ مثالی از روش LZ77

الگوریتم فشرده‌سازی

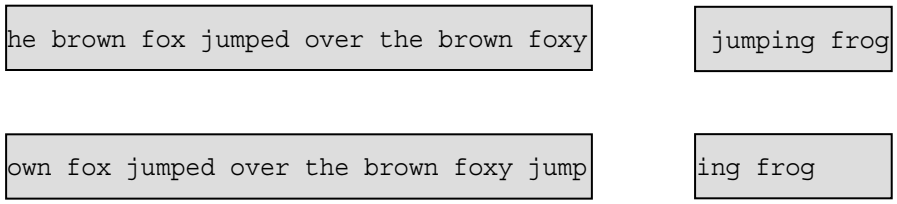
الگوریتم فشرده‌سازی LZ77 و انواع متنوع دیگر آن از دو حافظهٔ موقت استفاده می‌کنند. یک **حافظهٔ موقت sliding history** شامل آخرین N کاراکتر منبع که مورد پردازش قرار گرفته‌اند بوده و یک **حافظهٔ موقت look-ahead** که شامل L کاراکتر بعدی است که بایستی پردازش شوند (شکل۱۰-۵الف). الگوریتم تلاش می‌کند تا دو یا چند کاراکتر از شروع حافظهٔ موقت look-ahead را با یک دنباله در حافظهٔ موقت sliding history تطبیق دهد. اگر چنین تطبیقی یافت نشود، اولین کاراکتر در حافظهٔ look-ahead بصورت یک کاراکتر ۹- بیتی خارج شده و به درون پنجرهٔ لغزان شیفت داده می‌شود و ازآن طرف قدیمی‌ترین کاراکتر درون پنجرهٔ لغزان نیز بیرون رانده می‌شود. اگر تطبیقی یافت شود، الگوریتم به اسکن کردن ادامه داده تا طویل‌ترین تطبیق را پیدا کند. آنگاه دنبالهٔ تطبیق یافته بصورت یک میدان سه‌تائی (نمایشگر، نشانگر، طول) خارج می‌شود. برای یک دنبالهٔ K تائی، قدیمی‌ترین K کاراکتر موجود در پنجرهٔ لغزان بیرون رانده شده و K کاراکتر دنبالهٔ کُدشده به داخل پنجره رانده می‌شوند.

شکل ۱۰-۵ب این عملیات را بر روی دنبالهٔ مثال ما نشان می‌دهد. در این نمایش یک پنجرهٔ لغزان ۳۹-کاراکتری و یک حافظهٔ موقت look-ahead با ۱۳ کاراکتر در نظر گرفته شده‌اند. بخش بالای شکل، اولین ۴۰ کاراکتر پردازش شده و نسخهٔ فشرده نشدهٔ اخیرترین ۳۹ کاراکتر در داخل پنجرهٔ لغزان است. بقیهٔ کاراکترهای منبع در پنجرهٔ look-ahead قرار دارند. الگوریتم فشرده‌سازی تطبیق بعدی را تعیین نموده، ۵ کاراکتر را از حافظهٔ موقت look-ahead به داخل پنجرهٔ لغزان رانده و کُد خروجی این دنباله را تعیین می‌کند. وضعیت حافظهٔ موقت پس از این عملیات در قسمت پائین شکل نشان داده شده است.

در حالیکه LZ77 مفید بوده و سعی در تطبیق خود با ماهیت داده‌های ورودی دارد، ولی دارای نقاط ضعفی نیز هست. الگوریتم برای جستجو و تطبیق در متن قبلی از یک پنجرهٔ محدود استفاده می‌کند. برای یک بلوک خیلی طولانی از متن، که قابل مقایسه با اندازهٔ پنجره باشد، خیلی از تطبیق‌های مؤثر حذف می‌شوند. اندازهٔ پنجره را می‌توان افزایش داد ولی این امر شامل دو پنالتی خواهد بود: (۱) زمان پردازش الگوریتم افزایش می‌یابد زیرا الگوریتم بایستی برای هر مکان پنجرهٔ لغزان یک مقایسهٔ دنباله‌ای با حافظهٔ موقت look-ahead انجام دهد و (۲) میدان< نشانگر > بایستی وسیع‌تر بوده تا پرش‌های بزرگتری را امکان‌پذیر سازد.



(الف) ساختار عمومی



(ب) مثال

شکل ۱۰-۵ روش LZ77

الگوریتم معکوس فشرده‌سازی

از فشردگی خارج کردن یک متن فشرده شده بتوسط LZ77 کار ساده‌ای‌است. الگوریتم این عمل بایستی آخرین N کاراکتر خروجی بازشده را ذخیره نماید. وقتی به یک دنبالهٔ کُدشده برخورد می‌شود، الگوریتم بازکننده از میدان‌های< نشانگر> و < طول> استفاده کرده و کُد را با دنبالهٔ واقعی متن جایگزین می‌نماید.

ضمیمهٔ ۵- ب تبدیل Radix-64

هم PGP و هم S/MIME از یک روش کُدینگ که تبدیل radix-64 خوانده می‌شود استفاده می‌کنند. این تکنیک هر ورودی باینری دلخواه را به خروجی‌های قابل چاپ تبدیل می‌کند. فرم کُدکردن دارای خصوصیات مرتبط زیر است:

۱- برد تابع یک مجموعه از کاراکترهاست که بطور جهانی در هر سایتی قابل نمایش است و نه یک کُد باینری خاص از این مجموعهٔ کاراکتری. بنابراین خود کاراکترها می‌توانند بتوسط یک سیستم خاص به هر فرمی‌که لازم است کُد شوند. بعنوان مثال، کاراکتر "E" در یک سیستم مبتنی بر کُد ASCII بصورت 45 هکزادسیمال و در یک سیستم مبتنی بر کُد EBCDIC بصورت C5 هکزادسیمال نشان داده می‌شود:

۲- مجموعهٔ کاراکتری شامل ۶۵ کاراکتر قابل چاپ است که یکی از آنها برای لائی (padding) بکار می‌رود. با ۲۶=۶۴ کاراکتر موجود، هر کاراکتر میتواند برای نمایش ۶ بیت ورودی بکار رود.

۳- هیچ کاراکتر کنترلی در مجموعه وجود ندارد. بنابراین یک پیام کُدشده بصورت radix-64 می‌تواند در یک سیستم پستی که دنبالهٔ دیتا را بمنظور یافتن کاراکترهای کنترلی اسکن می‌کند به جلو رانده شود.

۴- کاراکتر خط فاصله ("-") بکار نمی‌رود. این کاراکتر در RFC 822 دارای استفادهٔ خاص بوده و بنابراین بایستی در اینجا از آن پرهیز شود.

جدول ۹-۵ کُدینگ Radix-64

6-bit value	Character encoding	6-bit value	Character encoding	6-bit value	Character encoding	6-bit value	Character encoding
0	A	16	Q	32	g	48	w
1	B	17	R	33	h	49	x
2	C	18	S	34	i	50	y
3	D	19	T	35	j	51	z
4	E	20	U	36	k	52	0
5	F	21	V	37	l	53	1
6	G	22	W	38	m	54	2
7	H	23	X	39	n	55	3
8	I	24	Y	40	o	56	4
9	J	25	Z	41	p	57	5
10	K	26	a	42	q	58	6
11	L	27	b	43	r	59	7
12	M	28	c	44	s	60	8
13	N	29	d	45	t	61	9
14	O	30	e	46	u	62	+
15	P	31	f	47	v	63	/
						(pad)	=

جدول ۹-۵ نگاشت مقادیر۶- بیتی ورودی به کاراکترها را نشان می‌دهد. مجموعهٔ کاراکترها شامل کاراکترهای حرفی و عددی باضافهٔ "+" و "/" است. کاراکتر "=" بعنوان کاراکتر padding مورد استفاده قرار می‌گیرد.

شکل ۱۱-۵ روش سادهٔ نگاشت را نشان می‌دهد. ورودی باینری بصورت بلوک‌های ۳- اُکتتی یا ۲۴- بیتی پردازش می‌شوند. هر گروه ۶- بیتی در بلوک ۲۴- بیتی به یک کاراکتر نگاشت می‌شود. در شکل کاراکترها بصورت مقادیر ۸ - بیتی کُد شده‌اند. در موارد معمول، یک ورودی ۲۴ - بیتی بصورت یک خروجی ۳۲ - بیتی توسعه می‌یابد.

برای مثال، دنبالهٔ متن خام ۲۴- بیتی 10010001 01011100 00100011 که در فرم هکزادسیمال می‌تواند بصورت 235C91 نشان داده شود را در نظر بگیرید. این ورودی را بصورت بلوک‌های ۶- بیتی مرتب می‌کنیم:

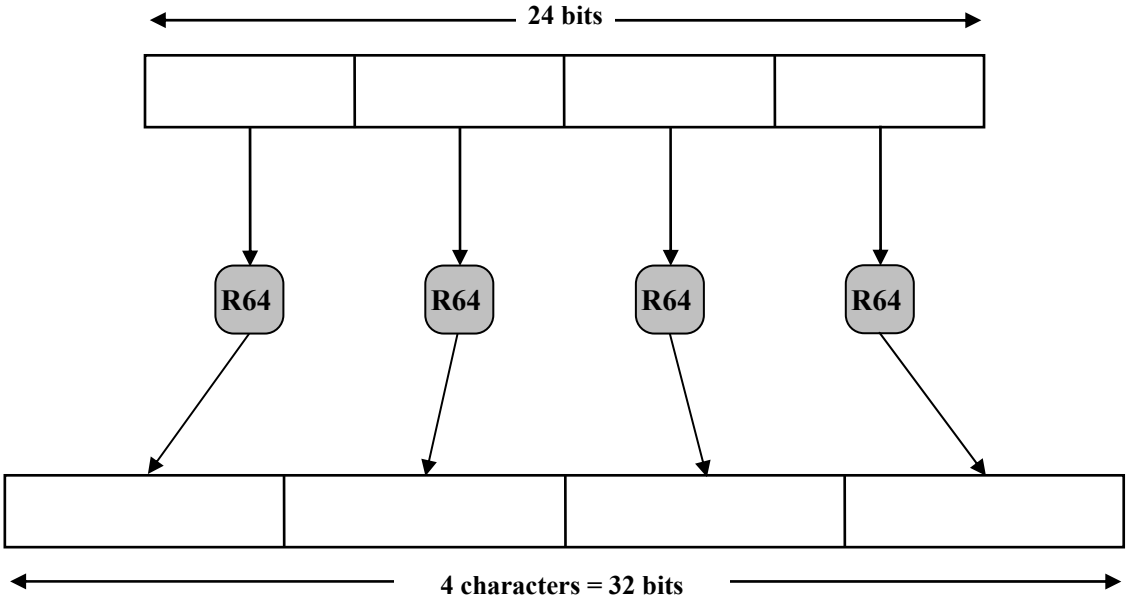
001000 110101 110010 010001

اندازه‌های دهدهی نظیر ۶- بیتی‌ها ۸، ۵۳، ۵۰ و ۱۷ هستند. با مراجعه به جدول ۹-۵ کُد radix-64 این دنباله I1yR خواهد بود. اگر این کاراکترها به فرمت ASCII با بیت parity صفر نگاشت شوند، خواهیم داشت

01001001 0011001 01111001 01010010

در هکزادسیمال نمایش این دنباله بصورت 49317952 خواهد بود. بطور خلاصه،

دادهٔ ورودی	
نمایش باینری	00100011 01011100 10010001
نمایش هکزادسیمال	235C91
کُدینگ Radix-64 دادهٔ ورودی	
نمایش علائم	I1yR
کُد ASCII(8 bit,zero parity)	01001001 00110001 01111001 01010010
نمایش هکزادسیمال	49317952



شکل ۱۱-۵ کُدینگ قابل چاپ داده‌های باینری به فرمی Radix-64

ضمیمهٔ ۵- ج تولید اعداد تصادفی در PGP

PGP از یک روش پیچیده و قدرتمند برای تولید اعداد تصادفی و اعداد شبه‌تصادفی، برای منظورهای متفاوت، استفاده می‌کند. PGP اعداد تصادفی را از نوع و زمان حرکت کلیدها بتوسط کاربر، و اعداد شبه تصادفی را با استفاده از یک الگوریتم که مبتنی بر روشی در ANSI X9.17 است تولید می‌کند. PGP از این اعداد برای مقاصد زیر استفاده می‌کند:

- اعداد تصادفی واقعی:
  - برای تولید جفت کلیدهای RSA
  - بعنوان بذر اولیه در تولید اعداد شبه‌تصادفی
  - برای تولید ورودی دیگری در خلال تولید اعداد شبه‌تصادفی
- اعداد شبه‌تصادفی:
  - برای تولیدکلیدهای اجلاس
  - برای تولید بردارهای شروع (IV) همراه با کلید اجلاس در مُود رمزنگاری CFB

اعداد تصادفی واقعی

PGP یک حافظهٔ موقت ۲۵۶- بایتی از بیت‌های تصادفی را نگاه می‌دارد. هربار که PGP انتظارحرکت کلیدی را دارد، زمان شروع انتظار را بصورت یک فرمت ۳۲- بیتی ثبت می‌کند. وقتی حرکت کلید دریافت می‌شود، زمان حرکت کلید و نوع کلید فشرده‌شده با یک اندازهٔ ۸- بیتی ثبت می‌گردد. اطلاعات زمان و حرکت کلید برای تولید یک کلید رمز بکار گرفته شده که این کلید بنوبهٔ خود برای رمزکردن اندازهٔ جاری حافظهٔ موقت بیت- تصادفی بکار می‌رود.

اعداد شبه‌تصادفی

تولید عدد شبه‌تصادفی از یک بذر ۲۴- اُکتتی استفاده کرده و یک کلید اجلاس ۱۶- اُکتتی ، یک بردار شروع ۸- اُکتتی و یک بذر جدید برای استفاده در دور بعدی تولید عدد تصادفی را تولید می‌کند. الگوریتم مورد استفاده مبتنی بر الگوریتم X9.17 بوده که برای رمزنگاری بجای DES از CAST-128 استفاده می‌کند. الگوریتم از ساختمان دادهٔ زیر استفاده می‌کند:

۱- ورودی

- randseed.bin (۲۴ اُکتت): اگر این فایل خالی باشد، با ۲۴ اُکتت تصادفی واقعی پر می‌شود.
- message: کلید اجلاس و IV که از آنها برای رمزنگاری یک پیام استفاده می‌شود خود تابعی از آن پیام هستند. این امر به تصادفی‌تر شدن کلید و IV کمک می‌کند و اگر یک دشمن قبلاً متن سادهٔ پیام را پیدا کرده باشد ظاهراً نیازی به کلید اجلاس یکبار- مصرف نیست.

۲- خروجی

- K (۲۴ اُکتت): اولین ۱۶ اُکتت،K[0…15]، شامل یک کلید اجلاس و آخرین ۸ اُکتت،K[16…23]، شامل یک IV است.
- randseed.bin (۲۴ اُکتت): یک اندازهٔ جدید بذر در این فایل قرار می‌گیرد.

۳- ساختمان دادهٔ داخلی

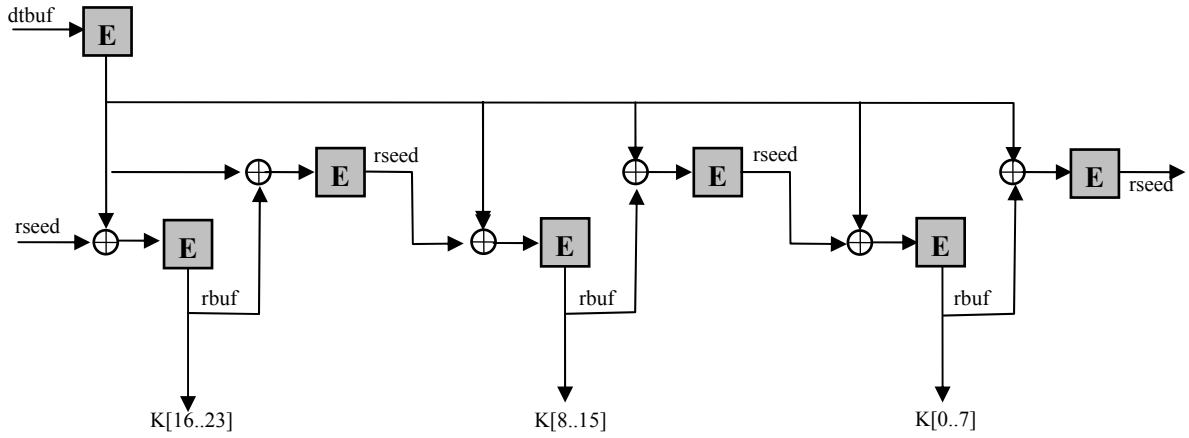
- dtbuf (۸ اُکتت): ۴ اُکتت اول، dtbuf[0…3]، در ابتدا با اندازه‌های جاری تاریخ/زمان پر می‌شوند. این حافظهٔ موقت، معادل متغیر DT در الگوریتم X12.17 است.
- rkey (۱۶ اُکتت): کلید رمزنگاری CAST-128 که در تمام مراحل الگوریتم از آن استفاده می‌شود.
- rseed (۸ اُکتت): معادل متغیر V_i در X12.17 .
- rbuf (۸ اُکتت): یک عدد شبه‌تصادفی که بتوسط الگوریتم تولید می‌شود. این حافظهٔ موقت معادل متغیر R_i در X12.17 است.
- K' (۲۴ اُکتت): حافظهٔ موقت برای اندازهٔ جدید randseed.bin.

الگوریتم شامل نه قدم G1 تا G9 است. قدم‌های اول و آخر قدم‌های ابهام‌زائی به منظور کاهش ارزش یک فایل randseed.bin در صورت کشف دشمن است. قدم‌های باقیمانده ضرورتاً معادل سه بار تکرار الگوریتم X12.17 بوده و در شکل ۱۲-۵ نشان داده شده است. بطور خلاصه:

G1. [Prewash previous seed]

الف- randseed.bin را در K[0…23] کپی کنید.

ب- hash پیام را حساب کنید (اگر پیام امضاء شده باشد این مقدار قبلاً محاسبه شده است. در غیر این‌صورت اولین 4K اُکتت پیام بکار خواهد رفت). از نتیجه به عنوان یک کلید استفاده کنید، از یک IV صفر استفاده کرده و K را در مُود CFB رمزنگاری نمائید. نتیجه را در K ذخیره نمائید.



شکل ۱۲-۵ تولید کلید اجلاس و IV در PGP (قدم‌های G2 تا G8)

G2 . [Set initial seed]

الف- dtbuf[0…3] را با ۳۲- بیت زمان محلی تنظیم کنید. dtbuf[4…7] را تماماً صفر بگذارید. rkey←K[0…15] و rseed←K[16…23] .

ب- dtbuf ۶۴- بیتی را با استفاده از rkey ۱۲۸- بیتی در مُود ECB رمزنگاری کرده و نتیجه را در dtbuf ذخیره کنید.

G3 . [Prepare to generate random octets] rcount← 0 و k← 23. قدم‌های G4-G7 بصورت یک حلقه ۲۴ بار (k=23…0) اجرا می‌شوند که هر بار برای یک اُکتت تصادفی تولید شده و قرارداده شده در K است. متغیر rcount تعداد اُکتت‌های تصادفی استفاده نشده در rbuf را نشان می‌دهد. برای تولید ۲۴ اُکتت سه بار از 8 تا 0 رو به پائین شمارش می‌شوند.

G4 . [Bytes available?] اگر rcount = 0 به G5 و در غیر اینصورت به G7 بروید. قدم‌های G5 و G6 یک مرتبه الگوریتم X12.17 را برای تولید یک گروه هشت‌تائی از اُکتت‌های تصادفی اجرا می‌کنند.

G5 . [Generate new random octets]

الف- rseed ← rseed ⊕ dtbuf

ب – rseed ← E (rkey , rseed) در مُود ECB.

G6 . [Generate next seed]

الف- rseed ← rbuf ⊕ dtbuf

ب- rseed ← E (rkey , rseed) در مُود ECB

ج- rcount ← 8 قرار داده شود.

[Transfer one byte at a time from rbuf to K] . G7

الف- rcount ← rcount-1 قرار داده شود.

ب- یک بایت تصادفی واقعی b تولید کرده و K[k] ← rbuf[rcount] ⊕ b

[Done?] If k=0 goto G9 else set k← k-1 and goto G4 . G8

[Postwash seed and return result] . G9

الف- ۲۴ بایت دیگر بتوسط روش G4-G7 تولید کنید باستثنای اینکه در G7 عمل XOR بایت تصادفی را انجام ندهید.

ب- 'K را با کلید K[0…15] و IV را با کلید K[16…23] در مُود CFB رمزنگاری نمائید. نتیجه را در randseed.bin ذخیره کنید.

ج- K را برگردانید.

قاعدتاً نبایستی بتوان کلید اجلاس را از ۲۴ اُکتت جدید تولیدشده در قدم G9 الف تعیین نمود. با وجود این برای اطمینان از اینکه فایل randseed.bin ذخیره‌شده هیچگونه اطلاعاتی در مورد آخرین کلید اجلاس به دست نمی‌دهد، ۲۴ اُکتت جدید، رمزنگاری شده و نتیجهٔ عمل بعنوان بذر جدید ذخیره می‌شود.

این الگوریتم پیچیده قاعدتاً اعداد شبه‌تصادفی قدرتمندی را فراهم می‌آورد.

فصل ۶

امنیت IP

جمعیت اینترنت، مکانیسم‌های امنیتی متفاوتی را برای کاربردهای مختلف و مختص به آنها طراحی نموده است که شامل پست الکترونیک (S/MIME, PGP)، کلاینت/ سرور (Kerberos)، دست‌یابی به وب (Secure Sockets Layer) و غیره است. با وجود این، کاربران در رابطه با امنیت دارای نگرانی‌های هستند که مربوط به لایه‌های پروتکلی است. بعنوان مثال یک بنگاه تجاری بزرگ می‌تواند یک شبکهٔ TCP/IP خصوصی امن را با جلوگیری از ارتباط با سایت‌های غیرمطمئن، رمزنگاری بسته‌هائی که از سازمان خارج می‌شوند و اعتبارسنجی بسته‌های دیتائی که به سازمان وارد می‌شوند بوجود آورد. با پیاده‌سازی امنیت در سطح IP، یک سازمان می‌تواند شبکه‌ای امن، نه تنها برای کاربردهائی که مکانیسم امنیتی دارند، بلکه برای بسیاری از کاربردهائی که از امنیت بی‌بهره اند بوجود آورد.

امنیت سطح IP سه محدودهٔ عملیاتی را در بر می‌گیرد: اعتبارسنجی، محرمانگی و مدیریت کلید. مکانیسم اعتبارسنجی این اطمینان را ایجاد می‌کند که یک بستهٔ دریافت شده در واقع بتوسط همان واحدی که در سرآیند بسته مشخص شده ارسال گردیده است. بعلاوه این مکانیسم اطمینان می‌دهد که بسته در مسیر ترانزیت بین فرستنده و گیرنده تغییر نکرده است. سرویس محرمانگی، گره‌های مرتبط را قادر می‌سازد پیام‌ها را رمزنگاری کرده تا از استراق سمع اشخاص ثالث محفوظ بمانند. تسهیلات مدیریت کلید، مرتبط با مبادلهٔ امن کلیدهاست.

این فصل را با مروری بر امنیت IP (IPSec) و معرفی معماری IPSec آغاز می‌کنیم. آنگاه به هریک از سه سطح عملیاتی نگاهی مفصل می‌اندازیم. ضمیمهٔ این فصل، مروری بر پروتکل‌های اینترنت است.

۶-۱ مروری بر امنیت IP

در سال ۱۹۹۴ میلادی، گروه معماری اینترنت (IAB) گزارشی را با عنوان « امنیت در معماری اینترنت» ارائه نمودند (RFC1636). گزارش بیانگر این اتفاق نظر بود که اینترنت نیاز به امنیت بیشتر و بهتری دارد. رئوس کلیدی این نیازها نیز در این گزارش ذکر شده بود. در بین اینها، نیاز به امن ماندن زیرساخت شبکه از پایش‌های غیرمجاز، نیاز به کنترل ترافیک شبکه و نیاز به امن نگاه داشتن ترافیک بین یک کاربر انتهائی و کاربر انتهائی دیگر با استفاده از سازوکارهای اعتبارسنجی و رمزنگاری وجود داشت.

این نگرانی‌ها کاملاً بجا هستند. در تأئید آن، گزارش سالیانهٔ ۲۰۰۱ تیم پاسخگوئی به فوریت‌های کامپیوتری (CERT) قریب به ۵۲,۰۰۰ پیشامد امنیتی را لیست نموده است. جدی‌ترین حملات، IP Spoofing بوده که در آن مهاجمین بسته‌هائی را با آدرس‌های IP جعلی خلق کرده و کاربردهائی که از اعتبارسنجی مبتنی بر IP استفاده می‌نمایند را مورد سوء استفاده قرار داده بودند. همچنین فرم‌های متفاوت استراق سمع و بوکشیدن بسته‌ها که درآن مهاجمین، اطلاعات ارسال شده شامل اطلاعات logon و محتویات پایگاه‌های داده، را خوانده بودند مشاهده می‌شد.

در پاسخ به این مقوله‌ها، IAB اعتبارسنجی و رمزنگاری را بعنوان مشخصه‌های امنیتی لازم در نسل بعد IP که بنام IPv6 نامیده شده بود جا داد. خوشبختانه این قابلیت‌های امنیتی طوری طراحی شده بودند که بتوانند هم در نسخهٔ فعلی IPv4 و هم در نسخهٔ آتی IPv6 قابل استفاده باشند. این بدین معنی است که فروشندگان محصولات کامپیوتری می‌توانند این مشخصه‌ها را در محصولات خود عرضه نمایند که هم اکنون بسیاری از آنها قابلیت‌های IPSec را در این محصولات گنجانده‌اند.

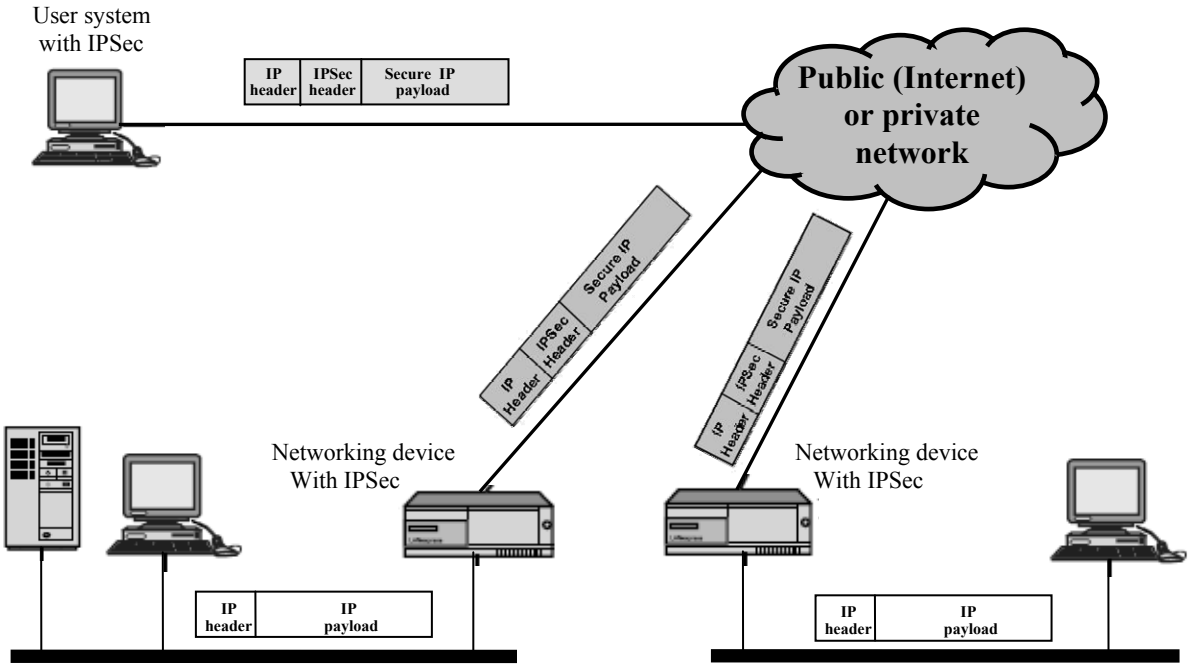
کاربردهای IPSec

IPSec قابلیت ارتباطات امن در عرض یک شبکهٔ LAN، در عرض شبکه‌های خصوصی و عمومی WAN و در عرض اینترنت را فراهم می‌آورد. مثال‌هائی از استفاده از IPSec بقرار زیر است:

- **اتصال امن شاخه‌های اداری از طریق اینترنت**: یک کمپانی می‌تواند یک شبکهٔ خصوصی مجازی امن را روی اینترنت و یا روی یک WAN عمومی بنا نماید. این امر باعث می‌شود تا این کسب وکار عمدتاً متکی به اینترنت بوده و نیاز آن به شبکه‌های خصوصی کمتر شود. در نتیجه هم هزینه و هم سربارهٔ مدیریت شبکه کاهش می‌یابد.
- **دست‌یابی امن به دوردست از طریق اینترنت**: یک کاربر انتهائی که سیستم او به پروتکل‌های امنیتی IP مجهز است می‌تواند به یک فراهم‌آورندهٔ سرویس اینترنتی(ISP) تلفن زده و به شبکهٔ یک کمپانی دسترسی یابد. این امر هزینهٔ تردد اداری کارمندان را کاهش خواهد داد.
- **برقراری ارتباط اینترانتی و اِکسترانتی با شرکاء**: IPSec می‌تواند برای ایمن‌سازی ارتباطات با سایر سازمان‌ها بکار رود که اطمینان از اعتبارسنجی صحیح و محرمانگی از خواص آن بوده و مکانیسم مبادلهٔ کلید نیز در آن فراهم است.
- **ارتقاء امنیت تجارت الکترونیک**: با وجود اینکه تعدادی از کاربردهای مربوط به تجارت الکترونیک و وِب، در پروتکل‌های امنیتی فراهم شده در محصولات موجود می‌باشند، ولی استفاده از IPSec این امنیت را ارتقاء می‌بخشد.

مشخصهٔ اصلی IPSec که آن را قادر می‌سازد تا از این کاربردهای متنوع حمایت نماید این است که می‌تواند تمام ترافیک در سطح IP را رمزنگاری و/ یا اعتبارسنجی کند. بنابراین تمام کاربردهای توزیع شده که شامل اتصال از دور، کلاینت/ سِرور، e-mail، انتقال فایل، دسترسی به وِب و غیره‌اند می‌توانند امن باشند.

شکل ۱-۶ یک سناریوی معمول استفاده از IPSec را نشان می‌دهد. یک سازمان می‌تواند LAN های متفاوتی را در موقعیت‌های جغرافیائی مختلف داشته باشد. برای هر LAN، ترافیک غیرامن IP در نظر گرفته شده است ولی برای ترافیک خارج از شبکه‌ها که از طریق نوعی WAN خصوصی یا عمومی صورت می‌پذیرد از پروتکل‌های IPSec استفاده می‌شود. این پروتکل‌ها در تجهیزات شبکه همچون یک مسیریاب و یا یک دیوار آتش، که یکLAN را به دنیای خارج پیوند می‌دهند،کار می‌کنند. تجهیزات شبکه‌ای IPSec معمولاً تمام ترافیک داخل‌شونده به WAN را فشرده سازی و رمزنگاری نموده و تمام ترافیک خروجی از WAN و ورودی به LAN را از فشردگی درآورده و رمزگشائی می‌نماید. تمام این عملیات برای ایستگاه‌های کاری و سِرورهای LAN نامرئی هستند. ارسال اطلاعات بصورت امن همچنین برای کاربران منفردی که از طریق تماس تلفنی وارد WAN می‌شوند نیز ممکن است. این ایستگاه‌های کاری بایستی پروتکل‌های IPSec را برای فراهم آوردن امنیت در درون خود پیاده‌سازی کنند.



شکل ۱-۶ یک سناریو برای امنیت IP

مزایای IPSec

[MARK97] مزایای زیر را برای IPSec ذکر می‌کند:

- وقتی IPSec در یک دیوار آتش یا مسیریاب بکار گرفته می‌شود، یک امنیت محکم برای تمام ترافیکی که از محدودهٔ این دو دستگاه عبور می‌کند فراهم می‌آورد. ترافیک داخل سازمان یا یک گروه کاری، از بکارگیری سربارهٔ مرتبط با پردازش‌های امنیتی آزادند.
- اگر تمام ترافیک خارج از محدوده بایستی از IP استفاده کنند و دیوار آتش تنها راه ورودی اینترنت به سازمان باشد، IPSec در این دیوار آتش در برابر نادیده گرفته شدن و میان بُر زدن دیتا مقاوم است.
- IPSec در زیرِ لایهٔ حمل‌ونقل (UDP,TCP) قرار گرفته و بنابراین برای کاربردها نامرئی است. وقتی IPSec در یک مسیریاب یا دیوار آتش بکار گرفته می‌شود، نیازی به تعویض نرم‌افزارهای کاربران و یا سِرورها نیست. حتی اگر IPSec در سیستم انتهائی هم بکار گرفته شود نرم‌افزارهای لایه‌های بالاتر که شامل کاربردها هم هستند تحت تأثیر واقع نمی‌شوند.
- IPSec می‌تواند برای کاربران انتهائی نامرئی باشد. نیازی نیست که کاربران را نسبت به مکانیسم‌های امنیتی آموزش داد و مثلاً لازم نیست خلق اقلام کلید برای هر کاربر و یا ابطال اقلام کلید در هنگام ترک سازمان را به آنها آموخت.

- IPSec می‌تواند در صورت لزوم امنیت را برای تک‌تک کاربران فراهم آورد. این مورد برای کار در خارج از محل سازمان و همچنین برای ایجاد یک زیرشبکهٔ مجازی در درون سازمان برای کاربردهای حساس مناسب است.

کاربردهای مسیریابی

علاوه بر حمایت از کاربران انتهائی و محافظت از سیستم‌ها و شبکه‌ها، IPSec می‌تواند نقشی حیاتی در معماری مسیریابی مورد نیاز عملیات بین‌شبکه‌ای داشته باشد. [HUIT98] مثال‌های زیر استفاده از IPSec را لیست کرده است. IPSec اطمینان می‌دهد که:

- اعلان حضور یک مسیریاب (یک مسیریاب حضور خود را اعلان می‌کند)، از یک مسیریاب معتبر آمده است.
- اعلان حضور یک مسیریاب به همسایگان (یک مسیریاب به دنبال برقراری و یا نگهداری یک رابطهٔ همسایگی با مسیریاب دیگر است)، از یک مسیریاب معتبر آمده است.
- یک پیام تغییر مسیر از همان مسیریابی آمده است که بستهٔ اولیه دیتا برای او ارسال شده بود.
- بروزرسانی یک مسیریاب، جعلی نیست.

بدون چنین معیارهای امنیتی، یک دشمن می‌تواند ارتباطات را مختل کرده و یا مسیر ترافیک را عوض کند. پروتکل‌های مسیریابی، مانند OSPF، بایستی در بالای اتحادهای امنیتی تعریف شده بین مسیریاب‌ها بتوسط IPSec کار کنند.

۶-۲ معماری امنیت IP

مشخصه‌های IPSec بسیار پیچیده شده‌اند. برای اینکه درکی ازکلّ معماری IPSec حاصل شود، به اسنادی که IPSec را تعریف می‌کنند نگاهی می‌اندازیم. آنگاه سرویس‌های IPSec را تعریف کرده و مفهوم اتحاد امنیتی (SA) را معرفی می‌کنیم.

اسناد IPSec

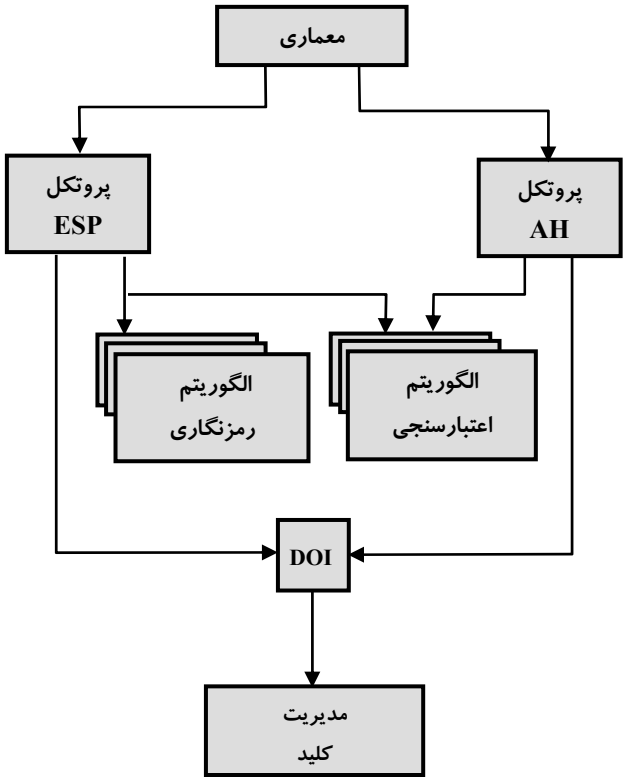
مشخصه‌های IPSec شامل اسناد متعددی است. مهم‌ترین آنها که در نوامبر ۱۹۹۸ میلادی منتشر شد، RFCهای 2401، 2402، 2406 و 2408 است:

- RFC 2401: مروری بر یک معماری امنیتی
- RFC 2402: توصیف یک الحاقیهٔ اعتبارسنجی بسته دیتا به IPv4 و IPv6
- RFC 2406: توصیف یک الحاقیهٔ رمزنگاری بستهٔ دیتا به IPv4 و IPv6
- RFC 2408: تعیین قابلیت‌های مدیریت کلید

تبعیت از این مشخصه‌ها برای IPv6 اجباری و برای IPv4 اختیاری است. در هردو مورد، مشخصه‌های امنیتی بصورت سرآیندهای الحاقی که بعد از سرآیند IP قرار می‌گیرند پیاده‌سازی می‌شوند. سرآیند الحاقی مربوط به اعتبارسنجی بنام سرآیند اعتبارسنجی (AH) و سرآیند الحاقی مربوط به رمزنگاری بنام سرآیندکپسولی‌کردن محمولهٔ امنیتی (ESP) نامیده می‌شود.

علاوه بر این چهار RFC، تعداد دیگری از پیش‌نویس‌ها از سوی IP Security Protocol Working Group که بتوسط IETF تأسیس شده است منتشر گردیده است. اسناد به هفت گروه، مطابق شکل ۲-۶ (RFC 2401) تقسیم شده‌اند:

- **معماری**: مفاهیم کلی، نیازهای امنیتی، تعاریف و مکانیسم‌هائی که تکنولوژی IPSec را تعریف می‌کنند، می‌پوشاند.
- **کپسولی کردن محمولهٔ امنیتی (ESP)**: فرمت بسته و مقوله‌های عمومی مرتبط با استفاده از ESP برای رمزنگاری بسته و اختیاراً اعتبارسنجی را توصیف می‌کند.
- **سرآیند اعتبارسنجی (AH)**: فرمت بسته و مقوله‌های عمومی مرتبط با استفاده از AH برای اعتبارسنجی بسته را توصیف می‌کند.
- **الگوریتم رمزنگاری**: مجموعه‌ای از اسناد که توصیف می‌نمایند چگونه الگوریتم‌های مختلف رمزنگاری برای ESP بکار می‌روند.
- **الگوریتم اعتبارسنجی**: مجموعه‌ای از اسناد که توصیف می‌نمایند چگونه الگوریتم‌های مختلف اعتبارسنجی برای AH و ESP بکار می‌روند.
- **مدیریت کلید**: اسنادی که روش‌های مدیریت کلید را توصیف می‌کنند.
- **محدودهٔ تعبیر (DOI)**: شامل اندازه‌های لازم برای سایر اسناد جهت مرتبط نمودن آنها به یکدیگر است. این شامل شناسه‌های رمزنگاری‌های معتبر و الگوریتم‌های اعتبارسنجی و همچنین پارامترهای اختیاری همچون طول عمر یک کلید است.



شکل ۲-۶ مروری بر اسناد IPSec

سرویس‌های IPSec

IPSec سرویس‌های امنیتی در سطح IP را بنحوی فراهم می‌آورد که سیستم قادر است تا پروتکل‌های امنیتی لازم را انتخاب کرده، الگوریتم(های) لازم برای سرویس(ها) را تعیین نموده و کلیدهای رمزنگاری لازم برای سرویس‌های درخواست شده را در محل مناسب قرار دهد. دو پروتکل برای ایجاد امنیت بکار می‌رود: یک پروتکل اعتبارسنجی که بتوسط سرآیند پروتکل یعنی Authentication Header (AH) شناسائی شده و یک پروتکل مخلوط رمزنگاری/ اعتبارسنجی که بتوسط فرمت بستهٔ آن پروتکل Encapsulating Security Payload (ESP) شناسائی می‌گردد. سرویس‌ها بقرار زیراند:

- کنترل دست‌یابی
- صحت دیتا در حالت غیراتصالی
- اعتبارسنجی مبدأ دیتا
- رد بسته‌های بازخوانی شده
- محرمانگی (رمزنگاری)
- محرمانگی محدود جریان ترافیک

جدول ۱-۶ نشان می‌دهد که کدام سرویس‌ها بتوسط پروتکل‌های AH و ESP ایجاد می‌شوند. برای ESP دو حالت وجود دارد: حضور و یا عدم حضور اعتبارسنجی بصورت اختیاری. AH و ESP هر دو محمل‌هائی برای کنترل دست‌یابی‌اندکه مبتنی بر توزیع کلیدهای رمزنگاری و مدیریت جریان‌های ترافیک مرتبط با این پروتکل‌های امنیتی می‌باشند.

اتحادهای امنیتی (Security Associations)

یک مفهوم کلیدی که در هردو مکانیسم اعتبارسنجی و محرمانگی IP ظاهر می‌شود، یک اتحاد امنیتی (SA) است. یک اتحاد یک رابطهٔ یک- طرفه بین یک فرستنده و یک گیرنده است که سرویس‌های امنیتی را برای ترافیک حمل شده روی آن فراهم می‌کند. اگر یک رابطهٔ نظیر نیز مورد نیاز باشد آنگاه برای مبادلهٔ امن دوطرفه، دو اتحاد امنیتی لازم است. سرویس‌های امنیتی فقط برای استفاده از AH یا ESP، و نه هردوی آنها، به یک SA داده می‌شود.

جدول ۱-۶ سرویس‌های IPSec

	AH	ESP (فقط رمزنگاری)	ESP (رمزنگاری بعلاوه اعتبارسنجی)
کنترل دست‌یابی	✓	✓	✓
صحت دیتا در حالت غیراتصالی	✓		✓
اعتبارسنجی منبع دیتا	✓		✓
رد بسته‌های بازخوانی شده	✓	✓	✓
محرمانگی دیتا		✓	✓
محرمانگی محدود جریان ترافیک		✓	

یک اتحاد امنیتی بطور یکتا با سه پارامتر تعیین می‌گردد:

- **شاخص پارامترهای امنیتی (SPI)**: دنباله‌ای از بیت‌ها که به این SA اختصاص داده شده و فقط اهمیت محلی دارد. SPI در سرآیندهای AH و ESP حمل شده تا سیستم گیرنده را قادر سازد تا یک SA که تحت آن یک بستهٔ دریافتی مورد پردازش قرار می‌گیرد را انتخاب کند.
- **آدرس IP مقصد**: در حال حاضر تنها آدرس‌های unicast مجاز است. این آدرس نقطهٔ انتهائی مقصد SA است که ممکن است یک سیستم انتهائی و یا یک سیستم شبکه مثل یک دیوار آتش و یا یک مسیریاب باشد.
- **شناسهٔ پروتکل امنیتی**: نمایشگر این است که آیا اتحاد، یک اتحاد AH و یا یک اتحاد ESP است.

بنابراین در هر بستهٔ IP، اتحاد امنیتی بطور یکتا بتوسط Destination Address در سرآیند IPv4 یا IPv6 و SPI در سرآیند الحاقی (AH یا ESP) مشخص می‌گردد.

پارامترهای SA

در هر پیاده‌سازی IPSec، یک پایگاه داده اتحاد امنیتی (Security Association Database) وجود دارد که پارامترهای مرتبط با هر SA را تعریف می‌کند. یک اتحاد امنیتی معمولاً با پارامترهای زیر تعریف می‌شود:

- **کنتر شمارهٔ ردیف**: یک اندازهٔ ۳۲- بیتی که برای تولید میدان Sequence Number سرآیندهای AH یا ESP بکار می‌رود و در بخش ۳-۶ تعریف شده است (برای همهٔ پیاده‌سازی‌ها لازم است).
- **سرریز کنتر شمارهٔ ردیف**: یک پرچم که نشان‌دهندهٔ این است که آیا سرریز کنتر شمارهٔ ردیف‌ها بایستی یک پیشامد قابل ممیزی را تولید کرده و از انتقال بیشتر بسته‌ها در این SA جلوگیری نماید (برای همهٔ پیاده‌سازی‌ها لازم است).
- **پنجرهٔ ضد- بازخوانی**: برای تعیین اینکه آیا یک بستهٔ AH یا ESP یک بازخوانی است یا نه، که در بخش ۳-۶ توضیح داده شده است (برای همهٔ پیاده‌سازی‌ها لازم است).
- **اطلاعات AH**: الگوریتم اعتبارسنجی، کلیدها، طول عمر کلیدها و پارامترهای مرتبطی که با AH بکار می‌رود (مورد نیاز پیاده‌سازی‌های AH).
- **اطلاعات ESP**: الگوریتم‌های رمزنگاری و اعتبارسنجی، کلیدها، مقادیر اولیه، طول عمر کلیدها و پارامترهای مرتبطی که با ESP بکار می‌رود (مورد نیاز پیاده‌سازی‌های ESP).
- **طول عمر این اتحاد امنیتی**: یک طول زمانی یا شمارش بایت که بعد از آن، این SA بایستی با یک SA جدید (و SPI جدید) تعویض شده و یا خاتمه یابد بعلاوهٔ نمایشگری برای اینکه نشان دهد کدامیک از این دو عمل بایستی واقع شود (برای همهٔ پیاده‌سازی‌ها لازم است).
- **مُود پروتکل IPSec**: Tunnel، transport و یا wildcard (برای همهٔ پیاده‌سازی‌ها لازم است). این مُودها بعداً در همین بخش توضیح داده می‌شوند.

- **MTU مسیر**: ماکزیمم واحد انتقال مشاهده شده در مسیر (اندازهٔ ماکزیمم بسته‌ای که می‌تواند بدون قطعه- قطعه شدن انتقال یابد) و متغیرهای نمایش طول عمر (برای همهٔ پیاده‌سازی‌ها لازم است).

مکانیسم مدیریت کلید که برای توزیع کلیدها بکار می‌رود با مکانیسم‌های اعتبارسنجی و محرمانگی تنها از طریق SPI مرتبط است. بنابراین اعتبارسنجی و محرمانگی مستقل از هرنوع مکانیسم مدیریت کلید خاصی تعریف شده‌اند.

انتخاب‌کننده‌های SA

IPSec انعطاف‌پذیری قابل ملاحظه‌ای را، در انتخاب اینکه کدام سرویس‌های IPSec به ترافیک IP اعمال شوند، برای کاربر ایجاد می‌کند. همانطور که بعداً خواهیم دید، SAها می‌توانند به روش‌های مختلف ترکیب شده و پیکربندی مناسب کاربر را ایجاد کنند. علاوه برآن IPSec درجهٔ بالائی از تشخیص برای تمایز بین ترافیکی که IPSec به آن اعمال شده با ترافیکی که می‌تواند IPSec را دور بزند ایجاد نموده که مورد اول ترافیک IP را به SAهای بخصوص پیوند می‌دهد.

ابزاری که ترافیک IP را به SAهای مشخص (یا نبود SA در مورد ترافیکی که می‌تواند IPSec را دور بزند) مرتبط می‌سازد، پایگاه دادهٔ خط‌مشی امنیتی Security Policy Database (SPD) است. در ساده‌ترین فرم خود، یک SPD شامل اقلامی است که هریک از آنها یک زیرمجموعه از ترافیک IP را تعیین کرده و به یک SA برای آن ترافیک اشاره می‌کند. در محیط‌های پیچیده‌تر، ممکن است اقلام متعددی وجود داشته باشند که بالقوه مرتبط با یک SA منفرد یا SAهای متعدد نظیر یک SPD منفرد باشند. خواننده در صورت نیاز بایستی به اسناد IPSec مراجعه نماید.

هر SPD بتوسط یک مجموعه از اندازهٔ میدان‌های پروتکل IP و لایهٔ بالاتر بنام *انتخاب‌کننده‌ها (selectors)* تعریف می‌شود. در واقع این انتخاب کننده‌ها، برای فیلترکردن ترافیک خروجی بمنظور نگاشت آنها به یک SA بخصوص استفاده می‌شوند. پردازش داده‌های خارج شونده، از مراحل عمومی زیر برای هر بستهٔ IP تبعیت می‌کند:

۱- اندازهٔ میدان‌های مرتبط در بسته (میدان‌های selector) را با SPD مقایسه کرده تا یک تطبیق پیدا شود که به هیچ و یا چند SA اشاره نماید.

۲- اگر SA برای این بسته موجود است آن را تعیین کرده و SPI مرتبط با آن را استخراج کند.

۳- پردازش لازم IPSec را انجام دهد (یعنی پردازش ESP یا AH).

انتخاب‌کننده‌های زیر تعیین‌کنندهٔ یک قلم موجود در SPD هستند:

- **آدرس IP مقصد**: این می‌تواند یک آدرس IP منفرد، محدوده‌ای از آدرس‌ها و یا یک آدرس عام (mask) باشد. دوتای آخر از این جهت مورد نیازند که بیش از یک سیستم مقصد با SA یکسان را حمایت کنند (مثل پشت یک دیوار آتش).

- **آدرس IP منبع**: این می‌تواند یک آدرس IP منفرد، محدوده‌ای از آدرس‌ها و یا یک آدرس عام باشد. دوتای آخر از این جهت مورد نیازند که بیش از یک سیستم منبع با SA یکسان را حمایت کنند (مثل پشت یک دیوار آتش).
- **شمارهٔ شناسائی کاربر**: یک شناسهٔ کاربر(UserID) در سیستم عامل. این یک میدان در سرآیندIP و یا سرآیندهای لایهٔ بالاتر نیست بلکه فقط در صورتی وجود دارد که IPSec روی همان سیستم عامل که کاربر به آن وصل است کار کند.
- **سطح امنیتی دیتا**: برای سیستم‌هائی که امنیت جریان اطلاعات را فراهم می‌کنند بکار می‌رود (مثلاً سرّی یا طبقه‌بندی نشده).
- **پروتکل لایهٔ حمل‌ونقل**: از پروتکل IPv4 و یا میدان IPv6 Next Header بدست می‌آید. این ممکن است شمارهٔ یک پروتکل منفرد، لیستی از شمارهٔ پروتکل‌ها و یا محدوده‌ای از شمارهٔ پروتکل‌ها باشد.
- **پورت‌های منبع و مقصد**: اینها ممکن است اندازهٔ یک پورت منفرد TCP یا UDP، لیستی از پورت‌های مختلف و یا یک پورت عام باشند.

مُودهای حمل‌ونقل و تونل

هم AH و هم ESP دو مُود استفاده دارند: مُود حمل‌ونقل (transport) و مُود تونل(tunnel). عملکرد این دو مُود به بهترین نحو پس از توصیف AH و ESP روشن خواهد شد که در بخش‌های ۳-۶ و ۴-۶ به آن خواهیم پرداخت. فعلاً مروری کوتاه بر آنها ارائه می‌دهیم.

مُود حمل‌ونقل

مُود حمل‌ونقل حفاظت را عمدتاً برای پروتکل‌های لایهٔ بالاتر فراهم می‌آورد. یعنی حفاظت مُود حمل‌ونقل به محمولهٔ بستهٔ IP اعمال می‌شود. مثال‌هائی از این دست یک سِگمنت TCP یا UDP و یا یک بستهٔ ICMP است که همهٔ آنها مستقیماً در بالای IP کار می‌کنند. معمولاً مُود حمل‌ونقل برای ارتباطات سر- به- سر بین دو میزبان بکار می‌رود (مثلاً یک کلاینت و یک سِرور و یا دو ایستگاه کاری). وقتی یک میزبان، AH یا ESP را روی IPv4 اجرا می‌کند، محموله همان دیتائی است که بطور نرمال بعد از سرآیند IP قرار می‌گیرد. برای IPv6، محموله دیتائی است که معمولاً بعد از سرآیند IP و هر سرآیند الحاقی موجود دیگر قرار دارد بجز حالت استثنائی سرآیند option که ممکن است در بخش حفاظت شده قرارگیرد.

ESP در مُود حمل‌ونقل، محمولهٔ IP و نه سرآیند IP را، رمزنگاری و بطور اختیاری اعتبارسنجی می‌نماید. AH در مُود حمل‌ونقل محمولهٔ IP و بخش‌های انتخاب شده‌ای از سرآیند IP را رمزنگاری می‌کند.

مُود تونل

مُود تونل حفاظت را برای تمام بستهٔ IP ایجاد می‌کند. برای این منظور پس از اینکه میدان‌های AH یا ESP به بستهٔ IP اضافه شدند، تمام بسته باضافهٔ میدان‌های امنیتی بصورت محمولهٔ یک بستهٔ IP جدید «بیرونی‌تر» با سرآیند IP جدید

درخواهند آمد. تمام بستهٔ اولیه یا درونی از درون یک «تونل» از یک نقطه شبکهٔ IP به نقطهٔ دیگر حرکت کرده و هیچ مسیریابی در مسیر آن قادر نیست سرآیند IP درونی را بررسی کند. چون بستهٔ اولیه کپسولی شده است، بستهٔ جدیدتر و بزرگتر ممکن است دارای آدرس‌های مبدأ و مقصد کاملاً متفاوت باشند که این خود به امنیت می‌افزاید. مُود تونل وقتی استفاده می‌شود که یک یا هردو انتهای SA یک دروازهٔ امنیتی همچون دیوار آتش یا مسیریابی باشد که IPSec را بکار می‌گیرد. در مُود تونل، تعدادی از میزبانان روی شبکه و پشت دیوار آتش می‌توانند بدون پیاده‌سازی IPSec ، ارتباطات امن داشته باشند. بسته‌های حفاظت نشده که از طرف چنین میزبان‌هائی تولید می‌شوند از درون شبکه‌های خارجی بتوسط SAهای مُود تونل که بتوسط نرم‌افزار IPSec در دیوار آتش و یا مسیریاب‌های امن در مرزهای شبکه فراهم گشته‌اند، تونل می‌شوند.

در اینجا مثالی از اینکه مُود تونل IPSec چطور کار می‌کند، ارائه می‌دهیم. میزبان A روی یک شبکه، یک بستهٔ IP با آدرس مقصد میزبان B روی شبکهٔ دیگری را تولید می‌کند. این بسته از میزبان مبدأ به یک دیوار آتش یا مسیریاب امن در مرز شبکهٔ A می‌رود. دیوار آتش تمام بسته‌های خروجی را فیلتر کرده تا نیاز به پردازش IPSec را تعیین کند. اگر این بسته از A به B نیاز به IPSec داشته باشد، دیوار آتش پردازش IPSec را انجام داده و بسته را با یک سرآیند IP بیرونی کپسولی می‌نماید. آدرس IP منبع این بسته IP بیرونی، این دیوار آتش بوده و آدرس مقصد ممکن است دیوار آتشی باشد که مرز شبکهٔ محلی B را تشکیل می‌دهد. حالا این بسته به سمت دیوار آتش B مسیریابی می‌شود و مسیریاب‌های وسط راه فقط سرآیند IP بیرونی را وارسی می‌کنند. در دیوار آتش B، سرآیند IP بیرونی کنده می‌شود و بسته درونی به B تحویل می‌گردد.

ESP در مُود تونل، تمام بستهٔ IP درونی که شامل سرآیند IP درونی نیز می‌شود را رمزنگاری و بطور اختیاری اعتبارسنجی می‌نماید. AH در مُود تونل تمام بستهٔ IP درونی و بخش‌های انتخاب‌شده‌ای از سرآیند IP بیرونی را اعتبارسنجی می‌نماید.

جدول ۲-۶ عملکرد مُودهای حمل‌ونقل و تونل را خلاصه کرده است.

۶-۳ سرآیند اعتبارسنجی (Authentication Header)

سرآیند اعتبارسنجی وظیفهٔ اطمینان از صحت دیتا و اعتبارسنجی بسته‌های IP را بعهده دارد. خاصیت مربوط به صحت دیتا این اطمینان را ایجاد می‌کند که دخل تصرف تشخیص داده نشده در محتویات بسته‌های در حال ترانزیت غیرممکن است. خاصیت اعتبارسنجی، یک سیستم یا یک دستگاه متصل به شبکه را قادر می‌سازد تا هویت یک کاربر و یا یک کاربرد را بررسی کرده و ترافیک را بر اساس آن فیلتر کند. اعتبارسنجی همچنین از حملات تقلیدآدرس(spoofing) که امروزه در اینترنت مشاهده می‌شود جلوگیری می‌کند. AH همچنین در برابر حملات بازخوانی (replay) ایجاد مصونیت می‌نماید.

اعتبارسنجی با استفاده از کُد اعتبارسنجی پیام (MAC) صورت می‌پذیرد که همانطور که قبلاً در مورد آن بحث شده است نیاز به اشتراک گذاردن یک کلید سّری بین طرفین ارتباط دارد.

سرآیند اعتبارسنجی شامل میدان‌های زیر است (شکل ۳-۶) :

- **Next Header (8 bits)**: نوع سرآیندی که بلافاصله پس از این سرآیند قرار می‌گیرد را مشخص می‌کند.

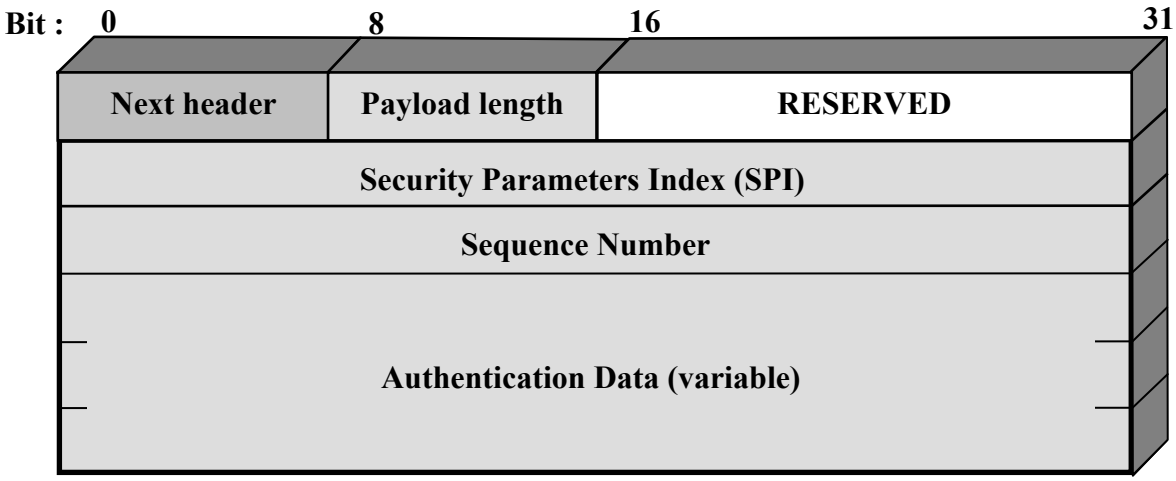
جدول ۲-۶ عملکرد مُود حمل‌ونقل و مُود تونل

	Transport Mode SA	Tunnel Mode SA
AH	محمولهٔ IP و بخش‌های انتخاب شده‌ای از سرآیند IP و سرآیندهای الحاقی IPv6 را اعتبارسنجی می‌نماید.	تمام بستهٔ IP درونی (سرآیند درونی بعلاوهٔ محمولهٔ IP) باضافهٔ قسمت‌های انتخاب‌شده‌ای از سرآیند IP بیرونی و سرآیندهای الحاقی IPv6 بیرونی را اعتبارسنجی می‌نماید.
ESP	محمولهٔ IP و هر سرآیند الحاقی IPv6 که بعد از سرآیند ESP قرار دارد را رمزنگاری می‌کند.	بستهٔ IP درونی را رمزنگاری می‌کند.
ESP با اعتبارسنجی	محمولهٔ IP و هر سرآیند الحاقی IPv6 که بعد از سرآیند ESP قرار دارد را رمزنگاری می‌کند. محمولهٔ IP و نه سرآیند IP را اعتبارسنجی می‌کند.	بستهٔ IP درونی را رمزنگاری می‌کند. بستهٔ IP درونی را اعتبارسنجی می‌کند.

- **Payload Length (8 bits)**: طول سرآیند اعتبارسنجی برحسب کلمات ۳۲- بیتی منهای ۲. بعنوان مثال، طول پیش فرض میدان اعتبارسنجی دیتا ۹۶ بیت و یا ۳ کلمهٔ ۳۲- بیتی است. با یک سرآیند ثابت سه کلمه‌ای، شش کلمه در سرآیند وجود خواهد داشت و اندازهٔ میدان Payload Length برابر ۴ خواهد بود.
- **Reserved (16 bits)**: برای مصارف آینده رزرو شده است.
- **Security Parameters Index (32 bits)**: یک اتحاد امنیتی را مشخص می‌کند.
- **Sequence Number (32 bits)**: یک شمارنده که اندازهٔ آن بطور یکنواخت زیاد می‌شود و بعداً مورد بحث قرار خواهد گرفت.
- **Authentication Data (variable)**: یک میدان با طول متغیر (که بایستی مضرب صحیحی از کلمات ۳۲- بیتی باشد) که شامل Integrity Check Value (ICV) ،یا MAC، برای این بسته است و بعداً در مورد آن صحبت خواهیم کرد.

سرویس ضد- بازخوانی (Anti-Replay Sevice)

یک حملهٔ بازخوانی چنین است که در آن حمله‌کننده یک نسخه از بستهٔ اعتبارسنجی شده را به دست آورده و بعداً آن را برای مقصد مورد نظر ارسال می‌دارد. دریافت مجدد بسته‌های اعتبارسنجی شده در مقصد، ممکن است سرویس را بنحوی مختل کرده و یا نتایج نامطلوب دیگری به دنبال داشته باشد. میدان Sequence Number برای مقابله با چنین حملاتی طراحی شده است. ابتدا نحوهٔ تولید شماره ردیف بتوسط فرستنده را مورد بحث قرار داده و سپس خواهیم دید که این میدان چگونه بتوسط گیرنده مورد پردازش قرار می‌گیرد.



شکل ۳-۶ سرآیند اعتبارسنجی (AH) در IPSec

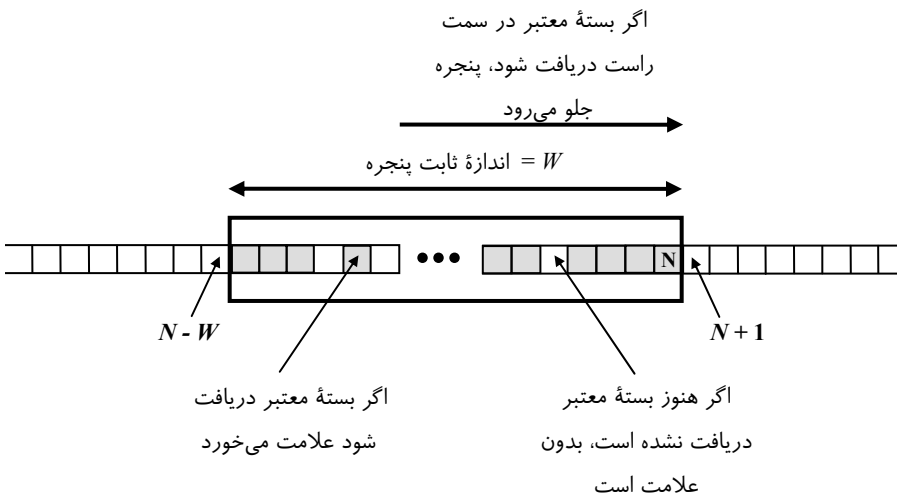
زمانی که یک SA جدید برپا می‌شود، **فرستنده** کنتر شمارهٔ ردیف را روی 0 تنظیم می‌کند. هربار که یک بسته روی این SA ارسال می‌گردد، فرستنده کنتر را یک واحد افزایش داده و اندازهٔ آن را در میدان Sequence Number قرار می‌دهد. بنابراین اولین اندازه‌ای که استفاده می‌شود 1 است. اگر سرویس ضد- بازخوانی فعال باشد (پیش فرض)، فرستنده نبایستی اجازه دهد تا شمارهٔ ردیف پس از عبور از $۲^{۳۲}-۱$ به صفر برگردد، در غیر اینصورت بسته‌های متعددی با شماره ردیف یکسان وجود خواهند داشت. اگر مرز $۲^{۳۲}-۱$ فرا رسد، فرستنده بایستی این SA را خاتمه داده و SA جدیدی با یک کلید جدید را با گیرنده تشکیل دهد.

چون IP یک سرویس غیراتصالی و غیرقابل اعتماد است، پروتکل تضمینی در برابر تحویل منظم بسته‌ها و همچنین تحویل تمام بسته‌ها ندارد. بنابراین اسناد اعتبارسنجی IPSec چنین دیکته می‌کند که **گیرنده** بایستی پنجره‌ای با اندازهٔ W را ایجاد نماید که پیش فرض آن $W = ۶۴$ است. لبهٔ سمت راست پنجره، بالاترین شماره ردیف N، مربوط به آخرین بستهٔ معتبر دریافت شده، را نشان می‌دهد. برای هر بسته‌ای با شمارهٔ ردیفی در محدودهٔ N-W+1 تا N که بطور صحیح دریافت شده است (یعنی اعتبار آن سنجیده شده است)، شیار نظیر آن در پنجره علامت می‌خورد (شکل ۴-۶). پس از دریافت یک بسته، یک پردازش بشکل زیر روی آن انجام می‌شود:

۱- اگر بستهٔ دریافت شده در داخل پنجره قرار داشته و جدید باشد، اندازهٔ MAC کنترل می‌شود. اگر بسته معتبر باشد، شیار نظیر آن در پنجره علامت‌گذاری می‌شود.

۲- اگر بستهٔ دریافت شده در سمت راست پنجره قرار داشته و جدید باشد، اندازهٔ MAC کنترل می‌شود. اگر بسته معتبر باشد، پنجره جلو می‌رود بنحوی که این شماره ردیف لبهٔ سمت راست پنجره را تشکیل دهد و شیار نظیرآن در پنجره علامت‌گذاری می‌شود.

۳- اگر بستهٔ دریافت شده در سمت چپ پنجره واقع باشد و یا اعتبارسنجی با شکست مواجه شود، بسته نابود شده و این یک پیشامد قابل ثبت و ممیزی است.



شکل ۴-۶ مکانیسم ضد- بازخوانی

اندازهٔ کنترل صحت (Integrity Check Value)

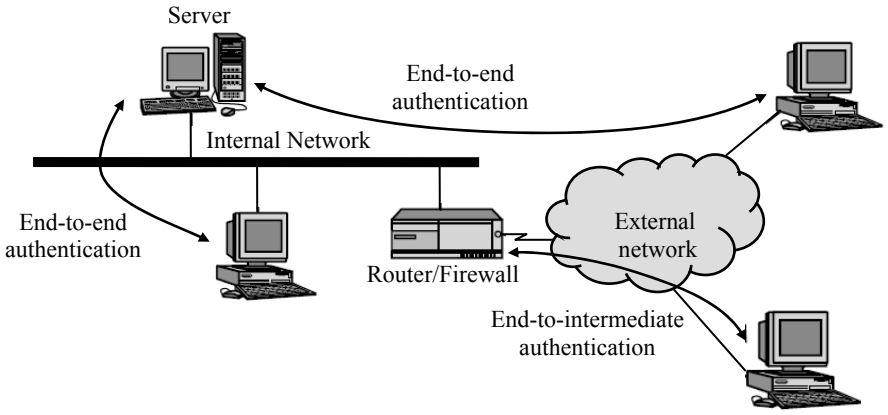
میدان Authentication Data اندازه‌ای را نگاه می‌دارد که به آن اندازهٔ کنترل صحت Integrity Check Value (ICV) گویند. ICV یک کُد اعتبارسنجی پیام و یا فرم مقطّعی از این کُد است که بتوسط الگوریتم MAC تولید می‌شود. مشخصه‌های فعلی، پیاده‌سازی و حمایت از دو الگوریتم زیر را دیکته می‌کنند:

- HMAC-MD5-96
- HMAC-SHA-1-96

هردوی اینها از الگوریتم HMAC استفاده می‌کنند که اولی کُد درهم‌ساز MD5 و دومی کُد درهم‌ساز SHA-1 را بکار می‌برد (تمام این الگوریتم‌ها در فصل ۳ مورد بحث قرار گرفته‌اند). در هردو حالت، اندازهٔ کامل HMAC محاسبه شده ولی بعداً بریده شده و تنها ۹۶ بیت اول آن مورد استفاده قرار می‌گیرد که طول پیش فرض میدان Authentication Data است.

MAC روی اقلام زیر محاسبه می‌شود:

- میدان‌های سرآیند IP که یا در هنگام ترانزیت تغییر نکرده‌اند(immutable) و یا اندازهٔ آنها در هنگام ورود به نقطهٔ انتهائی برای AH SA قابل پیش‌بینی است. میدان‌هائی که ممکن است در حال ترانزیت تغییر کرده و یا اندازهٔ آنها در هنگام ورود غیرقابل پیش‌بینی است بمنظور محاسبه در مبدأ و مقصد صفر منظور می‌شوند.
- سرآیند AH به غیر از میدان Authentication Data. میدان Authentication Data برای محاسبه در مبدأ و مقصد صفر منظور می‌شود.
- تمام دیتای پروتکل لایهٔ بالاتر که در هنگام ترانزیت تغییرناپذیر فرض شده‌اند(مثلاً یک سِگمنت TCP و یا یک بستهٔ IP درونی در مُود تونل).



شکل ۵-۶ اعتبارسنجی End-to-End در برابر اعتبارسنجی End-to-Intermediate

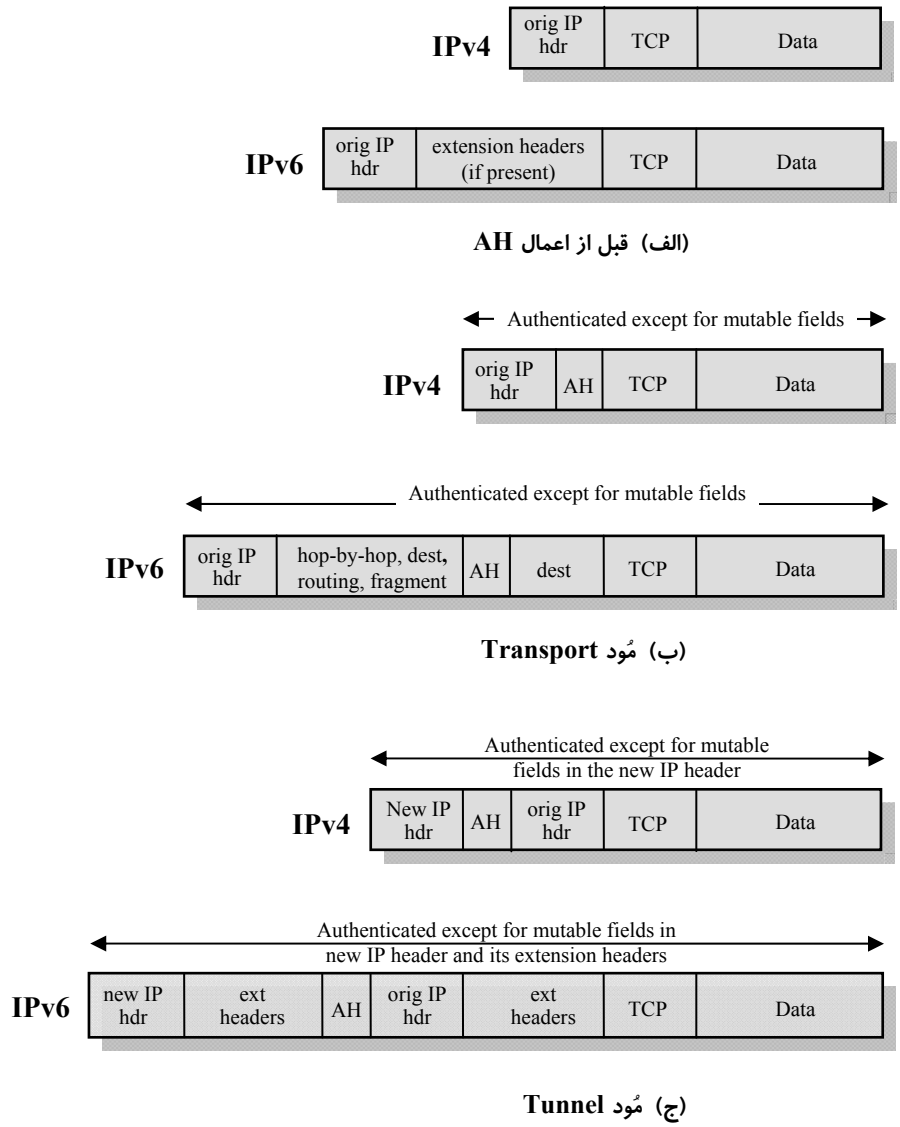
برای IPv4، مثال‌هائی از میدان‌های تغییرناپذیر Internet Header Length و Source Address هستند. مثـالی از یک میدان تغییرپذیر ولی قابل پیش‌بینی Destination Address است (با مسیریابی منبع). مثال‌هائی از میدان‌های تغییرپذیر که قبل از محاسبات ICV صفر هستند، میدان‌های Time to Live و Header Checksum هستند. توجه کنید که هـر دو میدان آدرس منبع و مقصد حفاظت شده‌اند بطوری که از جعل آدرس جلوگیری می‌شود.

برای IPv6، مثال‌هائی در سرآیند اصـلی، Version (تغییرناپـذیر)، Destination Address (تغییرپـذیر ولـی قابـل پیش‌بینی) و Flow Label (تغییرپذیر و برای محاسبات برابر صفر) می‌باشند.

مُودهای حمل‌ونقل و تونل

شکل ۵-۶ دو حالت که در آنها سرویس اعتبارسنجی IPSec می‌تواند مورد استفاده قرار گیـرد را نـشان مـی‌دهـد. در یـک حالت، اعتبارسنجی مستقیماً بین یک سِرور و ایستگاه کاری کلاینت انجام می‌شود که ایستگاه کاری می‌تواند روی همـان شـبکهٔ سِرور یا روی یک شبکهٔ خارجی باشد. تا زمانی که ایستگاه کاری و سِرور یک کلید سّری حفاظـت شـده را در اشـتراک دارنـد، عمل اعتبارسنجی امن است. این مورد از یک SA در مُود حمل‌ونقل استفاده می‌کند. در حالت دیگر، یـک ایـستگاه کـاری دور هویت خود را برای دیوار آتش همان سیستم مشخص می‌نماید، یا برای اینکه به تمام شبکهٔ داخلی دسترسـی یابـد و یـا بـدلیل اینکه سِرور درخواست شده اعتبارسنجی را حمایت نمی‌کند. این مورد از یک SA در مُود تونل استفاده می‌کند.

در این قسمت به قلمرو اعتبارسنجی ایجادشده بتوسط AH و محل قرارگرفتن سرآیند اعتبارسنجی بـرای ایـن دو مُـود نگاهی می‌اندازیم. ملاحظات برای IPv4 و IPv6 قدری متفاوت‌اند. شکل ۶-۶الف بسته‌های معمـول IPv4 و IPv6 را نـشان می‌دهد. در این شکل محمولهٔ IP یک سگمنت TCP است. این محموله می‌تواند یک واحد دیتا برای هر پروتکل دیگری مانند UDP و یا ICMP باشد که مستقیماً از IP استفاده می‌کند.



شکل ۶-۶ افق دید اعتبارسنجی AH

برای **AH مُود حمل‌ونقل** که از IPv4 استفاده می‌کند، AH بعد از سرآیند معمول IP و قبل از محمولهٔ IP (مثلاً یک سگمنتTCP)، قرار می‌گیرد که این امر در بخش بالائی شکل ۶-۶ب نشان داده شده است. اعتبارسنجی، تمام بسته بجز میدان‌های تغییرپذیر در سرآیند IPv4 که برای محاسبات MAC مساوی صفر قرار داده می‌شوند، را در بر می‌گیرد.

در مقولهٔ IPv6، AH بعنوان یک محمولهٔ سر- به- سر تلقی می‌گردد، یعنی نه بتوسط مسیریاب‌های میانی مورد بازبینی قرار گرفته و نه پردازشی روی آن صورت می‌پذیرد. بنابراین AH بعد از سرآیند اصلی IPv6 و سرآیندهای الحاقی hop-by-hop ، routing و fragment قرار می‌گیرد. سرآیندهای الحاقی اختیاری مربوط به مقصد می‌توانند قبل و یا بعد از سرآیند AH قرار گیرند که بستگی به منطق مورد استفاده دارد. بازهم اعتبارسنجی تمام بسته، بجز میدان‌های تغییرپذیر که برای محاسبات MAC برابر صفر قرار می‌گیرند، را پوشش می‌دهد.

برای **AH مُود تونل**، تمام بستهٔ IP اولیه اعتبارسنجی می‌شود و AH بین سرآیند اولیهٔ IP و سرآیند جدید بیرونی IP (شکل ۶-۶ج) وارد می‌گردد. سرآیند IP درونی، آدرس‌های مبدأ و مقصد نهائی را مشخص می‌کند در حالی که یک سرآیند IP بیرونی می‌تواند شامل آدرس‌های IP متفاوت باشد(مثلاً آدرس یک دیوار آتش و یا دروازه‌های امنیتی دیگر).

در مُود تونل تمام بستهٔ IP درونی، که شامل کل سرآیند IP درونی نیز هست، بتوسط AH محافظت می‌شود. سرآیند IP بیرونی (و در مورد IPv6 سرآیندهای الحاقی IP بیرونی)، بجز میدان‌های تغییرپذیر و غیرقابل پیش‌بینی، نیز در محدودهٔ حفاظتی قرار دارند.

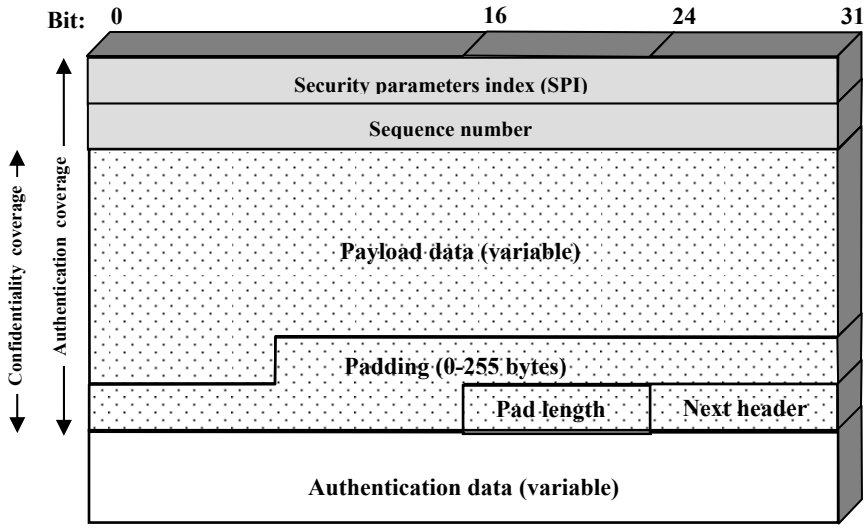
۶-۴ کپسولی‌کردن محمولهٔ امنیتی (Encapsulating Security Payload)

کپسولی‌کردن محمولهٔ امنیتی (ESP)، یک سرویس محرمانگی را فراهم می‌آورد که شامل محرمانگی محتویات پیام و محرمانگی محدود جریان ترافیک است. بصورت اختیاری، ESP می‌تواند یک سرویس اعتبارسنجی را نیز فراهم آورد.

فرمت ESP

شکل ۶-۷ فرمت بستهٔ ESP را نشان می‌دهد. این بسته شامل میدان‌های زیر است:

- **Security Parameters Index (32 bits)**: یک اتحاد امنیتی را مشخص می‌کند.
- **Sequence Number (32 bits)**: اندازهٔ یک شمارنده است که بطور یکنواخت اضافه می‌شود. این امر برای محافظت در برابر حملات بازخوانی، همانطور که در AH بحث شد، مورد استفاده قرار می‌گیرد.
- **Payload Data (variable)**: این یک سِگمنت سطح حمل‌ونقل و یا یک بستهٔ IP مُود تونل است که با رمزنگاری محافظت گردیده است.



شکل ۶-۷ فرمت ESP در IPSec

- **Padding (0-255 bytes)**: هدف این میدان بعداً تعریف خواهد شد.
- **Pad Length (8 bits)**: نمایش‌دهندهٔ تعداد بایت‌های لایی است که درست قبل از این میدان قرار دارند.
- **Next Header (8 bits)**: مشخص‌کنندهٔ نوع داده‌های موجود در میدان Payload Data است که بتوسط اولین سرآیندآن محموله تعیین می‌گردد (بعنوان مثال یک سرآیند الحاقی در IPv6 و یا یک پروتکل لایهٔ بالاتر شبیه TCP).
- **Authentication Data (variable)**: یک میدان با طول متغیر (بایستی مضربی از کلمات ۳۲- بیتی باشد) که شامل Integrity Check Value است که روی بستهٔ ESP منهای میدان Authentication Data محاسبه شده است.

الگوریتم‌های رمزنگاری و اعتبارسنجی

میدان‌های Payload Data، Padding، Pad Length و Next Header بتوسط سرویس ESP رمزنگاری می‌شوند. اگر الگوریتم بکار گرفته شده برای رمزنگاری محموله نیاز به داده‌هائی برای همزمانی رمزنگاری، نظیر بردار شروع (IV) داشته باشد آنگاه این داده‌ها ممکن است بطور صریح در شروع میدان Payload Data حمل شوند. اگر IV داشته باشیم، معمولاً رمزنگاری نمی‌شود، اگرچه اغلب به آن بعنوان بخشی از متن رمزشده نگاه می‌شود.

مشخصه‌های جاری چنین دیکته می‌کنند که یک پیاده‌سازی سازگار بایستی DES در مُود CBC را حمایت کند. تعدادی از الگوریتم‌های دیگر نیز در اسناد DOI دارای شناسه‌های معین بوده و بنابراین می‌توانند برای رمزنگاری مورد استفاده قرار گیرند. اینها شامل اقلام زیراند

- Three-key triple DES
- RC5
- IDEA
- Three-key triple IDEA
- CAST
- Blowfish

همانند AH، ESP استفاده از یک MAC با طول پیش فرض ۹۶ بیت را حمایت می‌نماید. همچنین همانند AH، مشخصهٔ جاری دیکته می‌کند که پیاده‌سازی سازگار بایستی HMAC-MD5-96 و HMAC-SHA-1-96 را حمایت نماید.

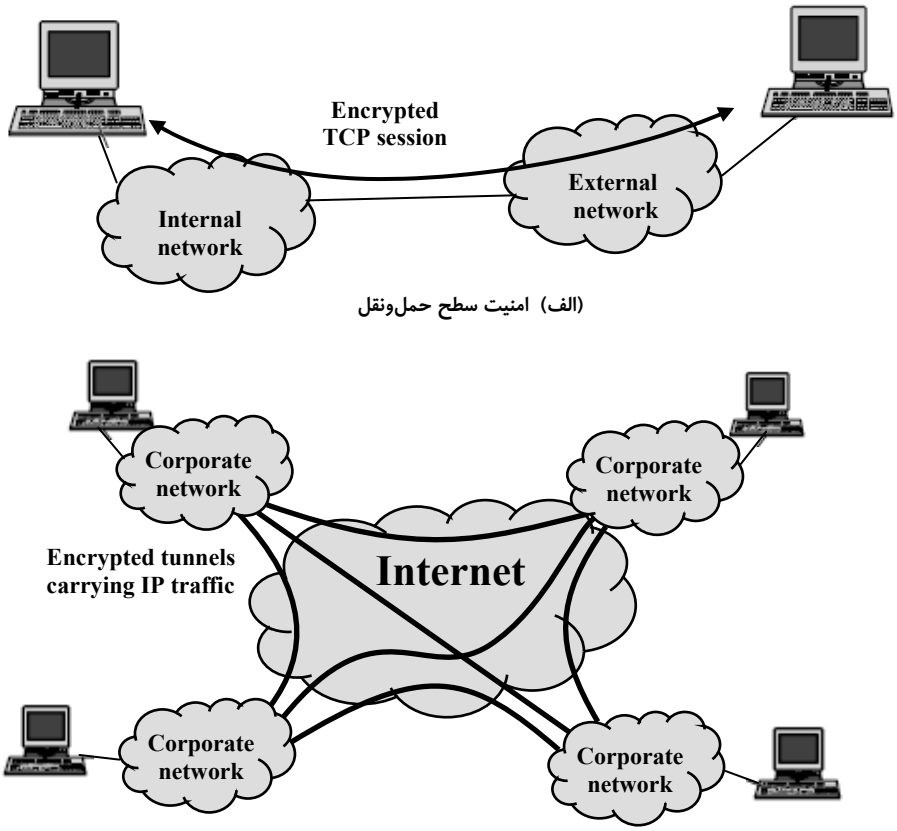
لائی (Padding)

میدان Padding دارای چند هدف است:

- اگر در یک الگوریتم رمزنگاری لازم باشد که متن ساده مضرب صحیحی از تعدادی بایت باشد (مثلاً مضربی از طول یک بلوک در رمز قالبی)، از میدان Padding برای توسعهٔ متن ساده (شامل میدان‌های Payload Data، Padding، Pad Length و Next Header) به طول مطلوب استفاده می‌شود.
- فرمت ESP نیاز دارد تا میدان‌های Pad Length و Next Header در سمت راست یک کلمهٔ ۳۲- بیتی قرار گیرند. بهمین ترتیب متن رمزشده بایستی مضربی از ۳۲ بیت باشد. میدان Padding برای اطمینان از این امر بکار می‌رود.
- Padding اضافه‌تری ممکن است برای ایجاد محرمانگی جریان ترافیک بکار رود تا طول واقعی محموله را پنهان سازد.

مُودهای حمل‌ونقل و تونل

شکل ۸-۶ دو روش که در آنها سرویس IPSec ESP را می‌توان بکار برد، نشان می‌دهد. در قسمت بالای شکل، رمزنگاری (و بطور اختیاری اعتبارسنجی) بین دو میزبان که مستقیماً بهم وصل‌اند فراهم شده است. شکل ۸-۶ ب نشان می‌دهد که چگونه



شکل ۸-۶ رمزنگاری مُود حمل‌ونقل در برابر رمزنگاری مُود تونل

عملیات مُود تونل می‌تواند برای برقراری یک شبکهٔ خصوصی مجازی (VPN) بکار رود. در این مثال، یک سازمان دارای چهار شبکهٔ خصوصی است که در عرض اینترنت بهم متصل‌اند. میزبان‌های روی شبکه‌های داخلی از اینترنت برای انتقال داده‌ها استفاده کرده ولی با سایر میزبان‌های روی اینترنت تعاملی ندارند. با خاتمه دادن به تونل‌ها در دروازه‌های امنیتی هر شبکهٔ داخلی، پیکربندی به میزبانان اجازه می‌دهد که از پیاده‌سازی قابلیت‌های امنیتی اجتناب نمایند. تکنیک قبلی از مُود حمل‌ونقل SA و تکنیک اخیر از مُود تونل SA استفاده می‌کند.

در این قسمت به افق دید ESP برای دو مُود توجه می‌کنیم. ملاحظات برای IPv4 و IPv6 قدری متفاوت‌اند. همانند بحث مربوط به افق دید AH، فرمت بسته‌ها در شکل ۶-۶الف را بعنوان نقطهٔ شروع بکار می‌بریم.

ESP مُود حمل‌ونقل

ESP مُود حمل‌ونقل برای رمزنگاری و اختیاراً اعتبارسنجی داده‌هائی که بتوسط IP حمل می‌شوند (مثلاً یک سِگمنت TCP)، همانند شکل ۹-۶الف، بکار می‌رود. برای این مُود و با استفاده از IPv4، سرآیند ESP در داخل بستهٔ IP درست قبل از سرآیند لایهٔ حمل‌ونقل (مثل TCP، UDP، ICMP) قرار گرفته و تەآیند ESP (میدان‌های Padding، Pad Length و Next Header) بعد از بستهٔ IP قرار می‌گیرند. اگر اعتبارسنجی نیز مورد انتخاب قرار گیرد،میدان ESP Authentication Data نیز پس از تەآیند ESP خواهد آمد. تمام سِگمنت سطح حمل‌ونقل بعلاوهٔ تەآیند ESP رمزنگاری می‌شود. اعتبارسنجی، تمام متن رمزشده بعلاوهٔ سرآیند ESP را شامل می‌شود.

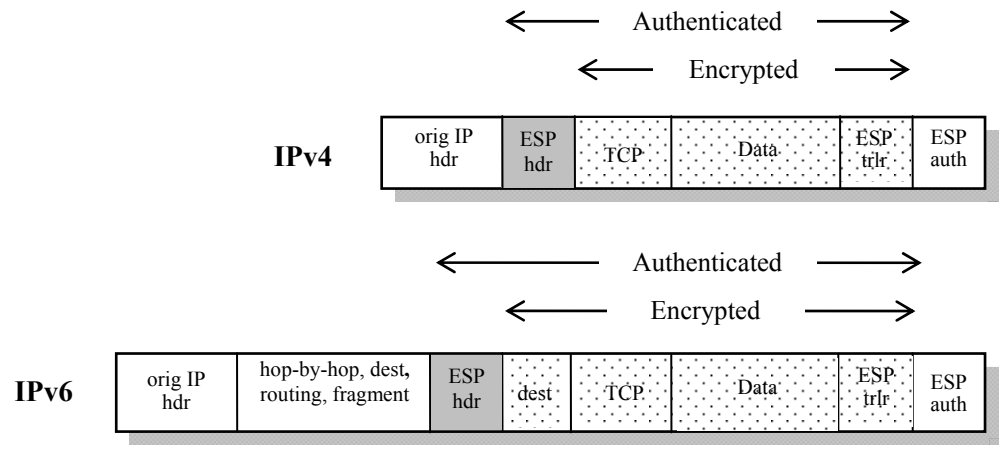
در مقولهٔ IPv6 به ESP بصورت یک محمولهٔ سر- به- سر نگاه می‌شود، یعنی بتوسط مسیریاب‌های میانی مورد پردازش قرار نمی‌گیرد. بنابراین سرآیند ESP بعد از سرآیند اصلی IPv6 base header و سرآیندهای hop-to-hop، routing و fragment extension قرار می‌گیرد. سرآیند الحاقی destination options می‌تواند قبل و یا بعداز سرآیند ESP قرار گیرد که بستگی به منطق مورد استفاده خواهد داشت. برای IPv6، رمزنگاری تمام سِگمنت لایهٔ حمل‌ونقل بعلاوهٔ تەآیند ESP و سرآیند الحاقی مقصد، اگر پس از سرآیند ESP قرار گیرد، را می‌پوشاند. بازهم اعتبارسنجی، متن رمزشده بعلاوهٔ سرآیند ESP را پوشش می‌دهد.

عملیات مُود حمل‌ونقل را می‌توان بصورت زیر خلاصه نمود:

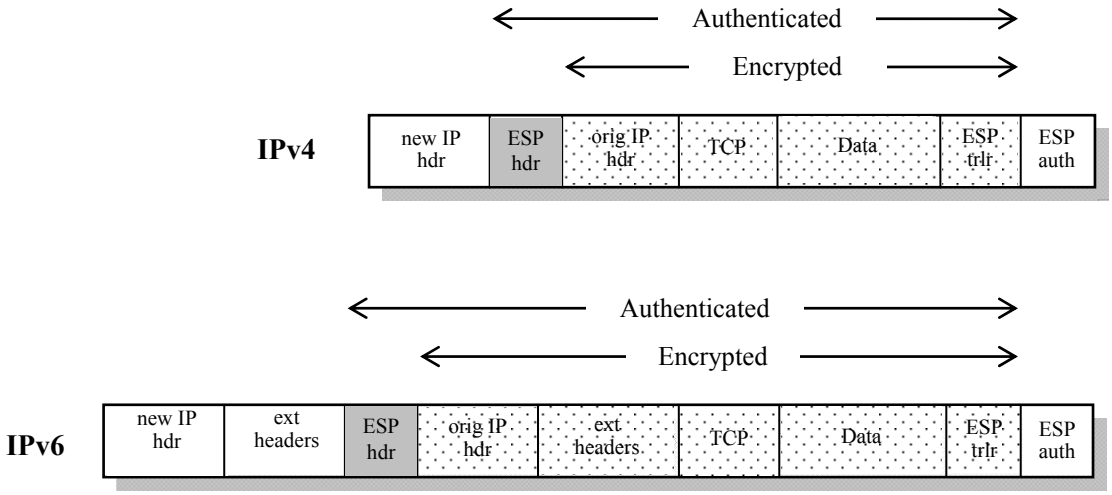
۱- در مبدأ، بلوک دیتا که شامل تەآیند ESP بعلاوهٔ تمام سِگمنت لایهٔ حمل‌ونقل است رمزنگاری شده و متن سادهٔ این بلوک با متن رمزنگاری آن تعویض شده تا انتقال یابد. اعتبارسنجی در صورت انتخاب به آن اضافه می‌گردد.

۲- بستهٔ دیتا آنگاه به سمت مقصد مسیریابی می‌گردد. هر مسیریاب میانی لازم است تا سرآیند IP بعلاوهٔ سرآیندهای الحاقی IP بصورت متن ساده را بررسی و پردازش نماید، ولی نیازی نیست تا متن رمزشده را وارسی کند.

۳- گرهٔ مقصد، سرآیند IP باضافهٔ سرآیندهای الحاقی IP را بصورت متن ساده بررسی می‌کند. آنگاه بر اساس SPI در سرآیند ESP، بقیهٔ بسته را برای دست‌یابی به سِگمنت لایهٔ حمل‌ونقل رمزگشائی می‌نماید.



(الف) مُود Transport



(ب) مُود Tunnel

شکل ۹-۶ افق دید رمزنگاری و اعتبارسنجی ESP

عملیات مُود حمل‌ونقل، برای هر کاربردی که آن را بکار می‌برد محرمانگی را فراهم می‌سازد و بنابراین از ایجاد محرمانگی در تک‌تک کاربردها اجتناب خواهد شد. این مُود عملیات بصورت معقولی بهره‌ور بوده زیرا مقدار نسبتاً کمی به طول بسته IP اضافه می‌نماید. یکی از نقاط ضعف این مُود این است که می‌توان روی بسته‌های انتقال یافته، تحلیل ترافیک انجام داد.

مُود تونل ESP

از ESP در مُود تونل برای رمزنگاری کل بستهٔ IP استفاده می‌شود (شکل ۹-۶ب). برای این مُود، سرآیند ESP در ابتدای بسته قرار گرفته و آنگاه بسته بعلاوهٔ تەآیند ESP رمزنگاری می‌شوند. این متد می‌تواند برای مقابله با تحلیل ترافیک بکار رود.

چون سرآیند IP شامل آدرس مقصد و احتمالاً دستورات مسیریابی منبع و اطلاعات اختیاری hop-to-hop است، ممکن نیست که بتوان بصورت آسان بستهٔ IP رمزنگاری شده که در ابتدای آن سرآیند ESP قرار دارد را منتقل کرد. مسیریاب‌های بین راه قادر نخواهند بود تا چنین بسته‌ای را پردازش نمایند. بنابراین لازم است که تمام بلوک (سرآیند ESP بعلاوهٔ متن رمزشده بعلاوهٔ Authentication Data، اگر موجود باشد) را با یک سرآیند IP جدید که حاوی اطلاعات کافی برای مسیریابی، ولی نه برای تحلیل تراقیک، باشد کپسولی کرد.

در حالی که مُود حمل‌ونقل برای محافظت اتصالات بین میزبان‌هائی که ESP را حمایت می‌کنند مناسب است، مُود تونل برای استفاده در پیکربندی‌هائی که شامل یک دیوار آتش و یا نوعی دروازهٔ امنیتی دیگر که یک شبکهٔ مورد اعتماد را از شبکه‌های خارجی محافظت می‌کند، مناسب می‌باشد. در این مورد آخر رمزنگاری تنها بین یک میزبان خارجی با دروازهٔ امنیتی و یا بین دو دروازهٔ امنیتی صورت می‌پذیرد. این امر میزبان‌های روی شبکهٔ داخلی را از رَنج رمزنگاری رها ساخته و کار توزیع کلید را با کاهش تعداد کلیدهای مورد نیاز آسان می‌کند. علاوه برآن با تحلیل ترافیک مبتنی بر مقصد نهائی مقابله می‌کند.

موردی را در نظر بگیرید که درآن یک میزبان خارجی می‌خواهد با یک میزبان روی یک شبکهٔ داخلی که بتوسط دیوار آتش از آن محافظت می‌شود و در آن ESP بین میزبان خارجی و دیوار آتش برقرار است، ارتباط پیدا کند. برای انتقال یک سِگمنت لایهٔ حمل‌ونقل از میزبان خارجی به میزبان داخلی قدم‌های زیر بایستی برداشته شود:

۱- مبدأ یک بستهٔ IP درونی با آدرس مقصد میزبان شبکهٔ داخلی را درست می‌کند. این بستهٔ با سرآیند ESP تجهیز شده و آنگاه بسته و ته‌آیند ESP رمزنگاری شده و Authentication Data ممکن است به آن اضافه گردد. بلوک منتجه با یک سرآیند IP جدید (برای IPv6 سرآیند اصلی بعلاوهٔ سرآیندهای الحاقی نظیر routing و hop-to-hop) که آدرس مقصد آن دیوار آتش است کپسولی می‌گردد. این بستهٔ IP بیرونی را شکل می‌دهد.

۲- بستهٔ بیرونی به سمت دیوار آتش مقصد مسیریابی می‌گردد. هر مسیریاب بین راه لازم است که سرآیند IP بیرونی بعلاوهٔ سرآیندهای الحاقی دیگر را وارسی و پردازش نموده ولی نیازی نیست که متن رمزشده را بازدید کند.

۳- دیوار آتش مقصد، سرآیند IP بیرونی باضافهٔ هر سرآیند الحاقی دیگر را بررسی و پردازش می‌کند. آنگاه بر اساس SPI موجود در سرآیند ESP، گرهٔ مقصد بقیهٔ بسته را رمزگشائی کرده تا به متن ساده بستهٔ IP درونی دست‌یابد. این بسته آنگاه در شبکهٔ داخلی انتقال می‌یابد.

۴- بستهٔ درونی از یک یا چند مسیریاب در شبکهٔ داخلی عبور کرده تا به میزبان مقصد برسد.

۶-۵ ترکیب اتحادهای امنیتی

یک SA منفرد می‌تواند یکی از پروتکل‌های AH و یا ESP و نه هردو را اجرا کند. گاهی اوقات یک جریان ترافیک بخصوص، نیازمند هر دو سرویس AH و ESP است. علاوه بر آن یک جریان ترافیک بخصوص ممکن است نیازمند سرویس‌های IPSec بین میزبان‌ها و برای همان جریان، سرویس‌های مجزا بین دروازه‌های امنیتی مثل دیوارهای آتش باشد. در تمام این موارد، SAهای متعددی بایستی برای همان جریان ترافیک بکار گرفته شود تا سرویس‌های IPSec مطلوب را ایجاد نماید. اصطلاح security association bundle به ردیفی از SAها اشاره می‌کند که ترافیک بایستی از درون آنها عبور کرده تا

مجموعهٔ مطلوبی از سرویس‌های IPSec برای آن فراهم شود. SAهای موجود در یک دسته می‌توانند در نقاط انتهائی مختلف و یا همه در یک نقطه خاتمه یابند.

اتحادهای امنیتی می‌توانند به دو صورت با هم دسته‌بندی شوند:

- **مجاورت مُودهای حمل‌ونقل**: به اعمال بیش از یک پروتکل امنیتی به یک بستهٔ IP، بدون استفاده از تونل اشاره می‌کند. این روش ترکیب AH و ESP، فقط ترکیب در یک سطح را مجاز می‌شمارد. لانه‌سازی کردن(nesting) بیشتر سودی ندارد زیرا پردازش در یک مورد IPSec و آنهم در مقصد انتهائی صورت می‌پذیرد.
- **تونل‌های تودرتو**: به اعمال لایه‌های متعدد پروتکل‌های امنیتی که از طریق IPSec اعمال می‌شوند اشاره دارد. این روش سطوح متعدد لانه‌سازی را مجاز دانسته زیرا هر تونل می‌تواند در سایت‌های متفاوت IPSec در طول مسیر، ایجاد شده و یا خاتمه یابد.

این دو روش می‌توانند با هم ترکیب شوند. مثالی در این مورد عبور یک SA حمل‌ونقل بین دو میزبان از درون SA تونل بین دروازه‌های امنیتی، در بخشی از مسیر است.

یک مطلب جالب توجه در هنگام ملاحظهٔ دسته‌های SA، ترتیب قرارگرفتن رمزنگاری و اعتبارسنجی بین یک زوج گرهٔ انتهائی و روش‌های انجام آن است. این مطلب را در دنبالهٔ این بحث مطالعه می‌کنیم. آنگاه به ترکیب‌هائی از SA که شامل حداقل یک تونل هستند اشاره می‌کنیم.

اعتبارسنجی بعلاوهٔ محرمانگی

رمزنگاری و اعتبارسنجی را می‌توان با هم ترکیب کرد تا یک بستهٔ IP را با محرمانگی و اعتبارسنجی بین میزبان‌ها انتقال داد. به چندروش ممکن نگاهی می‌اندازیم.

ESP با قابلیت اعتبارسنجی

این روش در شکل ۹-۶ نشان داده شده است. در این روش، کاربر ابتدا ESP را به دیتائی که باید محافظت شود اعمال کرده و آنگاه میدان Authentication Data را به آن اضافه می‌کند. در واقع دو حالت امکان‌پذیر است:

- **ESP مُود حمل‌ونقل**: اعتبارسنجی و رمزنگاری به محمولهٔ IP که به میزبان تحویل داده می‌شود اعمال شده ولی سرآیند IP محافظت نمی‌شود.
- **ESP مُود تونل**: اعتبارسنجی به تمام بستهٔ IP که به یک آدرس مقصد IP بیرونی (مثلاً دیوار آتش) تحویل می‌گردد اعمال شده و اعتبارسنجی در مقصد صورت می‌پذیرد. تمام بستهٔ IP درونی بتوسط مکانیسم سرّی کردن برای تحویل به مقصد IP درونی محافظت می‌شود.

برای هر دو مورد، اعتبارسنجی بجای اینکه به متن ساده اعمال شود به متن رمزشده اعمال می‌گردد.

مجاورت دو مُود حمل‌ونقل

روش دیگری برای اعمال اعتبارسنجی پس از رمزنگاری، استفاده از دو SA حمل‌ونقل است که درونی آن ESP SA و بیرونی آن AH SA باشد. در این مورد ESP بدون اعتبارسنجی خواهد بود. چون SA درونی یک SA حمل‌ونقل است، رمزنگاری به محمولهٔ IP اعمال می‌شود. بستهٔ منتجه شامل یک سرآیند IP (و احتمالاً ملحقات سرآیند IPv6) و بدنبال آن یک ESP خواهد بود. AH سپس در مُود حمل‌ونقل بکارگرفته شده بطوری که اعتبارسنجی ESP بعلاوهٔ سرآیند IP اولیه (و ملحقات) بغیر از میدان‌های تغییرپذیر را می‌پوشاند. مزیت این روش نسبت به استفاده ساده از یک ESP SA منفرد با اعتبارسنجی اختیاری ESP این است که اعتبارسنجی، میدان‌های بیشتری که شامل آدرس‌های IP مبدأ و مقصد است را می‌پوشاند. عیب آن وجود سربارهٔ دو SA در مقایسه با یک SA است.

مجاورت مُود حمل‌ونقل با مُود تونل

استفاده از اعتبارسنجی قبل از رمزنگاری می‌تواند به دلایل متعددی ارجح باشد. اول اینکه چون دیتای اعتبارسنجی بتوسط رمزنگاری محافظت می‌شود، غیرممکن است که کسی بدون اینکه لو رود بتواند پیام را گرفته و اطلاعات اعتبارسنجی آن را تغییر دهد. ثانیاً ممکن است لازم باشد که اطلاعات اعتبارسنجی همراه پیام را برای مصارف آتی در مقصد ذخیره کرد. این امر در صورتی که اطلاعات اعتبارسنجی به پیام رمزنگاری نشده اعمال گردد ساده‌تر خواهد بود، در غیراینصورت پیام بایستی دوباره رمزنگاری شود تا اطلاعات مربوط به اعتبارسنجی را بتوان تأئید نمود.

یکی از روش‌های اعمال اعتبارسنجی قبل از رمزنگاری بین دو میزبان این است که از یک دسته که شامل یک AH transport SA درونی و یک ESP tunnel SA بیرونی است استفاده کرد. در این مورد اعتبارسنجی به محمولهٔ IP باضافهٔ سرآیند IP (وملحقات)، بجز میدان‌های تغییرپذیر، اعمال خواهد شد. بستهٔ IP نتیجه شده آنگاه در مُود تونل بتوسط ESP پردازش خواهد شد که نتیجهٔ آن این است که تمام بستهٔ درونی اعتبارسنجی شده، رمزنگاری شده و یک سرآیند IP بیرونی جدید (وملحقات) به آن اضافه می‌گردد.

ترکیب‌های اصلی اتحادهای امنیتی

اسناد معماری IPSec چهار مثال از ترکیب SAها که بایستی بتوسط میزبان‌های منطبق با IPSec (مثل ایستگاه‌های کاری، سِرورها) و یا دروازه‌های امنیتی (مثل دیوار آتش، مسیریاب) مورد حمایت قرار گیرند را ذکر کرده است. این ترکیب‌ها در شکل ۱۰-۶ نشان داده شده‌اند. قسمت پائین هر مورد در شکل نمایش دهندهٔ اتصال فیزیکی عناصر است. قسمت فوقانی نمایشگر اتصال منطقی از طریق یک یا چند SA تودرتو است. هر SA می‌تواند یا AH و یا ESP باشد. برای SAهای میزبان- به- میزبان، مُود می‌تواند حمل‌ونقل و یا تونل باشد. در غیر اینصورت مُود حتماً تونل است.

در **مورد اول**، کل امنیت بتوسط سیستم‌های انتهائی که از IPSec استفاده می‌کنند فراهم شده است. برای هر دو سیستم انتهائی که از طریق SA باهم ارتباط برقرار می‌کنند، بایستی کلیدهای سرّی مناسب به اشتراک گذاشته شوند. موارد زیر ترکیب‌های ممکن را نشان می‌دهد:

الف- AH در مُود حمل‌ونقل

ب - ESP در مُود حمل‌ونقل

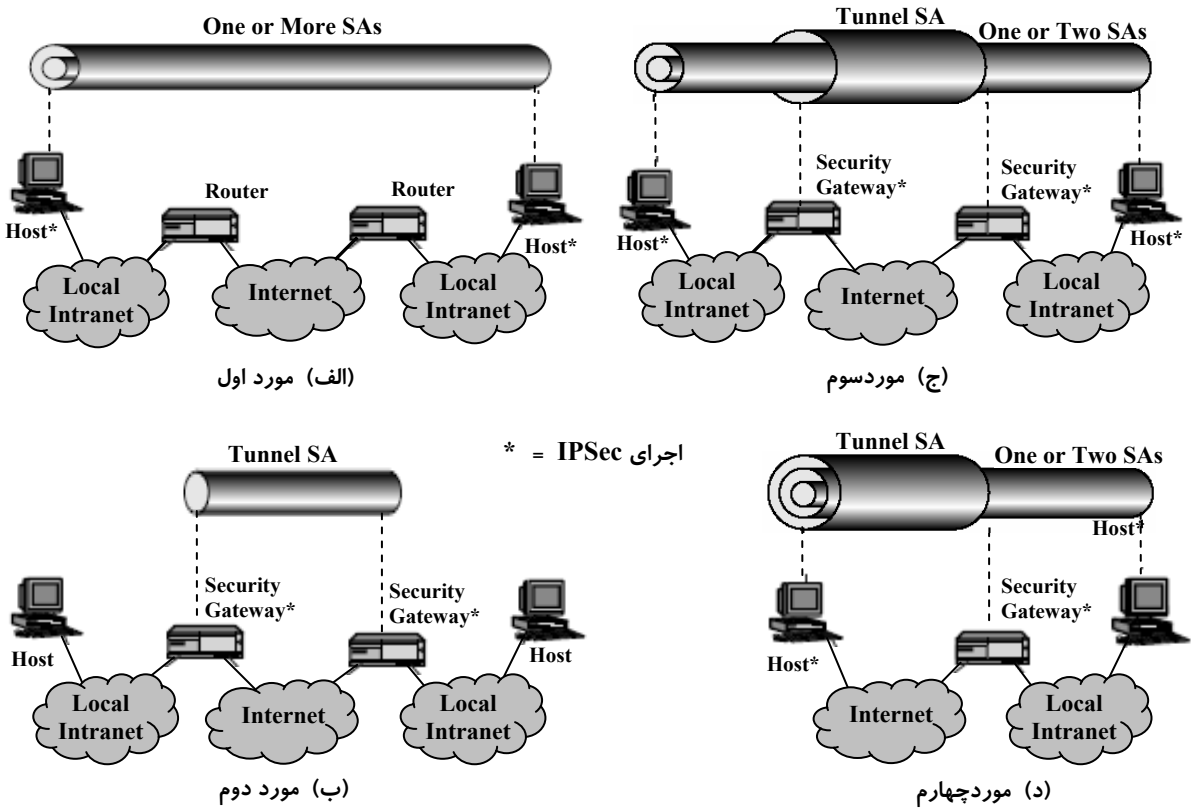
ج- AH به دنبال ESP در مُود حمل‌ونقل (یک ESP SA در درون یک AH SA).

د- هریک از موارد الف، ب، یا ج در داخل یک AH یا ESP در مُود تونل.

قبلاً در مورد انواع ترکیب‌های ذکرشده که می‌تواند برای اعتبارسنجی، رمزنگاری، اعتبارسنجی قبل از رمزنگاری و اعتبارسنجی بعد از رمزنگاری بکار گرفته شود صحبت کرده‌ایم.

برای **مورد دوم**، امنیت فقط بین دروازه‌ها (مسیریاب‌ها، دیوارهای آتش و غیره) فراهم شده و هیچ میزبانی IPSec را بکار نمی‌گیرد. این مثال، استفاده از یک شبکهٔ خصوصی مجازی را روشن می‌کند. سند معماری امنیتی تعیین می‌کند که تنها یک تونل منفرد SA برای این حالت مورد نیاز است. تونل می‌تواند AH، ESP و یا ESP با اعتبارسنجی را حمایت نماید. چون سرویس‌های IPSec به تمام بستهٔ درونی اعمال می‌شود، تونل‌های تودرتو مورد نیاز نیستند.

مورد سوم، روی مورد دوم و با اضافه کردن امنیت سر- به- سر ساخته شده است. همان ترکیب‌های بحث شده در موارد ۱ و ۲ در اینجا نیز مجاز هستند. تونل دروازه- به- دروازه، اعتبارسنجی، محرمانگی و یا هردوی آنها را بین سیستم‌های انتهائی ایجاد می‌کند. وقتی تونل دروازه- به- دروازه، ESP است تا حدی محرمانگی ترافیک را نیز ایجاد می‌کند. هر یک از میزبان‌ها خود می‌توانند سرویس‌های IPSec اضافی را نیز بوسیلهٔ SAهای سر- به- سر برای کاربردهای مختلف و یا کاربرهای مختلف بکار گیرند.



شکل ۱۰-۶ ترکیب‌های اصلی اتحادهای امنیتی

مورد چهارم، از یک میزبان راه دورکه از اینترنت برای دست‌یابی به یک دیوار آتش یک سازمان و سپس به یک سِرور و یا ایستگاه کاری پشت آن دیوار آتش استفاده می‌کند حمایت می‌کند. تنها مُود تونل بین میزبان دور و دیوار آتش مورد نیاز است. همانند مورد اول یک یا چند SA می‌تواند بین میزبان دور و میزبان محلی مورد استفاده قرارگیرد.

۶-۶ مدیریت کلید

بخش مدیریت کلید IPSec، تعیین و توزیع کلیدهای سرّی را بعهده دارد. یک مورد معمول ارتباط بین دو کاربرد، نیاز به چهار کلید دارد. یک جفت کلید ارسال و دریافت برای AH و یک جفت کلید ارسال و دریافت برای ESP. معماری اسناد IPSec به حمایت از دو نوع مدیریت کلید حکم می‌دهد:

- **دستی**: مدیر سیستم، هر سیستم را با کلیدهای خودش و کلیدهای سیستم‌های ارتباطی دیگر بصورت دستی پیکربندی می‌نماید.این مورد برای محیط‌های کوچک و نسبتاً استاتیک کارآئی دارد.
- **خودکار**: یک سیستم خودکار، خلق کلید برای SAها بر اساس تقاضا را برعهده داشته و استفاده از کلیدها در یک سیستم توزیع شده گسترده با پیکربندی در حال تکامل را تسهیل می‌نماید.

پروتکل مدیریت خودکار کلید IPSec را ISAKMP/Oakley می‌نامند و شامل عناصر زیر است:

- **پروتکل تعیین کلید Oakley**: Oakley یک پروتکل مبادلهٔ کلید است که مبتنی بر الگوریتم Diffie-Hellman بوده اما امنیت بیشتری را فراهم می‌آورد. Oakley از اینجهت عام است که فرمت خاصی را دیکته نمی‌کند.
- **پروتکل اتحاد امنیتی و مدیریت کلید اینترنت (ISAKMP)**: ISAKMP چهارچوبی را برای مدیریت کلید در اینترنت فراهم آورده و حمایت‌های جانبی همانند نوع فرمت‌ها بمنظور توافق بر روی جنبه‌های امنیتی را ایجاد می‌کند.

ISAKMP فی‌ذاته الگوریتم مبادلهٔ کلید خاصی را تعیین نمی‌کند بلکه ISAKMP شامل یک مجموعه از انواع پیام‌هاست که استفاده از الگوریتم‌های مبادلهٔ کلید متنوعی را ممکن می‌سازد. Oakley الگوریتم مبادلهٔ کلید خاصی است که برای استفاده از نسخهٔ اولیهٔ ISAKMP اجباری بود.

ابتدا مروری بر Oakley داشته و آنگاه به ISAKMP نگاهی می‌اندازیم.

پروتکل تعیین کلید Oakley

Oakley یک فرم پالایش شده از الگوریتم مبادلهٔ کلید Diffie-Hellman است. بیاد آورید که Diffie-Hellman شامل تعامل‌های زیر بین کاربر A و B بود. از قبل روی دو پارامتر q که یک عدد اول بزرگ و α که یک ریشهٔ اولیهٔ q است توافق می‌شود. A یک عدد صحیح تصادفی X_A را بعنوان کلید خصوصی خود انتخاب می‌کند و کلید عمومی خود یعنی $Y_A = \alpha^{X_A} \bmod q$ را برای B می‌فرستد. بهمین ترتیب B یک عدد صحیح تصادفی X_B را بعنوان کلید خصوصی خود

انتخاب کرده و کلید عمومی خود یعنی $Y_B = \alpha^{X_B} \bmod q$ را برای A ارسال می‌کند. هریک از دو طرف اکنون می‌توانند کلید سرّی اجلاس را بصورت زیر محاسبه نمایند:

$$K = (Y_B)^{X_A} \bmod q = (Y_A)^{X_B} \bmod q = \alpha^{X_A X_B} \bmod q$$

الگوریتم Diffie-Hellman دو مشخصهٔ جالب دارد:

- کلیدهای سرّی فقط وقتی مورد نیازاند خلق می‌شوند. هیچ نیازی نیست تا کلیدهای سرّی را برای مدتی طولانی ذخیره کرد و بدین ترتیب آنها را در مقابل آسیب‌پذیری‌های اضافی قرار داد.
- مبادله کلید نیاز به هیچ زیرساخت از قبل موجودی، بجز توافق روی پارامترهای q و α ندارد.

با وجود این ضعف‌هائی در روش Diffie-Hellman موجود است که در [HUIT98] به آنها اشاره شده است:

- هیچ اطلاعاتی در مورد هویت طرفین به دست نمی‌دهد.
- در معرض حملهٔ man-in-the-middle قرار دارد که در آن طرف سوم C خود را در هنگام مکالمه با A بجای B، و در هنگام مکالمه با B بجای A، جا می‌زند. هردو طرف A و B برای خلق کلید سرّی با C به توافق می‌رسند که در این صورت C می‌تواند به ترافیک گوش کرده و آن را عبور دهد. حملهٔ man-in-the-middle چنین جلو می‌رود:

۱- B کلید عمومی خود Y_B را در پیامی به مقصد A می‌فرستد.

۲- دشمن (E) این پیام را می‌گیرد. E کلید عمومی B را نگاه داشته و یک پیام به مقصد A ارسال کرده که ID کاربر B را داشته ولی کلید عمومی Y_E را حمل می‌کند. این پیام بنحوی ارسال می‌شود که بنظر می‌رسد از طرف سیستم میزبان B ارسال شده است. A پیام E را گرفته و کلید عمومی E که ID کاربر B را دارد نگاه می‌دارد. بهمین ترتیب، E یک پیام را با کلید عمومی E برای B فرستاده و چنین وانمود می‌کند که از A آمده است.

۳- B یک کلید سرّی K_1 بر اساس کلید خصوصی B و Y_E را محاسبه می‌کند. A یک کلید سرّی K_2 که بر اساس کلید خصوصی A و Y_B قرار دارد را محاسبه می‌کند. E کلید K_1 را با استفاده از کلید خصوصی X_E و Y_B، و کلید K_2 را با استفاده از X_E و Y_A محاسبه می‌نماید.

۴- از این به بعد E قادر است تا پیام‌های A به B و پیام‌های B به A را گرفته و رمزنگاری آنها را درطول مسیر تغییردهد. در این صورت نه A و نه B متوجه نمی‌شوندکه آنها با E ارتباط دارند و نه با یکدیگر.

- از نظر محاسباتی حجیم است. در نتیجه در برابر یک حملهٔ clogging که در آن دشمن تقاضای کلیدهای بسیاری را ارسال می‌نماید آسیب‌پذیر است. منابع قربانی بجای انجام کار واقعی درگیر انجام محاسبات بی‌حاصل نمائی و پیمانه‌ای می‌گردند.

Oakley برای بکارگیری مزایای Diffie-Hellman و در عین حال مقابله با ضعف‌های آن طراحی شده است.

خصوصیات Oakley

الگوریتم Oakley با پنج خاصیت مهم مشخص می‌گردد:

۱- از مکانیسمی بنام cookies برای مقابله با حملات clogging استفاده می‌کند.
۲- دو طرف را قادر می‌سازد تا برای ایجاد یک *group* به توافق برسند که این در اصل تعیین پارامترهای اصلی مبادلهٔ کلید Diffie-Hellman است.
۳- برای اطمینان از مقابله با حملات بازخوانی، از nonce استفاده می‌کند.
۴- مبادلهٔ کلیدهای عمومی Diffie-Hellman را ممکن می‌سازد.
۵- مبادلهٔ کلید Diffie-Hellman را برای مقابله با حملات man-in-the-middle اعتبارسنجی می‌نماید.

Diffie-Hellman را قبلاً مورد بحث قرار داده‌ایم. اجازه دهید تا بقیهٔ این عناصر را به نوبت بررسی کنیم. اول، مسألهٔ حملات clogging را در نظر می‌گیریم. در این حمله یک دشمن آدرس منبع یک کاربر قانونی را تقلیدکرده و یک کلید عمومی Diffie-Hellman را برای قربانی می‌فرستد. قربانی عملیات نمائی و پیمانه‌ای را انجام داده تا کلید سرّی را محاسبه کند. پیام‌های پشت سرهم و تکراری از این دست می‌توانند سیستم قربانی را با کارهای بی‌حاصل کُند کنند. **مبادلهٔ cookie** هریک از دو سمت را ملزم می‌سازد تا یک عدد تصادفی، یا همان cookie، را در پیام اولیه ارسال نمایند که طرف دیگر آن را تأئید کند. این تأئید بایستی در اولین پیام مبادلهٔ کلید Diffie-Hellman تکرار شود. اگر آدرس منبع جعل گردد، دشمن هیچ جوابی را دریافت نمی‌دارد. بنابراین یک دشمن تنها می‌تواند یک کاربر را به تولید تأئیدیه مشغول سازد و نه اینکه او را به محاسبات Diffie-Hellman مشغول نماید.

ISAKMP به ملاحظهٔ سه مطلب در تولید cookie حکم می‌دهد:

۱- cookie بایستی وابسته به طرف‌های مشخص باشد. این امر یک حمله‌کننده را از دریافت یک cookie با استفاده از یک آدرس IP حقیقی و پورت UDP، و سپس استفاده از آن به منظور فروبردن قربانی در باطلاق تقاضاهای مکرر از آدرس‌های IP و یا پورت‌های بصورت تصادفی انتخاب شده باز می‌دارد.
۲- نبایستی برای هیچکس بجز واحد صادرکننده cookie امکان داشته باشد که بتواند cookieای درست کندکه بتوسط همان واحد پذیرفته شود. برای تحقق این امر، واحد صادرکنندهٔ cookie بایستی از اطلاعات سرّی محلی در تولید و تأئید آتی یک cookie استفاده کند. بایستی ممکن نباشد که این اطلاعات سرّی را از هیچ cookie خاص استخراج کرد. نکته نهفته در این الزام این است که واحد صادرکننده لازم نیست تا کپی cookieهایش را ذخیره کند، که در این صورت در برابر کشف آسیب‌پذیرتر خواهند بود، بلکه باید بتواند در هر زمان که لازم است cookie ورودی را تأئید نماید.
۳- روش‌های تولید و تأئید cookie بایستی سریع باشند تا با حملاتی که هدف آنها تخریب منابع پردازشی و سرگرم کردن بی‌حاصل آنهاست مقابله شود.

روش توصیه شده برای تولید cookie این است که از یک تابع درهم‌ساز سریع (مثل MD5) روی آدرس‌های IP منبع و مقصد، پورت‌های UDP منبع و مقصد، و یک اندازهٔ سرّی تولید شده در محل استفاده گردد.

Oakley استفاده از **گروه**های مختلف برای مبادلهٔ کلید Diffie-Hellman را حمایت می‌کند. هر گروه شامل تعریف دو پارامتر عمومی و هویت الگوریتم مورد استفاده است. مشخصه‌های فعلی شامل گروه‌های زیر می‌باشند:

- به توان رساندن پیمانه‌ای با یک پیمانهٔ ۷۶۸- بیتی

$$q = 2^{768} - 2^{704} - 1 + 2^{64} \times (\lfloor 2^{638} \times \pi \rfloor + 149686)$$
$$\alpha = 2$$

- به توان رساندن پیمانه‌ای با یک پیمانهٔ ۱۰۲۴- بیتی

$$q = 2^{1024} - 2^{960} - 1 + 2^{64} \times (\lfloor 2^{894} \times \pi \rfloor + 129093)$$
$$\alpha = 2$$

- به توان رساندن پیمانه‌ای با یک پیمانهٔ ۱۵۳۶- بیتی
- پارامترها بایستی تعیین شوند
- گروه خَم بیضوی روی 2^{155}
  - موّلد (هکزادسیمال): $X = 7B$ و $Y = 1C8$
  - پارامترهای خم بیضوی (هکزادسیمال): $A = 0$ و $Y = 7338F$
- گروه خم بیضوی روی 2^{185}
  - موّلد (هکزادسیمال): $X = 18$ و $Y = D$
  - پارامترهای خم بیضوی (هکزادسیمال): $A = 0$ و $Y = 1EE9$

سه گروه اول الگوریتم‌های کلاسیک Diffie-Hellman هستند که از بتوان رساندن پیمانه‌ای استفاده می‌کنند. دو گروه آخر از خم بیضوی مشابه با Diffie-Hellman استفاده می‌کنند که قبلاً در مورد این روش صحبت شده است. Oakley از **nonce**ها برای اطمینان از مقابله در برابر حملات بازخوانی استفاده می‌کند. هر nonce یک عدد شبه تصادفی تولیدشده در محل است. nonceها در پاسخ‌ها ظاهر شده و در خلال بخش‌های معینی از عملیات مبادله برای امن ماندن رمزنگاری می‌شوند.

سه روش **اعتبارسنجی** متفاوت می‌تواند به همراه Oakley بکار گرفته شود:

- **امضاء دیجیتال**: مبادله با امضاء یک hash که در هر دو سمت قابل حصول باشد اعتبارسنجی می‌گردد. هر طرف hash را با کلید خصوصی خود رمزنگاری می‌کند. اندازهٔ hash روی پارامترهای مهم همانند ID کاربر و nonceها محاسبه می‌گردد.
- **رمزنگاری کلید- عمومی**: مبادله بتوسط پارامترهای رمزنگاری همچون ID ها و nonce ها و با استفاده از کلید خصوصی فرستنده اعتبارسنجی می‌شود.
- **رمزنگاری کلید- متقارن**: یک کلید که بتوسط یک مکانیسم خارج از محدوده تهیه شده است می‌تواند از طریق رمزنگاری متقارن پارامترها، برای اعتبارسنجی مبادله بکار رود.

I → R: CKY_I, OK_KEYX, GRP, g^x, EHAO, NIDP, ID_I, ID_R, N_I, $S_{KI}[ID_I \| ID_R \| N_I \| GRP \| g^x \| EHAO]$

R → I: CKY_R, CKY_I, OK_KEYX, GRP, g^y, EHAS, NIDP, ID_R, ID_I, N_R, N_I, $S_{KR}[ID_R \| ID_I \| N_R \| N_I \| GRP \| g^y \| g^x \| EHAS]$

I → R: CKY_I, CKY_R, OK_KEYX, GRP, g^x, EHAS, NIDP, ID_I, ID_R, N_I, N_R, $S_{KI}[ID_I \| ID_R \| N_I \| N_R \| GRP \| g^x \| g^y \| EHAS]$

Notation:

I	=	Initiator
R	=	Responder
CKYI, CKYR	=	Initiator, responder cookies
OK_KEYX	=	Key exchange message type
GRP	=	Name of Diffie-Hellman group for this exchange
g^x, g^y	=	Public key of initiator, responder; g^{xy} = session key from this exchange
EHAO, EHAS	=	Encryption, hash, authentication functions, offered and selected
NIDP	=	Indicates encryption is not used for remainder of this message
ID_I, ID_R	=	Identifier for initiator, responder
N_I, N_R	=	Random nonce supplied by initiator, responder for this exchange
$S_{KI}[X]$, $S_{KR}[X]$	=	Indicates the signature over X using the private key (signing key) of initiator, responder

شکل ۱۱-۶ مثال مبادله کلید Aggressive Oakley

مثالی از مبادلهٔ Oakley

مشخصه‌های Oakley شامل مثال‌هائی از مبادلاتی است که تحت این پروتکل قابل اجرا هستند. برای اینکه درک بهتری از Oakley وجود داشته باشد، یکی از این مثال‌ها که در مشخصات، aggressive key exchange نامیده می‌شود را ارائه می‌کنیم. دلیل این نام این است که تنها سه پیام ردوبدل می‌شود.

شکل ۱۱-۶ پروتکل aggressive key exchange را نشان می‌دهد. در قدم اول آغازگر (I) یک cookie، گروهی که بکار خواهد رفت، و کلید عمومی Diffie-Hellman آغازگر I برای این تبادل را ارسال می‌کند. I همچنین روش رمزنگاری کلید- عمومی، تابع درهم‌ساز و الگوریتم اعتبارسنجی پیشنهاد شده که در این مبادله بکار خواهد رفت را مشخص می‌سازد. در این پیام همچنین شناسه‌های I و R (پاسخ دهنده) و nonce مربوط به I وجود دارد. بالاخره I یک امضاء که با استفاده از کلید خصوصی I روی دو شناسه، nonce، گروه، کلید عمومی Diffie-Hellman و الگوریتم‌های پیشنهادی انجام شده است را به انتهای پیام وصل می‌کند.

وقتی R پیام را دریافت می‌کند، R امضاء را با استفاده از کلید عمومی I تأئید می‌نماید. R تأئید خود را با پس فرستادن cookie متعلق به I، شناسه، nonce و همچنین گروه به I انجام می‌دهد. R همچنین در پیام خود یک cookie، کلید عمومی Diffie-Hellman خود، الگوریتم‌های انتخاب شده (که بایستی از میان الگوریتم‌های ارسالی انتخاب شده باشد)، شناسهٔ R و nonce خود را می‌گنجاند. بالاخره R یک امضاء که دو شناسه، دو nonce، گروه، دو کلید عمومی Diffie-Hellman و الگوریتم‌های انتخاب شده را با کلید خصوصی R امضاء کرده است به پیام وصل می‌کند.

وقتی I پیام دوم را دریافت می‌کند، I امضاء را با استفاده از کلید عمومی R باز می‌کند. اندازه‌های nonce در پیام اطمینان می‌دهند که این بازخوانی یک پیام کهنه نیست. برای کامل کردن این مبادله، I بایستی پیام دیگری را برای R فرستاده و دریافت کلید عمومی R را اعلام نماید.

ISAKMP

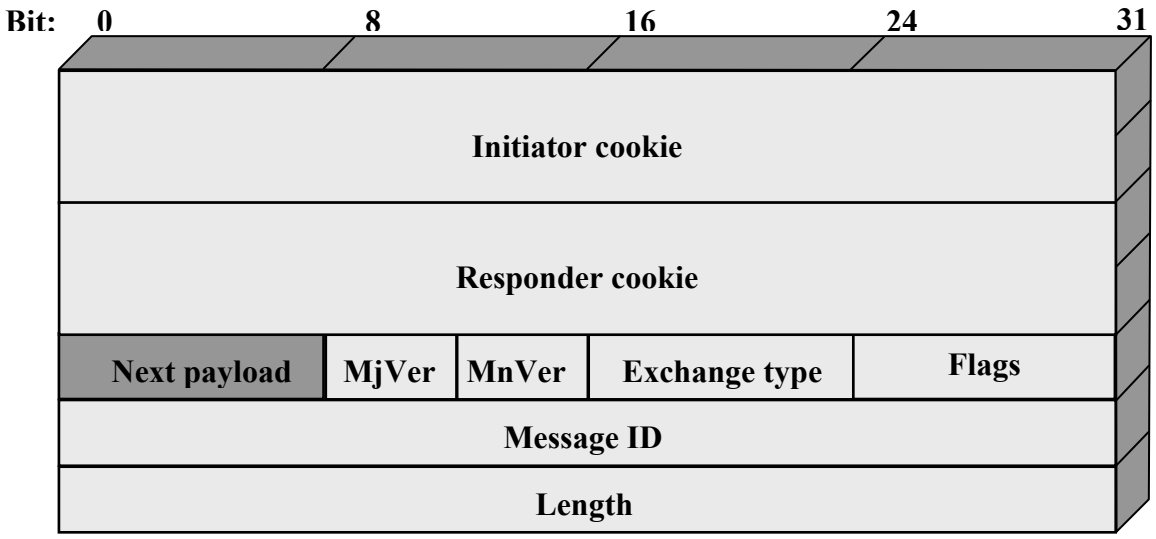
یک پیام ISAKMP (Internet Security Association and Key Management Protocol)، رویه‌ها و فرمت بسته‌ها را برای برقراری، توافق، جرح و تعدیل و حذف اتحادهای امنیتی تعریف می‌کند. بعنوان مرحله‌ای از برقراری SA، ISAKMP محموله‌های مربوط به مبادلهٔ تولید کلید و داده‌های اعتبارسنجی را تعریف می‌کند. این فرمت محموله‌ها یک چهارچوب مستقل از پروتکل خاص مبادلهٔ کلید، الگوریتم رمزنگاری و مکانیسم اعتبارسنجی را فراهم می‌آورد.

فرمت سرآیند ISAKMP

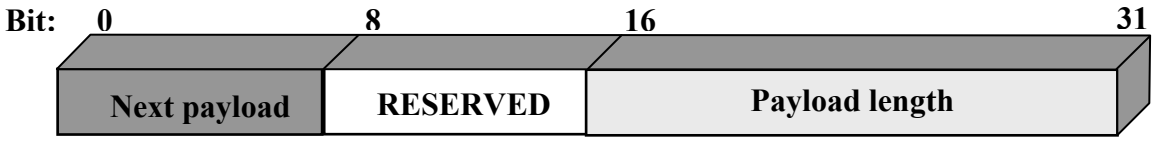
یک پیام ISAKMP شامل یک سرآیند ISAKMP است که بتوسط یک یا چند محموله دنبال می‌شود. تمام اینها در یک پروتکل حمل‌ونقل، حمل می‌شوند. مشخصه، حمایت از UDP بعنوان پروتکل حمل‌ونقل در پیاده‌سازی‌ها را اجباری می‌داند.

شکل ۱۲-۶ فرمت سرآیند یک پیام ISAKMP را نشان می‌دهد. این سرآیند شامل میدان‌های زیر است:

- **Initiator Cookie (64 bits)**: cookie واحدی که برای برقراری SA، تعیین SA و یا حذف SA اقدام کرده است.
- **Responder Cookie (64 bits)**: cookie واحد پاسخ‌دهنده که در اولین پیام از طرف آغازگر، خالی خواهد بود.
- **Next Payload (8 bits)**: نشان‌دهندهٔ نوع اولین محموله در پیام است. محموله‌ها در بخش بعد تعریف خواهند شد.
- **Major Version (4 bits)**: نسخهٔ اصلی ISAKMP مورد استفاده را نشان می‌دهد.
- **Minor Version (4 bits)**: نسخهٔ فرعی ISAKMP مورد استفاده را نشان می‌دهد.
- **Exchange Type (8 bits)**: نوع مبادله را نشان می‌دهد. بعداً در همین بخش به آن اشاره خواهد شد.
- **Flags (8 bits)**: موارد مختص به این مبادلهٔ ISAKMP را نشان می‌دهد. تا کنون دو بیت از این میدان تعریف شده است. بیت Encryption که در صورتی 1 است که تمام محموله‌های بعد از سرآیند با الگوریتم رمزنگاری مرتبط به این SA رمزنگاری شده باشند. بیت Commit برای اطمینان از این است که مواد رمزنگاری شده، قبل از کامل شدن برقراری SA دریافت نشده باشند.
- **Message ID (32 bits)**: ID یکتای مختص این پیام.
- **Length (32 bits)**: طول کل پیام (سرآیند بعلاوهٔ تمام محموله‌ها) بر حسب اُکتت.



(الف) سرآیند ISAKMP



(ب) سرآیند عمومی محموله‌ها (payload)

شکل ۱۲-۶ فرمت ISAKMP

انواع محموله‌های ISAKMP

تمام محموله‌های ISAKMP با یک سرآیند عمومی که در شکل ۱۲-۶ب نشان داده شده است، شروع می‌شوند. میدان Next Payload، اگر این آخرین محمولهٔ پیام باشد دارای اندازهٔ 0 و در غیر اینصورت اندازهٔ نوع محمولهٔ بعد را نشان می‌دهد. میدان Payload Length نشان‌دهندهٔ طول این محموله بر حسب اُکتت بوده که شامل سرآیند عمومی محموله نیز می‌گردد.

جدول ۳-۶ انواع محموله‌های تعریف شده برای ISAKMP را نشان داده و میدان‌ها یا پارامترهائی که بخشی از هر محموله هستند را نیز مشخص می‌نماید. **SA payload** برای شروع استقرار SA بکار می‌رود. در این محموله، پارامتر Domain of Interpretation نمایش‌دهندهٔ DOI است که توافق تحت نظر آن صورت می‌پذیرد. IPSec DOI یک مثال آن است ولی ISAKMP می‌تواند در مقوله‌های دیگر نیز بکار رود. پارامتر Situation، خط‌مشی امنیتی این توافق را تعریف می‌کند. در واقع سطوح امنیتی لازم برای رمزنگاری و محرمانگی مشخص می‌شوند (مثلاً سطح حساسیت، بخش امنیتی).

Proposal payload شامل اطلاعاتی است که در خلال توافق SA بکار گرفته می‌شود. محموله، نمایش‌دهندهٔ پروتکل این SA (ESP یا AH) است که برای آن سرویس‌ها و مکانیسم‌ها مورد توافق قرار می‌گیرند. محموله همچنین شامل SPI واحد فرستنده و تعداد تبدیل‌هاست. هر تبدیل در یک محمولهٔ تبدیل قرار دارد. استفاده از محموله‌های با چند تبدیل، آغازگر را قادر می‌سازد تا حالات ممکن متعددی را پیشنهاد نماید که از بین آنها پاسخ‌دهنده بایستی یکی را انتخاب کرده و یا پاسخ منفی دهد.

Transform payload یک تبدیل امنیتی را تعریف می‌کند که از آن برای امن کردن کانال ارتباطی برای پروتکل مشخص شده استفاده می‌شود. پارامتر # Transform بمنظور شناسائی این محمولهٔ مخصوص بکار می‌رود تا پاسخ‌دهنده بتواند از آن برای موافقت با این تبدیل استفاده کند (مثلاً 3DES برای ESP، HMAC-SHA-1-96 برای AH) که ملحقات مربوطه نیز در آن وجود دارد (مثلاً طول hash).

Key Exchange payload می‌تواند برای تکنیک‌های متنوع مبادلهٔ کلید بکار رود که شامل Oakley، Diffie-Hellman و مبادلهٔ کلید RSA-based برای PGP است. میدان دیتای Key Exchange شامل دیتای مورد نیاز برای تولید یک کلید اجلاس بوده و مستقل از الگوریتم مبادلهٔ کلید بکار رفته است.

Identification payload برای تعیین هویت طرفین ارتباط بکار رفته و ممکن است برای تعیین اعتبار اطلاعات استفاده شود. معمولاً میدان ID Data شامل آدرس‌های IPv4 یا IPv6 است.

Certificate payload یک گواهی‌نامهٔ کلید- عمومی را منتقل می‌کند. میدان Certificate Encoding نمایش‌دهندهٔ نوع گواهی‌نامه و یا اطلاعات مربوط به گواهی‌نامه است که ممکن است شامل موارد زیر باشد:

- PKCS#7 wrapped X.509 certificate
- PGP certificate
- DNS signed key
- X.509 certificate-signature
- X.509 certificate-key exchange
- Kerberos tokens
- Certificate Revocation List (CRL)
- Authority Revocation List (ARL)
- SPKI certificate

در هر نقطه از مبادلهٔ ISAKMP، فرستنده ممکن است یک محمولهٔ **Certificate Request** برای درخواست گواهی‌نامهٔ واحد مرتبط ارسال کند. محموله ممکن است بیش از یک نوع گواهی‌نامهٔ قابل قبول و یا بیش از یک مسئول صدور گواهی قابل قبول را تعیین نماید.

Hash Payload شامل دیتای تولیدشده بتوسط تابع درهم‌ساز در بخشی از پیام و/ یا حالت ISAKMP است. این محموله برای تأئید صحت داده‌ها در یک پیام و یا برای احراز هویت واحدهای نظیر بکار رود.

جدول ۳-۶ انواع محموله‌های ISAKMP

Type	Parameters	Description
Security Association (SA)	Domain of Interpretation, Situation	Used to negotiate security attributes and indicate the DOI and Situation under which negotiation is taking place.
Proposal (P)	Proposal #, Protocol-ID, SPI Size, # of Transforms, SPI	Used during SA negotiation; indicates protocol to be used and number of transforms.
Transform (T)	Transform #, Transform-ID, SA Attributes	Used during SA negotiation; indicates transform and related SA attributes.
Key Exchange (KE)	Key Exchange Data	Supports a variety of key exchange techniques.
Identification (ID)	ID Type, ID Data	Used to exchange identification information.
Certificate (CERT)	Cert Encoding, Certificate Data	Used to transport certificates and other certificate- related information.
Certificate Request (CR)	# Cert Types, Certificate Types, # Cert Auths, Certificate Authorities	Used to request certificates; indicates the types of certificates requested and the acceptable certificate authorities.
Hash (HASH)	Hash Data	Contains data generated by a hash function.
Signature (SIG)	Signature Data	Contains data generated by a digital signa ture function.
Nonce (NONCE)	Nonce Data	Contains a nonce.
Notification (N)	DOI, Protocol-ID, SPI Size, Notify Message Type, SPI, Notification Data	Used to transmit notification data, such as an error condition.
Delete (D)	DOI, Protocol-ID, SPI Size, # of SPIs, SPI (one or more)	Indicates an SA that is no longer valid.

Signature payload شامل دیتای تولیدشده بتوسط یک تابع امضاء دیجیتال روی بخشی از پیام و/ یا حالت ISAKMP است. این محموله برای تأئید صحت دیتا در یک پیام بکار رفته و ممکن است برای سرویس‌های عدم انکار نیز مورد استفاده قرار گیرد.

Nonce payload شامل یک سری داده‌های تصادفی است که از آنها برای بهنگام بودن و جلوگیری از حملات بازخوانی استفاده می‌شود.

Notification payload شامل اطلاعات خط و یا حالت مرتبط با این SA و یا توافقات این SA است. پیام‌های خطا در ISKAMP شامل موارد تعریف شدهٔ زیراند:

Invalid Payload Type	Invalid Protocol ID	Invalid Cert Encoding
DOI Not Supported	Invalid SPI	Invalid Certificate
Situation Not Supported	Invalid Transform ID	Bad Cert Request Syntax
Invalid Cookie	Attributes Not Supported	Invalid Cert Authority
Invalid Major Version	No Proposal Chosen	Invalid Hash Information
Invalid Minor Version	Bad Proposal Syntax	Authentication Failed
Invalid Exchange Type	Payload Malformed	Invalid Signature
Invalid Flags	Invalid Key Information	Address Notification
Invalid Message ID		

تنها پیام حالت که تاکنون تعریف شده است، Connected است. علاوه براین یادآوری‌های ISAKMP، یادآوری‌های مختص به DOI نیز مورد استفاده قرار می‌گیرند. برای IPSec پیام‌های حالت اضافی زیر تعریف شده‌اند:

- **Responder-Lifetime**: زمان حیات SA که بتوسط پاسخ‌دهنده انتخاب شده است را نشان می‌دهد.
- **Replay-Status**: برای پاسخ مثبت پاسخ‌دهنده به این سؤال که آیا او عملیات تشخیص anti-replay را انجام خواهد داد یا نه مورد استفاده است.
- **Initial-Contact**: طرف دیگر را از اینکه آیا این اولین SA برقرار شده با سیستم دور است مطلع می‌سازد. گیرندهٔ این یادآوری آنگاه بایستی هر SA ای که برای سیستم فرستنده دارد را، با فرض اینکه سیستم فرستنده reboot کرده و دیگر دسترسی به این SAها ندارد، حذف نماید.

Delete payload یک یا چند SA که فرستنده از پایگاه دادهٔ خود حذف کرده و دیگر معتبر نیستند را نشان می‌دهد.

جدول ۴-۶ انواع مبادله‌های ISAKMP

مبادله	توضیح
(a) Base Exchange	
(1) **I → R:** SA; NONCE	Begin ISAKMP-SA negotiation
(2) **R → I:** SA; NONCE	Basic SA agreed upon
(3) **I → R:** KE; ID_I; AUTH	Key generated; Initiator identity verified by responder
(4) **R → I:** KE; ID_R; AUTH	Responder identity verified by initiator; Key generated; SA established
(b) Identity Protection Exchange	
(1) **I → R:** SA	Begin ISAKMP-SA negotiation
(2) **R → I:** SA	Basic SA agreed upon
(3) **I → R:** KE; NONCE	Key generated
(4) **R → I:** KE; NONCE	Key generated
(5)* **I → R:** ID_I; AUTH	Initiator identity verified by responder
(6)* **R → I:** ID_R; AUTH	Responder identity verified by initiator; SA established
(c) Authentication Only Exchange	
(1) **I → R:** SA; NONCE	Begin ISAKMP-SA negotiation
(2) **R → I:** SA; NONCE; ID_R; AUTH	Basic SA agreed upon; Responder identity verified by initiator
(3) **I → R:** ID_I; AUTH	Initiator identity verified by responder; SA established
(d) Aggressive Exchange	
(1) **I → R:** SA; KE; NONCE; ID_I	Begin ISAKMP-SA negotiation and key Exchange
(2) **R → I:** SA; KE; NONCE; ID_R; AUTH	Initiator identity verified by responder; Key generated; Basic SA agreed upon
(3)* **I → R:** AUTH	Responder identity verified by initiator; SA established
(e) Informational Exchange	
(1)* **I → R:** N/D	Error or status notification, or deletion

علائم اختصاری : I = آغازگر(Initiator) R = پاسخ‌دهنده (Responder)
* = رمزنگاری محموله بعد از سرآیند ISAKMP واقع می‌شود
AUTH = از مکانیسم اعتبارسنجی استفاده شده است.

مبادله‌های ISAKMP

ISAKMP یک چهارچوب برای مبادلهٔ پیام را فراهم می‌سازد که انواع محموله‌ها، عوامل تشکیل‌دهندهٔ آنها هستند. مشخصه پنج نوع مبادلهٔ پیش‌فرض را که بایستی حمایت گردند تعیین کرده است که در جدول ۴-۶ خلاصه شده‌اند. در این جدول، SA به یک محمولهٔ SA با محموله‌های نظیر Protocol و Transform اشاره می‌نماید.

Base Exchange اجازه می‌دهد تا مبادلهٔ کلید و مواد اعتبارسنجی با هم انتقال یابند. این امر تعداد تبادل‌ها را به حداقل می‌رساند ولی البته هویت‌ها مورد حفاظت قرار نمی‌گیرند. اولین دو پیام، cookieها را تولید کرده و یک SA با پروتکل و تبدیل‌های توافق شده را فراهم می‌آورد. هر دو طرف از یک nonce برای اطمینان‌بخشی در برابر حملات بازخوانی استفاده می‌کنند. آخرین دو پیام، مواد کلید و ID کاربران را مبادله کرده و از یک مکانیسم اعتبارسنجی برای تأئیدکلیدها، هویت‌ها و nonceهای دو پیام اول استفاده می‌کند.

Identity Protection Exchange برای محافظت از هویت کاربران، Base Exchange را بسط می‌دهد. دو پیام اول، SA را مستقر می‌کنند. دو پیام بعدی مبادلهٔ کلید را انجام می‌دهند و از nonceها برای محافظت جواب استفاده می‌شود. بمحض اینکه کلید اجلاس محاسبه گردید، دو طرف ارتباط به مبادلهٔ پیام‌های رمزنگاری شده که شامل اطلاعات اعتبارسنجی، همانند امضاءهای دیجیتال و احیاناً گواهی‌نامه‌های تأئیدکنندهٔ کلیدهای عمومی، است اقدام می‌کنند.

Authentication Only Exchange برای انجام اعتبارسنجی دوطرفه، بدون یک مبادلهٔ کلید بکار می‌رود. دو پیام اول، SA را مستقر می‌کنند. علاوه براین پاسخ‌دهنده از پیام دوم برای رساندن ID خود استفاده کرده و اعتبارسنجی را برای حفاظت پیام بکار می‌برد. آغازکننده، پیام سوم را ارسال نموده تا ID اعتبارسنجی شده را منتقل کند.

Aggressive Exchange تعداد مبادلات را به قیمت عدم حفاظت از هویت‌ها می‌نیمم می‌کند. در اولین پیام، آغازگر، یک SA با پروتکل پیشنهادی و تبدیل‌های ممکن را ارسال می‌کند. شروع‌کننده همچنین یک مبادلهٔ کلید را آغاز کرده و ID آن را فراهم می‌سازد. در پیام دوم، پاسخ‌دهنده، پذیرش این SA را با یک پروتکل و تبدیل بخصوص نشان‌داده، مبادلهٔ کلید را کامل ساخته و اطلاعات انتقال یافته را اعتبارسنجی می‌نماید. در پیام سوم، آغازکننده نتیجهٔ اعتبارسنجی بر روی اطلاعات قبلی، که با استفاده از کلید سّری باشتراک گذاشته شده رمزنگاری شده است را می‌فرستد.

از **Information Exchange** برای انتقال یکطرفهٔ اطلاعات برای مدیریت SA استفاده می‌شود.

۶-۷ منابع مطالعاتی

IPv6 و IPv4 بطور مفصل‌تری در [STAL04] پوشش داده شده‌اند. [CHEN98] بحث مفیدی در مورد طراحی IPSec دارد. [FRAN01] و [DORA03] پوشش تفصیلی‌تری از IPSec دارند.

CHEN98 Cheng, P., et al. "A Security Architecture for the Internet Protocol." *IBM Systems Journal*, Number 1, 1998.
DORA03 Doraswamy, N., and Harkins, D.*IPSec*.Upper Saddle River, NJ: Prentice Hall, 2003.
FRAN01 Frankel, S. *Demystifying the IPSec Puzzle*. Boston: Artech House, 2001.
STAL04 Stallings, W. *Computer Networking with Internet Protocols and Technology*. Upper Saddle River, NJ: Prentice Hall, 2004.

وب سایت‌های مفید

- **NIST IPSEC Project**: شامل مقالات، ارائهٔ مطالب و پیاده‌سازی‌های مرجع است.

۶-۸ واژه‌های کلیدی، سؤالات مرورکنندهٔ بحث و مسائل

واژه‌های کلیدی

نسخهٔ چهارم IP	IPv4	سرویس ضد- بازخوانی	anti-replay service
نسخهٔ ششم IP	IPv6	سرآیند اعتبارسنجی	authentication header (AH)
پروتکل تعیین کلید اُکلی	Oakley key determination protocol	کپسولی کردن محمولهٔ امنیتی	encapsulating security payload (ESP)
حملهٔ بازخوانی	replay attack	پروتکل اتحاد امنیتی و مدیریت کلید اینترنت	Internet Security Association and Key Management Protocol (ISAKMP)
اتحاد امنیتی	security association (SA)	امنیت IP	IP Security (IPSec)
مُود حمل‌ونقل	transport mode		
مُود تونل	tunnel mode		

سؤالات مرورکننده بحث

۶-۱ مثال‌هائی از کاربرد IPSec را بیان کنید.

۶-۲ چه سرویس‌هائی بتوسط IPSec فراهم می‌گردند؟

۶-۳ چه پارامترهائی یک SA را معرفی و چه پارامترهائی ماهیت یک SA بخصوص را تعیین می‌کنند؟

۶-۴ تفاوت بین مُود حمل‌ونقل و مُود تونل چیست؟

۶-۵ حملهٔ بازخوانی کدام است؟

۶-۶ چرا ESP دارای یک میدان padding است؟

۶-۷ روش‌های اصلی ترکیب SAها کدامند؟

۶-۸ نقش پروتکل‌های تعیین کلید Oakley و ISAKMP در IPSec چیست؟

مسائل

۶-۱ در بحث پردازش AH، خاطرنشان شده بود که تمام میدان‌های سرآیند IP در محاسبات MAC وارد نمی‌شوند.

الف- برای هریک از میدان‌های سرآیند IPv4، نشان دهید که آیا آن میدان تغییرناپذیر، تغییرپذیر ولی قابل پیش‌بینی و یا تغییرپذیر (که قبل از محاسبات ICV باید صفر شوند) است.

ب- همین کار را برای IPv6 انجام دهید.

ج - همین کار را برای سرآیندهای الحاقی IPv6 انجام دهید.

در هر مورد دلیل خود برای هر میدان را توجیه کنید.

۶-۲ وقتی از مُود تونل استفاده می‌شود، یک سرآیند IP بیرونی ساخته می‌شود. برای هر دو نسخهٔ IPv4 و IPv6 رابطهٔ بین میدان‌های سرآیند IP بیرونی و هر سرآیند الحاقی در بستهٔ بیرونی را با میدان و یا سرآیند الحاقی بستهٔ درونی نشان دهید. یعنی نشان دهید که کدام اندازه‌های بیرونی از مقادیر درونی مشتق شده و کدام اندازه‌های بیرونی مستقل از مقادیر درونی ساخته می‌شوند.

۶-۳ رمزنگاری و اعتبارسنجی سر- به- سر بین دو میزبان کاری مطلوب است. شکل‌هائی شبیه به شکل‌های ۶-۶ و ۶-۹ کشیده که نشان دهد:

الف- مجاورت مُودهای حمل‌ونقل با رمزنگاری قبل از اعتبارسنجی.

ب- یک SA حمل‌ونقل در داخل یک SA تونل که در آن رمزنگاری قبل از اعتبارسنجی انجام شود.

ج - یک SA حمل‌ونقل در داخل یک SA تونل که در آن اعتبارسنجی قبل از رمزنگاری انجام شود.

۶-۴ اسناد معماری IPSec بیان می‌کنند که وقتی دو SA مُود حمل‌ونقل با هم ترکیب شده تا هم پروتکل AH و هم پروتکل ESP را روی یک جریان سر- به- سر ایجاد کنند تنها یک روش مناسب بنظر می‌رسد که آنهم اجرای پروتکل ESP قبل از اجرای AH است. چرا این روش پیشنهاد شده و اعتبارسنجی قبل از رمزنگاری پیشنهاد نگردیده است؟

۶-۵ **الف**- کدامیک از انواع مبادلات ISAKMP (جدول ۴-۶) نظیر مبادلهٔ کلید aggressive Oakley است(شکل ۱۱-۶)؟

ب- برای مبادلهٔ کلید aggressive Oakley نشان دهید که کدام پارامترها در هر پیام، در کدام نوع محمولهٔ ISAKMP حمل می‌شوند.

ضمیمهٔ ۶- الف عملیات بین‌شبکه‌ای و پروتکل‌های اینترنت

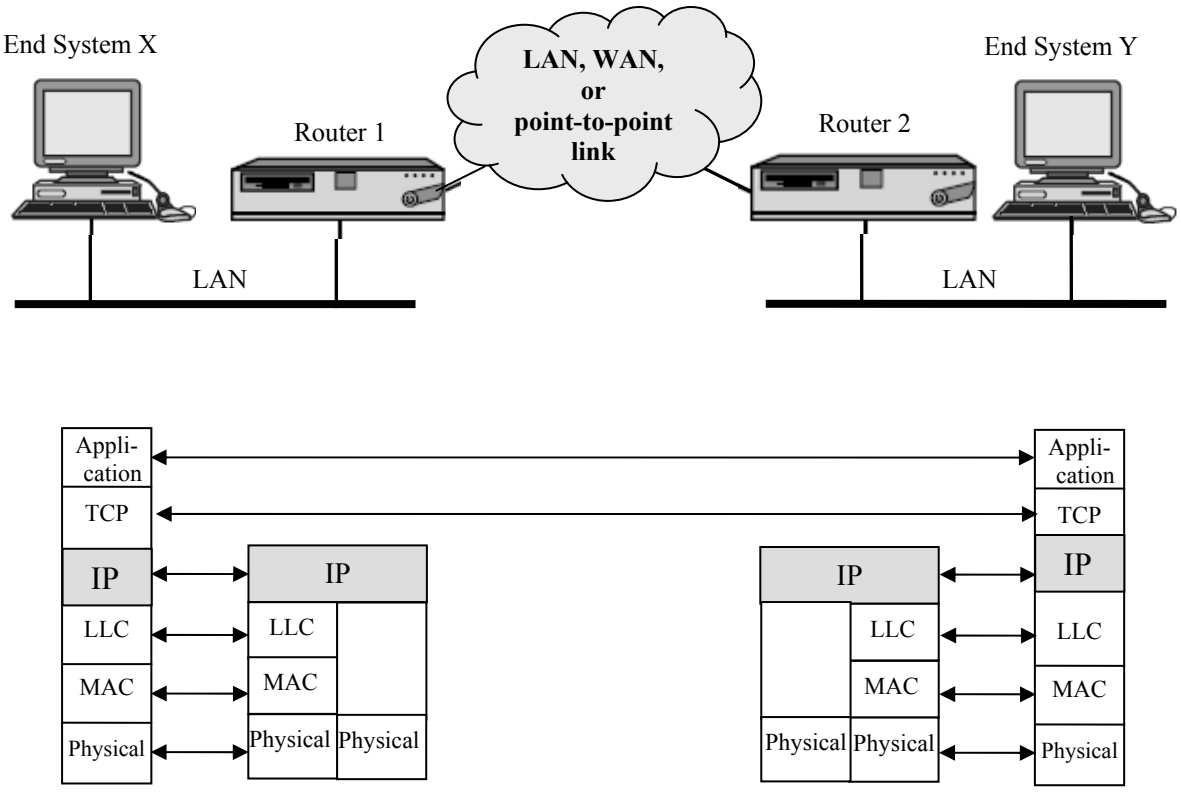
این ضمیمه، مروری بر پروتکل‌های بین‌شبکه‌ای دارد. بحث را با بیان خلاصهٔ نقش یک پروتکل بین‌شبکه‌ای در فراهم‌آوردن عملیات بین‌شبکه‌ای آغاز می‌کنیم. آنگاه دو پروتکل بین‌شبکه‌ای اصلی یعنی IPv4 و IPv6 را معرفی می‌نمائیم.

نقش یک پروتکل بین‌شبکه‌ای

یک پروتکل بین‌شبکه‌ای (IP)، نیازهای مربوط به اتصال دو سیستم انتهائی در عرض شبکه‌های متعدد را برآورده می‌سازد. برای این منظور، IP در هر سیستم انتهائی و همچنین در مسیریاب‌ها که دستگاه‌هائی برای ایجاد اتصال بین شبکه‌ها هستند پیاده‌سازی می‌شود. داده‌های لایهٔ بالاتر در یک سیستم انتهائی، برای انتقال، در یک واحد پروتکلی دیتای IP (IP PDU) کپسولی می‌شود. این PDU آنگاه از یک یا چند شبکه و همچنین مسیریاب‌های ارتباط‌دهنده عبور کرده تا به سیستم انتهائی مقصد برسد.

مسیریاب بایستی بتواند خود را با تنوعی که بین شبکه‌های مختلف وجود دارد وفق دهد. برخی از تفاوت‌های بین شبکه‌ها بقرار زیر است:

- **روش‌های آدرس‌دهی**: شبکه‌ها ممکن است روش‌های متفاوتی را برای تخصیص آدرس‌ها به دستگاه‌ها بکار گیرند. مثلاً یک IEEE802 LAN برای هر دستگاه شبکه یک آدرس ۱۶- بیتی و یا ۴۸- بیتی را بکار می‌برد. یک شبکهٔ سوئیچینگ بسته‌ای X.25 از آدرس‌های ۱۲ رقمی اعشاری (۴ بیت برای هر رقم و در جمع ۴۸ بیت) استفاده می‌کند. نوعی آدرس‌دهی عمومی بعلاوهٔ فهرستی از آدرس‌ها بایستی فراهم شود.
- **اندازهٔ ماکزیمم بسته‌ها**: بسته‌های یک شبکه ممکن است برای عبور از شبکه دیگر نیاز به قطعه‌قطعه شدن داشته باشند که این عمل را fragmentation گویند. بعنوان مثال Ethernet اندازهٔ ماکزیمم بسته‌ها را ۱,۵۰۰ بایت قرار داده است در حالی‌که اندازهٔ ماکزیمم بسته‌ها در شبکه‌های X.25 برابر ۱,۰۰۰ بایت است. یک بسته که روی سیستم Ethernet انتقال یافته و بمنظور عبور به یک شبکهٔ X.25 از یک مسیریاب عبور می‌کند، ممکن است نیاز به تبدیل به دو بستهٔ کوچک‌تر داشته باشد.
- **واسط‌ها**: واسط‌های (interfaces) سخت‌افزاری و نرم‌افزاری در شبکه‌های مختلف متفاوت‌اند. ماهیت عمل مسیریاب بایستی مستقل از چنین تفاوت‌هائی باشد.
- **قابلیت اعتماد**: سرویس‌های مختلف شبکه می‌توانند از یک مدار مجازی سر- به- سر تا یک سرویس غیرقابل اعتماد متغیر باشند. عمل مسیریاب‌ها بایستی مستقل از فرض مورد اعتماد بودن شبکه و یا خلاف آن باشد.



شکل ۱۳-۶ پیکربندی برای مثال TCP/IP

عمل مسیریاب همانطور که در شکل ۱۳-۶ نشان داده شده است، وابسته به یک پروتکل بین شبکه‌ای است. در این مثال، پروتکل اینترنت (IP) از مجموعهٔ پروتکلی TCP/IP این عمل را انجام می‌دهد. IP بایستی در تمام سیستم‌های انتهائی، روی تمام شبکه‌ها و همچنین در مسیریاب‌ها، تعبیه شود. علاوه برآن، هر سیستم انتهائی بایستی پروتکل‌های سازگاری در بالای IP داشته تا ارتباط بصورت موفق انجام پذیرد. مسیریاب‌های بین راه تنها کافی است که تا سطح IP را حمایت نمایند.

انتقال یک بلوک دیتا از سیستم انتهائی X به سیستم انتهائی Y در شکل ۱۳-۶ را در نظر بگیرید. لایهٔ IP در X بلوک‌های دیتا که بایستی برای Y ارسال شوند را از لایهٔ TCP سیستم X تحویل می‌گیرد. لایهٔ IP، یک سرآیند که آدرس جهانی Y را مشخص می‌کند به دیتا اضافه می‌نماید. این آدرس دارای دو قسمت شناسهٔ شبکه و شناسهٔ سیستم انتهائی است. اجازه دهید که این بلوک را یک بستهٔ IP بنامیم. در مرحلهٔ بعد IP درمی‌یابد که مقصد (Y) روی زیرشبکهٔ دیگری است. بنابراین اولین قدم این است که بسته را به یک مسیریاب، که در این مورد مسیریاب 1 است، ارسال کند. برای انجام این امر، IP واحد دیتای خود را با اطلاعات کامل آدرس‌دهی به لایهٔ LLC در قسمت پائین‌تر می‌دهد. LLC واحد دیتای LLC PDU را خلق کرده که در مرحلهٔ بعد به لایهٔ MAC داده می‌شود. لایهٔ MAC یک بستهٔ MAC که سرآیند آن شامل آدرس مسیریاب 1 است را می‌سازد.

سپس بسته از درون شبکهٔ LAN به مسیریاب 1 می‌رود. مسیریاب، سرآیندها و ته‌آیندهای بسته و LLC را کنده و سرآیند IP را بررسی می‌کند تا مقصد نهائی دیتا که در این مورد Y است را تعیین کند. مسیریاب در اینجا بایستی نسبت به مسیریابی تصمیم‌گیری نماید. دو امکان وجود دارد:

۱- سیستم انتهائی مقصد (Y) مستقیماً به یکی از زیرشبکه‌هائی متصل است که مسیریاب نیز در آنها قرار دارد.
۲- برای رسیدن به مقصد، بایستی از یک یا چند مسیریاب دیگر نیز عبور کرد.

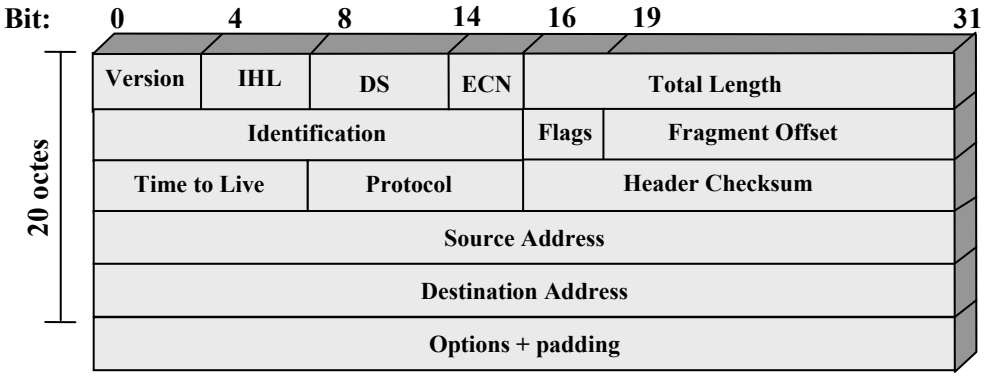
در این مثال، بسته بایستی قبل از رسیدن به مقصد از مسیریاب 2 عبور کند. بنابراین مسیریاب 1 بستهٔ IP را از طریق شبکهٔ میانی به مسیریاب 2 می‌فرستد. برای این مقصود، پروتکل‌های آن شبکه بکار گرفته می‌شوند. مثلاً اگر شبکهٔ میانی یک شبکهٔ X.25 است، واحد دیتای IP، به همراه اطلاعات آدرس‌دهی مرتبط برای رسیدن به مسیریاب 2، در یک بستهٔ X.25 پیچیده می‌شود. وقتی این بسته به مسیریاب 2 وارد می‌شود، سرآیند بسته کنده می‌شود. مسیریاب تعیین می‌کند که این بسته IP به مقصد Y است که مستقیماً روی زیرشبکه‌ای که مسیریاب به آن متصل است قرار دارد. در نتیجه مسیریاب یک بسته با آدرس مقصد Y را خلق کرده و آن را روی شبکهٔ LAN می‌فرستد. نهایتاً دیتا وارد Y میگردد که در آنجا سرآیندها و ته‌آیندهای بسته، LLC و اینترنت می‌توانند از آن جدا شوند.

سرویسی که بتوسط IP فراهم می‌شود یک سرویس غیرقابل اعتماد است. یعنی IP تضمین نمی‌کند که تمام دیتا به مقصد تحویل شده و یا اینکه دیتا با نظم اولیه وارد مقصد گردد. این وظیفهٔ یک لایهٔ بالاتر، در این مورد TCP، است که هر خطائی را که ممکن است واقع شود تصحیح نماید. این روش انعطاف‌پذیری زیادی را به ارمغان می‌آورد. چون تحویل بسته‌ها تضمین شده نیست، نیازی به اعتماد به زیرشبکه‌ها وجود ندارد. در نتیجه پروتکل با هر ترکیبی از انواع زیرشبکه‌ها کار می‌کند. چون تضمینی برای تحویل منظم بسته‌ها وجود ندارد، بسته‌های پشت سرهم می‌توانند مسیرهای متفاوتی را از درون اینترنت طی کنند. این امر به پروتکل اجازه می‌دهد تا در صورت مواجهه با تراکم و یا خرابی در شبکه، مسیر بستهٔ دیتا را عوض کند.

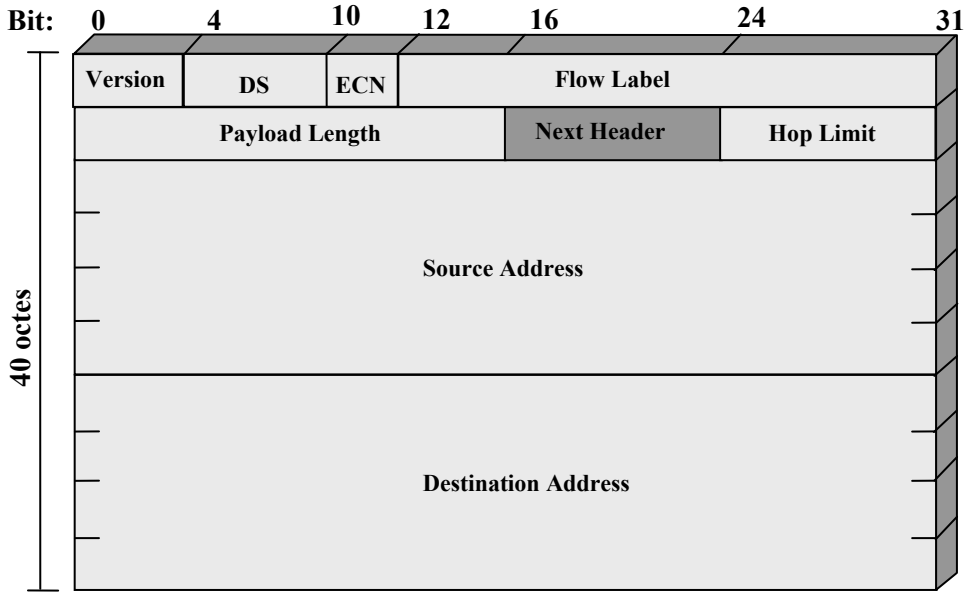
IPv4

برای دهه‌های متوالی سنگ‌بنای معماری پروتکل TCP/IP، پروتکل اینترنت (IP) نسخهٔ ۴ بوده است. شکل ۱۴- ۶الف فرمت سرآیند IP را نشان می‌دهد که حداقل دارای ۲۰ اُکتت و یا ۱۶۰ بیت است. میدان‌ها بقرار زیراند:

- **Version (4 bits)**: نشان‌دهندهٔ شمارهٔ نسخهٔ پروتکل است تا بعداً بتوان نسخ تکامل‌یافته‌تر را با شمارهٔ جدیدی نشان داد. در اینجا اندازهٔ آن ۴ است.
- **Internet Header Length (IHL) (4 bits)**: طول سرآیند بر حسب کلمات ۳۲- بیتی است. حداقل آن ۵ است که طول می‌نیمم سرآیند یعنی ۲۰ اُکتت را نشان می‌دهد.



(الف) سرآیند IPv4



(ب) سرآیند IPv6

DS = Differentiated services field
ECN = Explicit congestion notification field

توجه : میدان هشت بیتی DS/ECN قبلاً در سرآیند IPv4 به نام میدان Type of Service و درسرآیند IPv6 به نام Traffic Class خوانده می‌شد

شکل ۱۴-۶ سرآیندهای IP

- **DS/ECN (8 bits)**: قبل از معرفی سرویس‌های مشتق شده، این میدان بنام میدان **Type of Service** خوانده می‌شد و قابلیت اعتماد، اولویت، تأخیر و پارامترهای مربوط به توان عملیاتی را مشخص می‌نمود. این تعبیر اکنون کنار گذاشته شده است. اولین ۶ بیت میدان TOS اکنون با نام میدان DS (Differentiated Services) خوانده می‌شود. ۲ بیت باقیمانده برای میدان ECN (Explicit Congestion Notification) رزرو شده‌اند.
- **Total Length (16 bits)**: طول کلّ بستهٔ IP بر حسب اُکتت.
- **Identification (16 bits)**: یک شمارهٔ ردیف که به همراه آدرس منبع، آدرس مقصد و پروتکل کاربر یک بسته را بطور یکتا مشخص می‌سازد. بنابراین این عدد برای آدرس منبع بسته، آدرس مقصد بسته و پروتکل کاربر در خلال مدتی که بسته در اینترنت می‌ماند بایستی یکتا باشد.
- **Flags (3 bits)**: در حال حاضر تنها دو بیت آن تعریف شده است. وقتی بسته‌ای قطعه‌قطعه می‌گردد (fragmentation)، بیت More نشان می‌دهد که آیا این بسته آخرین قطعهٔ بستهٔ اولیه است. بیت Don't Fragment اگر set باشد به مفهوم این است که این بسته نبایستی قطعه‌قطعه گردد. این بیت زمانی ممکن است مفید واقع شود که بدانیم مقصد قابلیت دوباره سرهم‌کردن (reassembly) قطعه‌ها را نخواهد داشت. از طرفی وقتی این بیت set باشد، اگر اندازهٔ بسته از اندازهٔ ماکزیمم مجاز در زیرشبکه‌های مسیر بیشتر شود، بسته معدوم خواهد شد. بنابراین اگر این بیت set است، عاقلانه‌تر است که از مسیریابی منبع استفاده شود تا از عبور بسته از زیرشبکه‌هائی که اندازهٔ ماکزیمم بسته در آنها کوچک است اجتناب گردد.
- **Fragment Offset (13 bits)**: نشان می‌دهد که این قطعه به کجای بستهٔ اصلی تعلق دارد و اندازهٔ آن بر حسب واحدهای ۶۴- بیتی مشخص می‌گردد. این امر لازم می‌دارد که قطعه‌ها بجز قطعهٔ آخر بایستی دارای میدان دیتائی باشند که طول آن مضربی از ۶۴ بیت باشد.
- **Time to Live (8 bits)**: تعیین می‌کند که این بسته برای چه مدتی، بر حسب ثانیه، مجاز به ماندن در اینترنت است. هر مسیریابی که یک بسته را پردازش می‌کند بایستی TTL را حداقل 1 واحد کاهش دهد. بنابراین TTL تا حدودی شبیه شمارش‌گر پرش‌ها (hops) در مسیر است.
- **Protocol (8 bits)**: پروتکل لایهٔ بالاترکه بایستی میدان دیتا در مقصد را تحویل بگیرد نشان می‌دهد. بنابراین این میدان نوع سرآیند بعدی در بسته، بعد از سرآیند IP، را تعیین می‌کند.
- **Header Checksum (16 bits)**: یک کُد تشخیص خطاست که تنها به سرآیند اعمال می‌گردد. چون برخی از میدان‌های سرآیند ممکن است در زمان ترانزیت تغییر کنند (مثلاً Time to Live یا میدان‌های مربوط به سِگمنت)، این کُد در هر مسیریاب کنترل و مجدداً محاسبه می‌گردد. میدان checksum یک جمع متمم یک ۱۶- بیتی از تمام کلمات ۱۶- بیتی سرآیند است. در محاسبات، میدان checksum در ابتدا برابر صفر قرار داده می‌شود.
- **Source Address (32 bits)**: بیت‌های کُد شده‌ای است که بمنظور تعیین یک شبکه و سیستم انتهائی متصل به آن بکار می‌رود (۷ و ۲۴ بیت، ۱۴ و ۱۶ بیت یا ۲۱ و ۸ بیت).
- **Destination Address (32 bits)**: همان مشخصه‌های آدرس منبع را داراست.

- **Options (variable)**: مقوله‌های اختیاری تقاضاشده بتوسط کاربر ارسال‌کننده را نشان می‌دهد. اینها می‌توانند شامل برچسب امنیتی، مسیریابی منبع، ثبت مسیریابی و برچسب زمانی باشند.
- **Padding (variable)**: برای تکمیل طول سرآیند بسته به مضربی از ۳۲ بیت بکار می‌رود.

IPv6

در سال ۱۹۹۵ میلادی، IETF (Internet Engineering Task Force) که استانداردهای پروتکلی اینترنت را فراهم می‌آورد، مشخصه‌ای برای IP نسل بعد را انتشار داد که در آن زمان به IPng معروف شد. این مشخصه در سال ۱۹۹۶ بصورت استاندارد درآمده و IPv6 نام گرفت. IPv6 نسبت به IP فعلی (IPv4)، تعدادی عملیات اضافی در بر دارد که بمنظور کارآئی بیشتر در شبکه‌های پرسرعت امروزی و اختلاط جریان‌های دیتا که شامل گرافیک و ویدئو بوده و پیوسته خواستاران بیشتری دارد طراحی شده است. ولی انگیزهٔ اصلی در فراهم آوردن پروتکل جدید، نیاز به آدرس‌های بیشتر بود. IPv4 از یک آدرس ۳۲- بیتی برای مشخص کردن منبع و مقصد استفاده می‌کند. با رشد انفجاری اینترنت و شبکه‌های خصوصی متصل به آن، این طول آدرس برای برآوردن نیازهای تمام سیستم‌های نیازمند به آدرس کافی نیست. همانطور که شکل ۱۴-۶ب نشان می‌دهد، IPv6 شامل آدرس‌های ۱۲۸- بیتی منبع و مقصد در میدان آدرس خود است. در نهایت تمام پیاده‌سازی‌های TCP/IP از IP کنونی به سمت IPv6 سوق داده خواهند شد که البته این امر ممکن است سال‌ها و بلکه ده‌ها سال بطول انجامد.

سرآیند IPv6

سرآیند IPv6 دارای طول ثابت ۴۰ اُکتت است که شامل میدان‌های زیر است (شکل ۱۴-۶ب):

- **Version (4 bits)**: شمارهٔ نسخهٔ پروتکل اینترنت. این اندازه برابر ۶ است.
- **DS/ECN (8 bits)**: قبل از معرفی سرویس‌های مشتق شده، این میدان بنام میدان **Traffic Class** خوانده می‌شد و برای استفادهٔ گره‌های آغازگر و/ یا مسیریاب‌های جلوبرنده بمنظور تشخیص و تمایز بین کلاس‌های مختلف و یا اولویت‌های مختلف بسته‌های IPv6 رزرو شده بود. اولین ۶ بیت میدان Traffic Class اکنون بنام میدان DS (Differentiated Services) خوانده می‌شود. دو بیت باقیمانده برای میدان ECN (Explicit Congestion Notification) رزرو شده‌اند.
- **Flow Label (20 bits)**: می‌تواند بتوسط یک میزبان برای تعیین آن بسته‌هائی بکار رودکه سرویس‌های خاصی را از مسیریاب‌های بین راه طلب می‌کنند. تعیین برچسب برای جریان ترافیک می‌تواند در رزرو کردن منبع و پردازش بلادرنگ ترافیک مؤثر باشد.
- **Payload Length (16 bits)**: اندازهٔ بقیهٔ بسته IPv6 بر حسب اُکتت که پس از سرآیند قرار می‌گیرد. بعبارت دیگر، این طول کل تمام سرآیندهای الحاقی باضافهٔ طول PDU سطح حمل‌ونقل است.
- **Next Header (8 bits)**: نوع سرآیندی که بلافاصله بعد از سرآیند IPv6 قرار می‌گیرد را تعیین می‌کند. این یا یک سرآیند الحاقی IPv6 و یا یک سرآیند لایهٔ بالاتر مانند TCP و یا UDP است.

- **Hop Limit (8 bits)**: تعداد باقیماندهٔ پرش‌های مجاز (hops) این بسته است. حد پرش بتوسط منبع به یک مقدار ماکزیمم دلخواه تنظیم شده و پس از عبور از هر گره که بسته را جلو می‌راند، یک واحد کم می‌شود. در صورتی که Hop Limit به صفر تقلیل یابد، این بسته معدوم خواهد شد.
- **Source Address (128 bits)**: آدرس منبع آغازگر این بسته است.
- **Destination Address (128 bits)**: آدرس گیرندهٔ بستهٔ مورد نظر است. اگر یک سرآیند الحاقی Routing وجود داشته باشد، این آدرس ممکن است آدرس مقصد نهائی نباشد.

اگرچه سرآیند IPv6 طولانی‌تر از بخش اجباری سرآیند IPv4 است (۴۰ اُکتت در برابر ۲۰ اُکتت)، ولی دارای میدان‌های کمتری است (۸ در برابر ۱۲). بنابراین مسیریاب‌ها برای هر سرآیند پردازش کمتری انجام می‌دهند که این امر مسیریابی را سرعت می‌بخشد.

سرآیندهای الحاقی IPv6

یک بستهٔ IPv6، شامل سرآیند IPv6 که هم اکنون تشریح گردید، بعلاوهٔ هیچ یا چند سرآیند الحاقی دیگر است. خارج از IPSec، سرآیندهای الحاقی زیر تعریف شده‌اند:

- **Hop-by-Hop Options Header**: موارد اختیاری بخصوصی را تعریف می‌کند که بتوسط پردازش hop-by-hop مورد نیاز است.
- **Routing Header**: مسیریابی گسترده‌تری همانند مسیریابی منبع در IPv4 را فراهم می‌آورد.
- **Fragment Header**: اطلاعات fragmentation و reassembly را شامل می‌شود.
- **Authentication Header**: مکانیسم اعتبارسنجی و سنجش صحت بسته را فراهم می‌کند.
- **Encapsulating Security Payload Header**: خصوصی ماندن بسته را تأمین می‌کند.
- **Destination Options Header**: اطلاعات اختیاری که بتوسط گرهٔ مقصد مورد مداقّه قرار می‌گیرد را فراهم می‌سازد.

استاندارد IPv6 توصیه می‌کند که وقتی سرآیندهای الحاقی متعددی بکار گرفته می‌شوند، سرآیندهای IPv6 دارای نظم زیر باشند:

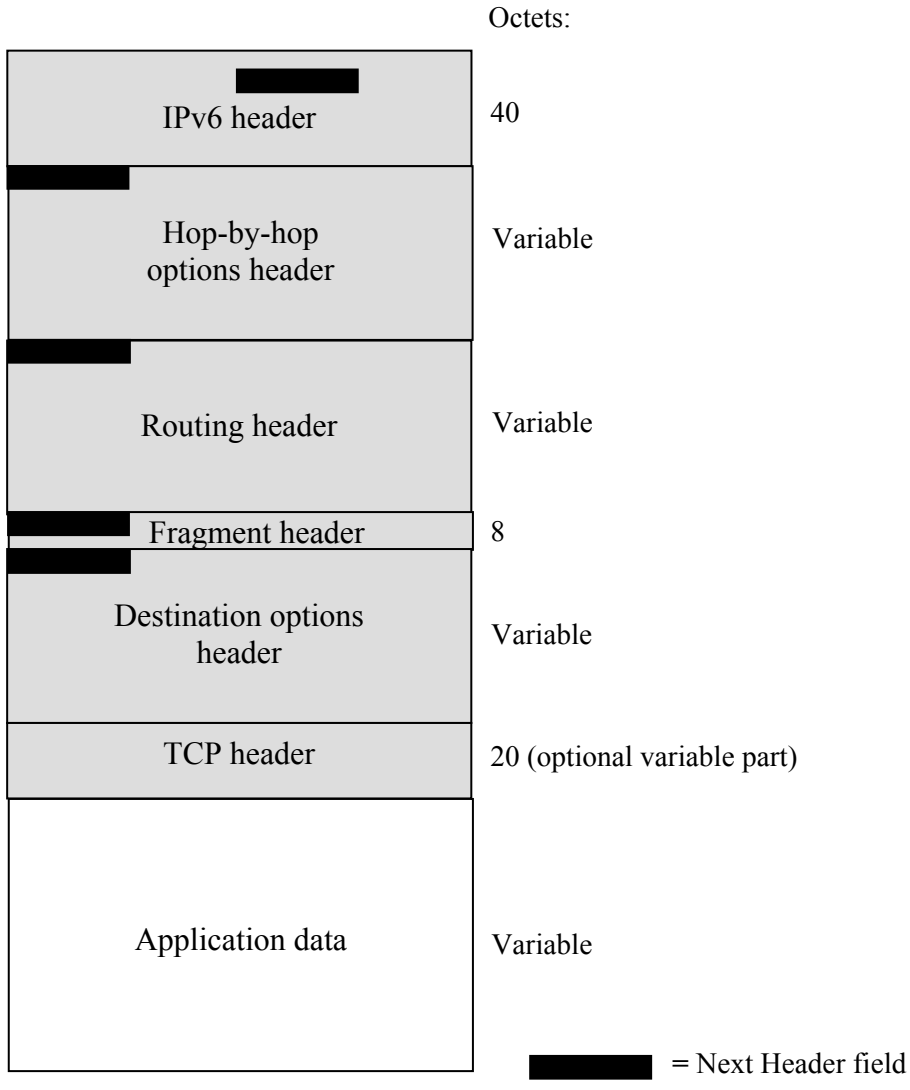
۱- سرآیند IPv6: اجباری و همیشه در ابتدا
۲- سرآیند Hop-by-Hop Options
۳- سرآیند Destination Options: برای موارد اختیاری که بایستی بتوسط اولین مقصدی که در میدان IPv6 Destination Address مشخص شده و مقصدهائی که بعد از آن در سرآیند Routing آمده است، پردازش شود
۴- سرآیند Routing
۵- سرآیند Fragment

۶- سرآیند Authentication

۷- سرآیند Encapsulating Security Payload

۸- سرآیند Destination Options: برای موارد اختیاری که تنها بتوسط مقصد انتهائی بسته پردازش می‌شود

شکل ۱۵-۶ مثالی از یک بستهٔ IP را نشان می‌دهد که شامل موردی از هر سرآیند غیرامنیتی است. توجه کنید که سرآیند IPv6 و هر سرآیند الحاقی، دارای یک میدان Next Header است. این میدان نوع سرآیندی را که بلافاصله بعد از آن قرار دارد، تعیین می‌کند. در غیر اینصورت این میدان شامل شناسهٔ پروتکل لایهٔ بالاتری است که از IPv6 استفاده می‌کند. در شکل، پروتکل لایهٔ بالاتر TCP بوده و بنابراین دیتای لایهٔ بالاتر که بتوسط بستهٔ IPv6 حمل می‌گردد شامل یک سرآیند TCP است که به دنبال آن یک بلوک از دیتای کاربردی قرار دارد.



شکل ۱۵-۶ بستهٔ IPv6 با سرآیندهای الحاقی (شامل یک سِگمنت TCP)

سرآیند Hop-by-Hop Options: اطلاعات اختیاری را حمل می‌کند که اگر وجود داشته باشد بایستی بتوسط هر مسیریاب در طول مسیر مورد بازبینی قرار گیرد. این سرآیند شامل میدان‌های زیر است:

- **Header Extension Length (8 bits)**: طول این سرآیند بر حسب واحدهای ۶۴- بیتی را مشخص می‌کند که شامل اولین ۶۴ بیت نیست.
- **Options**: شامل یک یا چند مورد اختیاری است. هر مورد اختیاری از سه زیرمیدان تشکیل می‌شود: یک tag که نمایش‌دهندهٔ نوع مورد اختیاری، یک طول و یک اندازه است.

تا کنون تنها یک مورد اختیاری تعریف شده است: مورد Jumbo Payload که برای ارسال بسته‌های IPv6 با محموله‌هائی طولانی‌تر از ۶۵,۵۳۵ = ۱-$۲^{۳۲}$ اُکتت بکار می‌رود. میدان Option Data در اینجا ۳۲ بیت طول داشته و اندازهٔ طول بسته را، بجز سرآیند IPv6، بر حسب اُکتت مشخص می‌کند. برای هر بسته، میدان Payload Length در سرآیند IPv6 بایستی برابر صفر قرار داده شود و بایستی هیچ سرآیند Fragment وجود نداشته باشد. با این اختیار، IPv6 بسته‌های دیتائی تا طول چهار میلیارد اُکتت را حمایت می‌کند. این امر انتقال بسته‌های بزرگ ویدئو را تسهیل نموده و IPv6 را قادر می‌سازد تا از ظرفیت موجود محیط انتقال حداکثر استفاده را بنماید.

سرآیند Routing شامل لیستی از یک یا چند گرهٔ میانی است که بایستی در مسیر بسته تا مقصد ملاقات شوند. تمام سرآیندهای مسیریابی با یک بلوک ۳۲- بیتی که شامل چهار میدان ۸- بیتی است آغاز می‌شوند و به دنبال آن داده‌های مسیریابی متعلق به روش مسیریابی خاصی قرار می‌گیرد. چهار میدان ۸ - بیتی عبارت از Next Header، Extension Length Header و دو میدان زیراند:

- **Routing Type**: یک سرآیند Routing خاص را تعیین می‌نماید. اگر مسیریابی اندازهٔ Routing Type را شناسائی نکند، بایستی بسته را معدوم سازد.
- **Segments Left**: تعداد صریح گره‌های میانی که بایستی قبل از رسیدن به مقصد نهائی ملاقات شوند.

علاوه براین تعریف سرآیند عمومی، مشخصه‌های IPv6 سرآیند Type 0 Routing را تعریف می‌کند. وقتی از سرآیند Type 0 Routing استفاده می‌شود، گرهٔ منبع، نهائی‌ترین آدرس مقصد را در سرآیند IPv6 قرار نمی‌دهد. در عوض آن آدرس، آخرین آدرس لیست شده در سرآیند Routing بوده و سرآیند IPv6 شامل آدرس مقصد اولین مسیریاب موجود در مسیر خواهد بود. سرآیند Routing تا زمانی که بسته به گرهٔ مقصد سرآیند IPv6 نرسد، بررسی نخواهد شد. در آن نقطه، محتویات سرآیند IPv6 و Routing به روزرسانی شده و بسته مجدداً به جلو رانده می‌شود. به روزرسانی شامل قرار دادن آدرس بعدی در سرآیند IPv6 و کاهش دادن میدان Segments Left در سرآیند Routing است.

در IPv6 لازم است تا یک گره IPv6، مسیرها را در یک بستهٔ دریافت شدهٔ شامل سرآیند Routing معکوس کرده تا بسته بتواند به فرستنده برگردد.

سرآیند Fragment وقتی بتوسط یک منبع بکار می‌رود که قطعه قطعه کردن دیتا مورد نیاز باشد. در IPv6 قطعه‌قطعه کردن دیتا تنها می‌تواند در گره‌های مبدأ انجام شود و نه بتوسط مسیریاب‌هائی که در مسیر تحویل بسته واقع‌اند. برای استفادهٔ کامل از امتیازات محیط بین شبکه‌ها، یک گره بایستی یک الگوریتم کشف مسیر را به اجرا گذاشته که او را قادر

می‌سازد تا کوچک‌ترین واحد انتقال ماکزیمم (MTU) که بتوسط هر زیرشبکه مسیر حمایت می‌شود را پیدا کند. بعبارت دیگر، الگوریتم کشف مسیر، یک گره را قادر می‌سازد تا MTU زیرشبکهٔ «گلوگاه» در روی مسیر را کشف نماید. با این معلومات، گرهٔ منبع، در صورت نیاز، برای هر آدرس مقصد داده شده دیتا را قطعه‌قطعه خواهد کرد. در غیراینصورت منبع بایستی تمام بسته‌ها را به ۱,۲۸۰ اُکتت محدود سازد که حداقل MTU ای است که بایستی بتوسط هر زیرشبکه حمایت گردد.

علاوه بر میدان Next Header، سرآیند Fragment شامل میدان‌های زیر است:

- **Fragment Offset (13 bits)**: نشان می‌دهد که محمولهٔ این fragment به کجای بستهٔ اصلی تعلق دارد. این برحسب واحدهای ۶۴- بیتی محاسبه می‌شود. این امر بطور ضمنی شامل این مطلب است که قطعه‌ها (بغیر از آخرین قطعه) بایستی دارای میدان دیتائی باشند که مضربی از ۶۴ بیت است.
- **Res (2 bits)**: برای مصارف آتی رزرو شده است.
- **M Flag (1 bit)**: اگر مساوی 1 باشد یعنی قطعه‌های دیگری وجود دارند و اگر مساوی 0 باشد یعنی آخرین قطعه است.
- **Identification (32 bits)**: هدف آن شناسائی بستهٔ اولیه بطور یکتاست. این شناسهٔ بایستی برای آدرس منبع و آدرس مقصد بسته، برای مدت زمانی که بسته در اینترنت خواهد ماند، یکتا باشد. تمام قطعه‌هائی که دارای شناسهٔ یکسان، آدرس منبع یکسان و آدرس مقصد یکسان می‌باشند دوباره بهم پیوسته و بستهٔ اولیه را درست خواهند کرد.

سرآیند Destination Options اطلاعات اختیاری را حمل می‌کنند که اگر وجود داشته باشد تنها بتوسط گرهٔ مقصد مورد بررسی قرار می‌گیرد. فرمت این بستهٔ سرآیند همانند فرمت سرآیند Hop-by-Hop Options است.

فصل ۷

امنیت WEB

۷-۱ ملاحظات امنیت وِب

تهدیدهای امنیتی وِب
روش‌های برخورد با امنیت ترافیک وِب

۷-۲ لایهٔ سوکت اَمن و امنیت لایهٔ حمل و نقل (SSL/TLS)

معماری SSL
پروتکل SSL Record
پروتکل Change Cipher Spec
پروتکل Alert
پروتکل Handshake
محاسبات رمزنگاری
امنیت لایهٔ حمل و نقل

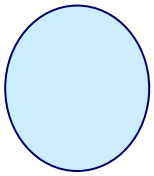
۷-۳ معاملهٔ الکترونیکی اَمن (SET)

مروری بر SET
امضاء دوگانه
عملیات پرداخت

۷-۴ منابع مطالعاتی

۷-۵ واژه‌های کلیدی، سؤالات مرورکنندهٔ بحث و مسائل

واژه‌های کلیدی
سؤالات مرورکنندهٔ بحث
مسائل



قریباً تمام کسب و کارها، بیشتر سازمان‌های دولتی و شمار بسیاری از افراد، امروزه صاحب وِب سایت هستند. تعداد افراد و شرکت‌هائی که دسترسی به اینترنت دارند بسرعت افزایش یافته و تمام آنها به مرورگرهای گرافیکی وِب مجهزاند. در همین رابطه بازرگانان علاقه‌مندند تا بمنظور تجارت الکترونیک، تسهیلاتی را روی وِب فراهم نمایند. اما واقعیت این است که اینترنت و وِب شدیداً در مقابل دست‌یابی‌های غیرمجاز از انواع مختلف آسیب‌پذیرند. همین‌طور که کار و پیشه به این امر وقوف بیشتری پیدا می‌کند، تقاضا برای سرویس‌های امن وِب بیشتر می‌شود.

مقولهٔ امنیت وِب بسیار وسیع بوده و خود می‌تواند به‌تنهائی موضوع یک کتاب باشد. در این فصل ابتدا نیازهای عمومی امنیت وِب را بررسی کرده و آنگاه روی دو روش استاندارد SSL/TLS و SET که در بحث تجارت وِب اهمیت فزاینده‌ای پیدا کرده‌اند، متمرکز می‌شویم.

۷-۱ ملاحظات امنیت وِب

World Wide Web یا تار جهان‌گستر اصولاً یک کاربرد کلاینت/سِرور بوده که روی اینترنت و اینترانت‌های TCP/IP کار می‌کند. با چنین نگرشی، ابزارهای امنیتی و روش‌هائی که تا کنون در این کتاب مورد بحث قرار گرفته‌اند به مسألهٔ امنیت وِب هم مربوط می‌شوند. اما همانطور که در [GARF97] خاطرنشان شده است، وِب چالش‌های جدیدی را بوجود می‌آورد که معمولاً در مقولهٔ امنیت کامپیوتر و امنیت شبکهٔ نمی‌گنجند:

- اینترنت دوطرفه است. برخلاف محیط‌های انتشاراتی سنّتی و حتی سیستم‌های انتشاراتی الکترونیک که شامل تله‌تکست، پاسخ صوتی و یا فاکس‌برگردان می‌باشند، وِب نسبت به حملاتی که از طریق اینترنت روی سِرورهای وِب می‌شود آسیب‌پذیر است.
- وِب بصورت فزاینده‌ای بعنوان یک خروجی بسیار مرئی برای اطلاعات مربوط به سازمان‌ها و محصولات مختلف بکار گرفته شده و پایگاهی برای اسناد تجاری محسوب می‌گردد. اگر سِرورهای وِب مورد تهاجم قرار گیرند، شهرت و اعتبار و سرمایهٔ شرکت‌ها مورد تهدید قرار می‌گیرند.
- اگرچه مرورگرهای وِب به آسانی مورد استفاده قرار می‌گیرند، مدیریت و پیکربندی سِرورهای وِب نسبتاً آسان است و تهیهٔ محتویات وِب روز به روز سهل‌تر می‌شود، ولی نرم‌افزار زیربنای وِب بطور فوق‌العاده‌ای پیچیده است. این نرم‌افزار پیچیده ممکن است بسیاری از نقصان‌های امنیتی وِب را پنهان سازد. تاریخچهٔ کوتاه وِب پر از مثال‌هائی از سیستم‌های جدید، به روزرسانی شده و صحیح نصب شده است که در مقابل حملات امنیتی متعددی آسیب‌پذیر بوده‌اند.

- یک سِرور وِب می‌تواند بعنوان یک پایگاه عملیاتی مورد سوءاستفادهٔ مهاجمین قرار گرفته تا به تمام سیستم کامپیوتری یک سازمان حمله نمایند.
- کاربران متنوع و بی‌اطلاع (از دید مسائل امنیتی)، معمولاً کلاینت‌های سِرورهای وِب را تشکیل می‌دهند. این کاربران الزاماً از ریسک‌های موجود اطلاع نداشته و ابزار یا معلومات لازم برای مقابلهٔ مؤثر با این تهدیدها را ندارند.

تهدیدهای امنیتی وِب

جدول ۱-۷ خلاصه‌ای از انواع تهدیدهای امنیتی که در استفاده از وِب با آنها مواجهیم را نشان می‌دهد. یک روش برای دسته‌بندی این تهدیدها این است که آنها را بر اساس حملات غیرفعال و فعال دسته‌بندی کنیم. حملات غیرفعال شامل استراق‌سمع ترافیک شبکه بین مرورگر و سِرور و دست‌یابی به اطلاعات یک سایت وِب است که قرار بوده است محرمانه باشد. حملات فعال شامل جا زدن خود بجای شخص دیگر، تغییر پیام‌های در حال ترانزیت بین کلاینت و سِرور و همچنین تغییردادن اطلاعات یک وِب سایت هستند.

راه دیگری برای طبقه‌بندی تهدیدهای وِب این است که آنها را بر حسب محل تهدید طبقه‌بندی نمائیم: سِرور وِب، مرورگر وِب و ترافیک شبکه‌ای بین مرورگر و سِرور.

مقوله‌های امنیت سِرور و امنیت مرورگر در شاخهٔ امنیت سیستم‌های کامپیوتری جای دارند. در قسمت‌های بعدی این کتاب، مقولهٔ امنیت سیستم بطور کلی مورد بحث قرار می‌گیرد که البته قابل اعمال به امنیت سیستم‌های وِب نیز هست. مقولهٔ مربوط به امنیت ترافیک در طبقه‌بندی امنیت شبکه بوده و در این فصل به آن اشاره می‌شود.

جدول ۱-۷ یک مقایسه از تهدیدهای امنیتی وِب [RUBI97]

	تهدیدها	پیامد تهدیدها	روش‌های مقابله
صحت	• دستکاری در داده‌های کاربر • مرورگر اسب تروا • دستکاری حافظه • دستکاری ترافیک پیام در حال ترانزیت	• از بین رفتن اطلاعات • لورفتن ماشین • آسیب‌پذیری به انواع تهدیدهای دیگر	استفاده از جمع‌های کنترلی مرتبط با رمزنگاری
محرمانگی	• استراق سمع روی اینترنت • دزدی اطلاعات از سِرور • کسب اطلاعات در مورد پیکربندی شبکه • کسب اطلاعات در مورد این‌که کدام کلاینت با سِرور در تماس است.	• از بین رفتن اطلاعات • از بین رفتن محرمانگی	رمزنگاری، استفاده از پروکسی‌های وِب
انکار سرویس	• قطع رشته‌های ارتباطی کاربر • ایجاد سیلابی از تهدیدهای ساختگی • پرکردن حافظه‌های کامپیوتر • ایزوله کردن ماشین با حملات DNS	• ایجاد وقفه • ایجاد اذیت و آزار • بازماندن کاربر از انجام کارهای عادی	جلوگیری از این تهدیدها مشکل است
اعتبارسنجی	• جعل هویت کاربران قانونی • تقلب در داده‌ها	• معرفی غلط کاربر • ایجاد باور نسبت به صحت اطلاعات غلط	تکنیک‌های رمزنگاری

روش‌های برخورد با امنیت ترافیک وِب

روش‌های مختلفی برای تأمین امنیت وِب وجود دارد. روش‌های متفاوتی که مورد توجه قرار گرفته‌اند از نظر سرویس‌هائی که بوجود می‌آورند مشابهت داشته و حتی تا حد زیادی از مکانیسم‌های یکسانی استفاده می‌کنند. تفاوت عمدهٔ آنها در محدودهٔ عملیاتی روش‌ها و همچنین مکان آنها در پشتهٔ پروتکلی TCP/IP است.

شکل ۱-۷ تفاوت اخیر را نشان می‌دهد. یکی از روش‌های فراهم آوردن امنیت وِب، استفاده از امنیت IP است (شکل ۱-۷الف). مزیت استفاده از IPSec این است که برای کاربران و کاربردهای انتهائی نامرئی بوده و یک راه حل همه منظوره بشمار می‌آید. علاوه براین IPSec شامل یک قابلیت فیلترینگ بوده بطوری که تنها ترافیک انتخاب شده، لازم است از سربارهٔ پردازش IPSec استفاده کنند.

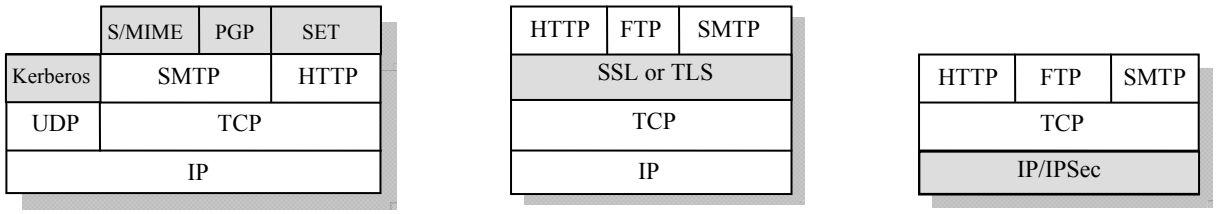
یک روش همه منظورهٔ دیگر این است که امنیت را درست در بالای لایهٔ TCP/IP پیاده‌سازی نمائیم (شکل ۱-۷ب). در رأس این روش لایهٔ سوکت امن Secure Socket Layer (SSL) و پس از آن استاندارد بعدی اینترنت بنام امنیت لایهٔ حمل‌ونقل Transport Layer Security (TLS) قرار دارد. در این سطح دو روش پیاده‌سازی مختلف وجود دارد. برای عمومیت کامل، SSL (یا TLS) می‌تواند بعنوان بخشی از پشتهٔ پروتکلی قلمداد شده و در نتیجه برای کاربردها نامرئی جلوه کند. روش دیگر این است که SSL در بسته‌های نرم‌افزاری مشخص جای داده شود. بعنوان مثال مرورگرهای Netscape و Microsoft Explorer مجهز به SSL به بازار می‌آیند. همچنین بیشتر سِرورهای وِب این پروتکل را بکار گرفته‌اند.

سرویس‌های امنیتی مختص به کاربرد، در شکم کاربردهای مختلف جای دارند. شکل ۱-۷ج مثال‌هائی از این معماری را نشان می‌دهد. حسن این روش این است که سرویس را می‌توان بر حسب نیازهای مشخص یک کاربرد خاص دستکاری نمود. در محدودهٔ امنیت وِب، یک مثال مهم از این نوع برخورد، Secure Electronic Transaction (SET) است.

بقیهٔ این فصل به توصیف SSL/TLS و SET می‌پردازد.

۷-۲ لایهٔ سوکت اَمن و امنیت لایهٔ حمل و نقل (SSL/TLS)

SSL را Netscape پایه‌گذاری کرد. نسخهٔ سوم این پروتکل با نظرسنجی عمومی و بازخورد گرفته شده از صنعت، طراحی شد و به عنوان پیش‌نویس یک سند اینترنت منتشر گردید. بعد از آن وقتی یک توافق کلی برای تسلیم پروتکل بعنوان یک استاندارد اینترنت حاصل شد، گروه کاری TLS در درون IETF تشکیل گردید تا یک استاندارد مشترک را فراهم آورد. اولین نسخهٔ منتشرشده TLS را می‌توان SSLv3.1 دانست که تا حد زیادی با نسخهٔ قدیمی SSLv3 سازگار است.



(الف) سطح شبکه (ب) سطح حمل و نقل (ج) سطح کاربرد

شکل ۱-۷ موقعیت نسبی تسهیلات امنیتی در پشتهٔ پروتکلی TCP/IP

عمده مطالب این بخش به بحث در مورد SSLv3 اختصاص دارد. در انتهای بخش، تفاوت‌های بین SSLv3 و TLS توضیح داده شده است.

معماری SSL

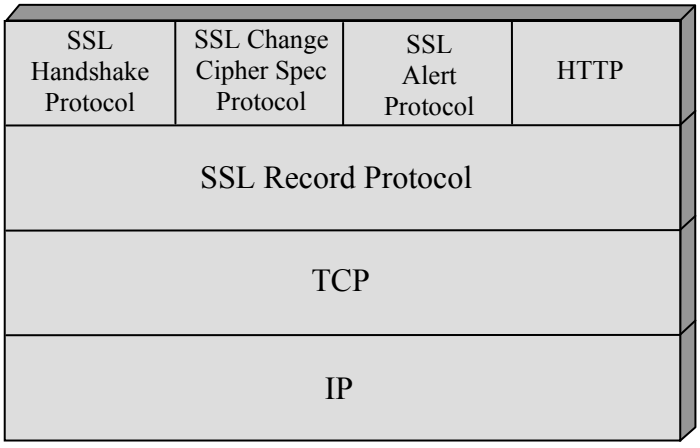
SSL برای فراهم آوردن یک سرویس امن سر- به- سر با استفاده از TCP/IP طراحی شده است. SSL یک پروتکل تنها نبوده بلکه همانطور که در شکل ۲-۷ نشان داده شده است، دو لایه از پروتکل‌ها آن را تشکیل می‌دهند.

پروتکل SSL Record سرویس‌های امنیتی پایه را برای پروتکل‌های مختلف لایهٔ بالاتر فراهم می‌آورد. علی‌الخصوص پروتکل http (hypertext transfer protocol) که سرویس انتقال برای تعامل کلاینت/سِرور روی وِب را بوجود می‌آورد می‌تواند در بالای SSL کار کند. سه پروتکل لایهٔ بالاتر به عنوان بخشی از SSL تعریف شده‌اند: پروتکل Handshake، پروتکل Change Cipher Spec، و پروتکل Alert. این پروتکل‌های مختص به SSL، در مدیریت ارتباطات SSL مورد استفاده قرار گرفته و بعداً در مورد آنها صحبت خواهد شد.

دو مفهوم مهم SSL، یکی اتصال SSL و دیگری اجلاس SSL است که در مشخصات به صورت زیر تعریف شده‌اند:

- **اتصال (Connection)**: یک اتصال، یک بستر حمل‌ونقل اطلاعات (در تعریف لایه‌های OSI) است که نوع مناسبی از سرویس را فراهم می‌آورد. برای SSL چنین اتصالاتی دارای رابطهٔ نظیر- به- نظیراند. اتصالات موقتی هستند. هر اتصال مرتبط با یک اجلاس است.
- **اجلاس (Session)**: یک اجلاس SSL، یک اتحاد بین یک کلاینت و یک سِرور است. اجلاس‌ها بتوسط پروتکل Handshake شکل می‌گیرند. اجلاس‌ها، مجموعه‌ای از پارامترهای امنیتی مرتبط با رمزنگاری را تعریف می‌کنند که می‌توانند بین اتصالات مختلف به اشتراک گذاشته شوند. از اجلاس‌ها بمنظور اجتناب از توافقات گران قیمت در مورد پارامترهای امنیتی جدید برای هر اتصال استفاده می‌شود.

بین دو زوج مرتبط (کاربردهائی همانند HTTP روی کلاینت و سِرور)، ممکن است اتصالات امن متعددی وجود داشته باشد. همچنین از نظر تئوری، ممکن است اجلاس‌های هم‌زمان متعددی نیز بین طرفین موجود باشد اما از این خصوصیت در عمل استفاده‌ای نمی‌شود.



شکل ۲-۷ پشتهٔ پروتکلی SSL

در حقیقت با هر اجلاس حالات متعددی مرتبط است. همین که اجلاسی برقرار شد، یک حالت عملیاتی جاری هم برای خواندن و هم برای نوشتن (یعنی دریافت و ارسال) وجود دارد. علاوه بر آن در خلال اجرای پروتکل Handshake حالات موقت خواندن و نوشتن ایجاد می‌شود. پس از پایان موفقیت‌آمیز پروتکل Handshake، حالات موقت به حالات جاری تبدیل می‌شوند.

حالت یک اجلاس بتوسط پارامترهای زیر تعریف می‌شود (تعاریف از مشخصه‌های SSL اقتباس شده‌اند):

- **Session identifier**: یک دنبالهٔ اختیاری از بایت‌ها که بتوسط سِرور انتخاب شده تا نشان دهد که حالت اجلاس فعال و یا قابل تداوم است.
- **Peer certificate**: یک گواهی X509.v3 واحد نظیراست. این عنصر حالت ممکن است خالی باشد.
- **Compression method**: الگوریتم استفاده شده برای فشرده‌سازی دیتا قبل از رمزنگاری است.
- **Cipher spec**: الگوریتم رمزنگاری دیتای اصلی (مانند null، DES و غیره)، الگوریتم hash (مانند SHA-1 یا MD5) که برای محاسبه MAC بکار می‌رود و همچنین سایر مشخصات رمزنگاری مثل اندازهٔ hash را مشخص می‌سازد.
- **Master secret**: ۴۸ بایت سرّی که بین کلاینت و سِرور به اشتراک گذاشته می‌شود.
- **Is resumable**: یک پرچم که مشخص می‌کند آیا اجلاس می‌تواند برای ایجاد اتصالات جدید بکار رود.

حالت یک اتصال با پارامترهای زیر تعریف می‌شود:

- **Server and client random**: دنباله‌ای از بایت‌ها که بتوسط سِرور و کلاینت برای هر اتصال انتخاب می‌شوند.
- **Server write MAC secret**: کلید سرّی بکار رفته در عملیات تولید MAC بر روی دیتائی که بتوسط سِرور ارسال شده است.
- **Client write MAC secret**: کلید سرّی بکار رفته در عملیات تولید MAC بر روی دیتائی که بتوسط کلاینت ارسال شده است.
- **Server write key**: کلید رمزنگاری سنّتی برای دیتائی که بتوسط سِرور رمزنگاری شده و بتوسط کلاینت رمزگشائی می‌شود.
- **Client write key**: کلید رمزنگاری سنّتی برای دیتائی که بتوسط کلاینت رمزنگاری شده و بتوسط سِرور رمزگشائی می‌شود.
- **Initialization vectors**: وقتی از یک رمز قالبی در مُود CBC استفاده می‌شود، یک بردار ابتدائی (IV) برای هر کلید وجود دارد. این میدان ابتدا بتوسط پروتکل SSL Handshake انتخاب می‌شود. بعد از آن، از آخرین قالب رمزشده هر رکورد، برای IV رکورد بعدی استفاده می‌شود.
- **Sequence numbers**: هر یک از طرفین ارتباط، شماره ردیف‌های متفاوتی را برای پیام‌های ارسال شده و دریافت شده برای هر اتصال نگهداری می‌کند. وقتی یک طرف ارتباط، یک پیام تغییر Cipher spec را ارسال می‌کند، شمارهٔ ردیف مرتبط صفر می‌شود. شماره ردیف‌ها نمی‌توانند از $۲^{۶۴}-۱$ تجاوز نمایند.

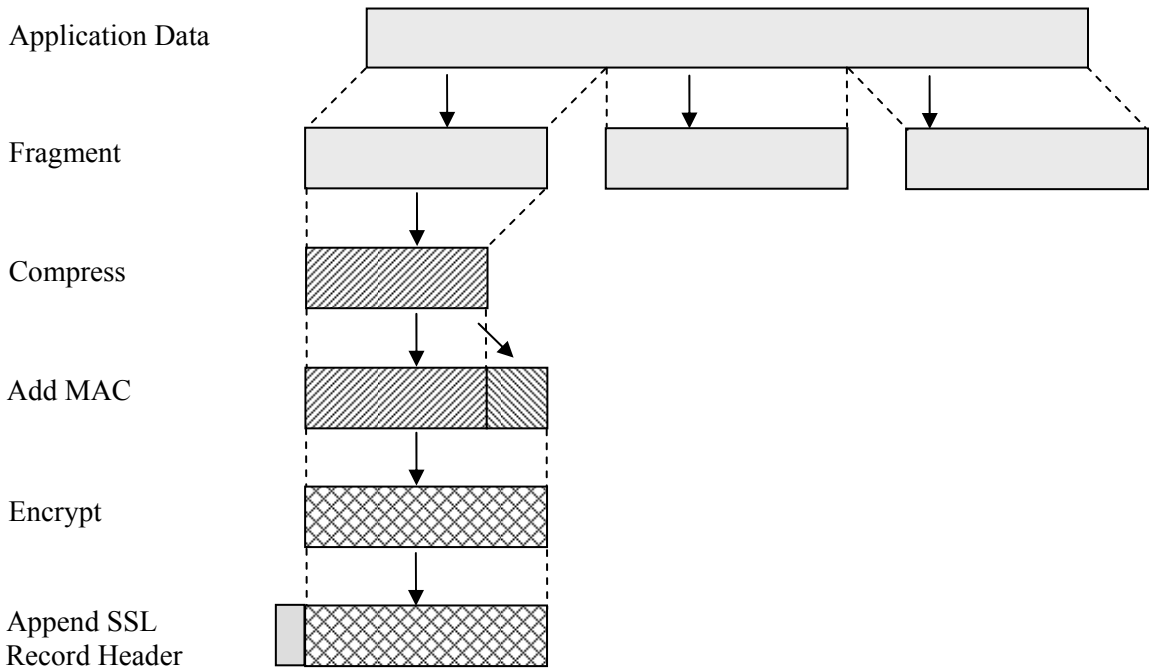
پروتکل SSL RECORD

پروتکل SSL Record دو سرویس را برای اتصالات SSL فراهم می‌آورد:

- **محرمانگی**: پروتکل Handshake، یک کلید سرّی مشترک را تعریف می‌کند که برای رمزنگاری سنّتی محمولهٔ دیتای SSL بکار می‌رود.
- **صحت پیام**: پروتکل Handshake، همچنین یک کلید سرّی مشترک دیگر را تعریف می‌کند که برای تشکیل کُد اعتبارسنجی پیام (MAC) مورد نیاز است.

شکل ۳-۷ عملیات کلی پروتکل SSL Record را نشان می‌دهد. این پروتکل یک پیام کاربردی را که قرار است ارسال شود اخذ کرده، دیتا را بصورت قالب‌های کوچکتر قابل پردازش درآورده، دیتا را بصورت اختیاری فشرده‌سازی نموده، یک سرآیند (header) به آن اضافه کرده و واحد نتیجه شده را در یک سِگمنت TCP انتقال می‌دهد. دیتای دریافت شده رمزگشائی شده، تأئید شده، از فشرده‌سازی درآمده و دوباره بهم می‌پیوندد و سپس به کاربران لایهٔ‌های بالاتر تحویل داده می‌شود.

اولین قدم، **قطعه قطعه کردن دیتا (fragmentation)** است. هر پیام لایهٔ بالاتر بصورت بلوک‌هائی با اندازهٔ $۲^{۱۴}$=۱۶,۳۸۴ بایت یا کمتر در می‌آید. سپس **فشرده‌سازی (compression)** بصورت اختیاری و در صورت نیاز روی آن صورت می‌پذیرد. فشرده‌سازی بایستی بدون تلفات بوده و نبایستی طول محتویات را بیش از ۱,۰۲۴ بایت افزایش دهد. در SSLv3 (و همچنین در نسخهٔ فعلی TLS) نوع الگوریتم فشرده‌سازی مشخص نشده و بنابراین پیش فرض، عدم حضور آن است.



شکل ۳-۷ عملیات SSL Record Protocol

قدم بعدی پردازش، محاسبهٔ یک **کُد اعتبارسنجی پیام (MAC)** از دیتای فشرده شده است. برای این کار از یک کلید سرّی مشترک استفاده می‌شود. محاسبات چنین تعریف می‌شوند:

hash(MAC_write_secret || pad_2 ||

hash(MAC_write_secret || pad_1 || seq_num || SSLCompressed.type ||

SSLCompressed.length || SSLCompressed.fragment))

که درآن

||	= (concatenation) جمع رشته‌ای
MAC_write_secret	= کلید سرّی مشترک
hash	= الگوریتم رمزی hash که یا MD5 و یا SHA-1 است
pad_1	= بایت 0110 0011 (36 هکزادسیمال) که ۴۸ بار برای MD5 (۳۸۴ بیت) و ۴۰ بار برای SHA-1 (۳۲۰ بیت) تکرار می‌شود
pad_2	= بایت 1100 0101 (5C هکزادسیمال)که ۴۸ بار برای MD5 و ۴۰ بار برای SHA-1 تکرار می‌شود
seq _num	= شماره ردیف این پیام
SSLCompressed.type	= پروتکل لایهٔ بالاتر که برای پردازش این فِرگمنت بکار می‌رود
SSLCompressed.length	= طول فِرگمنت فشرده شده
SSLCompressed.fragment	= فِرگمنت فشرده شده (اگر از فشرده‌سازی استفاده نشده است، متن ساده)

توجه کنید که این خیلی شبیه الگوریتم HMAC است که در فصول قبل از آن یاد شد. تفاوت در این است که دو مقدار pad در SSLv3 باهم جمع رشته‌ای شده درحالی که در HMAC باهم XOR می‌شوند. الگوریتم MAC در SSLv3 مبتنی بر پیش نویس اینترنتی اولیه برای HMAC بوده که از جمع رشته‌ای استفاده می‌نماید. آخرین نسخهٔ HMAC که در RFC 2104 تعریف شده است از XOR استفاده می‌کند.

سپس پیام فشرده‌شده بعلاوهٔ MAC با استفاده از روش رمزنگاری متقارن **رمزنگاری** می‌شود. رمزنگاری نمی‌تواند طول محتوی را بیش از۱,۰۲۴ بایت افزایش دهد و بنابراین طول نهائی نبایستی از ۲,۰۴۸ + 2^{14} تجاوز کند. الگوریتم‌های رمزنگاری زیر الگوریتم‌های مجازند:

رمز قالبی		رمز دنباله‌ای	
الگوریتم	طول کلید	الگوریتم	طول کلید
AES	128,256	RC4-40	40
IDEA	128	RC4-128	128
RC2-40	40		
DES-40	40		
DES	56		
3DES	168		
Fortezza	80		

Fortezza می‌تواند در روش رمزنگاری یک کارت هوشمند بکار رود.

برای رمزنگاری دنباله‌ای، پیام فشرده شده باضافهٔ MAC رمزنگاری می‌شوند. توجه کنید که MAC قبل از اینکه رمزنگاری صورت پذیرد محاسبه می‌شود و آنگاه MAC باضافهٔ متن ساده یا متن سادهٔ فشرده شده رمزنگاری می‌گردد.

برای رمزنگاری قالبی، padding می‌تواند پس از محاسبهٔ MAC و قبل از رمزنگاری انجام شود. padding عبارت از تعدادی بایت لائی است که در انتهای آن یک بایت که اندازهٔ لائی را نشان می‌دهد قرار دارد. اندازهٔ کل لائی کوچکترین مقداری است که بتواند اندازهٔ کل دیتائی که باید رمزنگاری شود (متن ساده باضافهٔ MAC باضافهٔ padding) را به اندازهٔ مضربی از طول بلوک رمز قالبی درآورد. مثال این مورد یک متن سادهٔ ۵۸- بایتی (یا اگر فشرده‌سازی صورت پذیرفته است متن فشرده) با یک MAC با اندازهٔ ۲۰ بایت (با استفاده از SHA-1) است که با یک الگوریتم که طول قالب آن ۸ بایت است (مثل DES) رمزنگاری شود. با محاسبهٔ بایت مربوط به طول لائی این مجموعه ۷۹ بایت خواهد شد. برای اینکه این مقدار مضرب ۸ شود، ۱ بایت لائی به آن اضافه می‌شود.

قدم نهائی در پردازش پروتکل SSL Record این است که یک سرآیند(header) که شامل میدان‌های زیر است به ابتدای آن اضافه شود:

- **نوع محتوا (۸ بیت)**: پروتکل لایهٔ بالاتر که برای پردازش این فِرگمنت بکار می‌رود.
- **نسخهٔ اصلی (۸ بیت)**: نسخهٔ اصلی SSL مورد استفاده را نشان می‌دهد. برای SSLv3 این اندازه 3 خواهد بود.
- **نسخهٔ فرعی (۸ بیت)**: نسخهٔ فرعی مورد استفاده را نشان می‌دهد. برای SSLv3 این مقدار 0 است.
- **طول فشرده‌شده (۱۶ بیت)**: طول فِرگمنت متن ساده بر حسب بایت (در صورت فشرده‌سازی، طول فِرگمنت فشرده شده). اندازهٔ ماکزیمم ۲,۰۴۸ + $۲^{۱۴}$ است.

محتواهای تعریف شده عبارت از change_cipher_spec، alert، handshake و application_data هستند. سه‌تای اول پروتکل‌های مختص SSL بوده که موضوع بحث بعدی است. توجه کنیدکه هیچ تفاوتی بین انواع کاربردهائی (مثلاً http) که می‌توانند از SSL استفاده کنند وجود ندارد. محتوای داده‌های خلق شده بتوسط چنین کاربردهائی از دید SSL غیرقابل رؤیت‌اند.

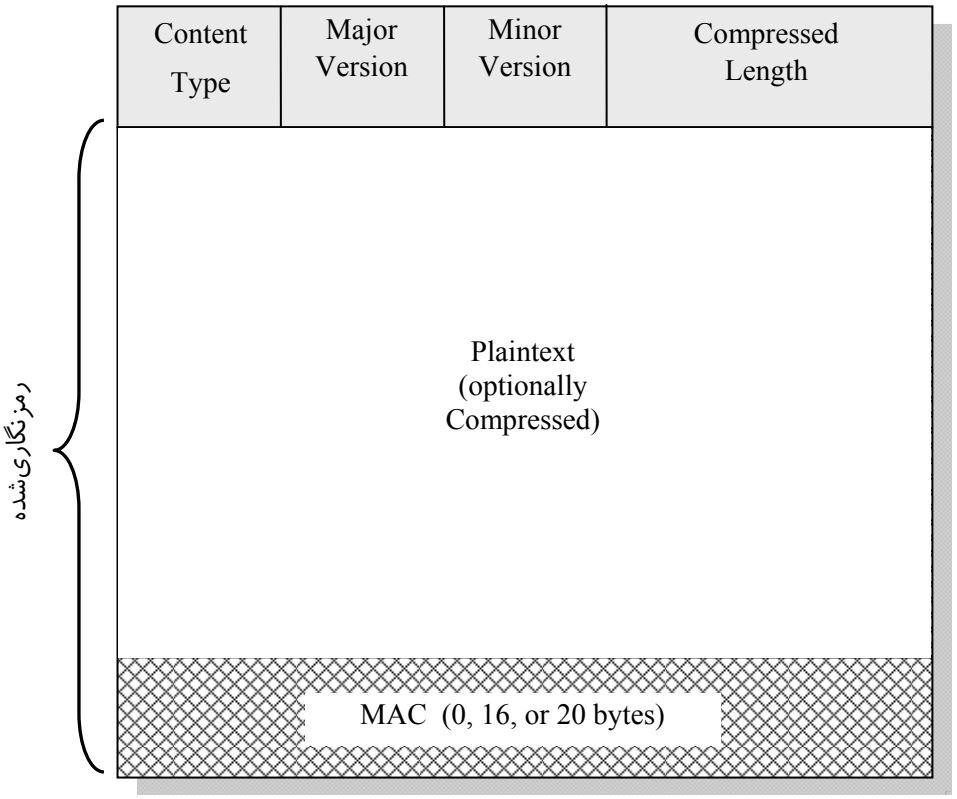
شکل ۴-۷ فرمت SSL Record را نشان می‌دهد.

پروتکل Change Cipher Spec

پروتکل Change Cipher Spec یکی از سه پروتکل مختص SSL است که از پروتکل SSL Record استفاده کرده و ساده‌ترین آنهاست. این پروتکل از یک پیام تنها تشکیل شده (شکل ۵-۷الف) که شامل یک بایت منفرد با مقدار 1 است. تنها هدف این پیام این است که باعث شود تا وضعیت موقت در وضعیت جاری کپی شده و مجموعهٔ رمزهائی که باید در این اتصال بکار روند بروز گردد.

پروتکل Alert

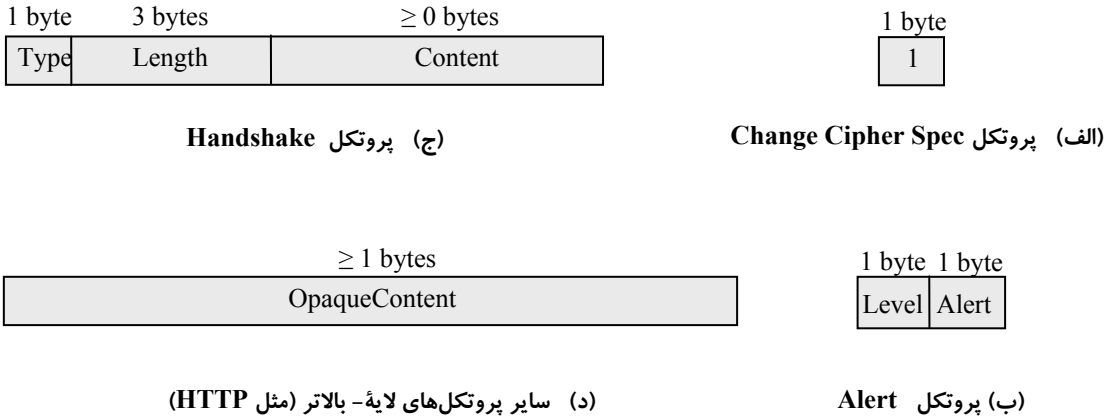
پروتکل Alert برای رساندن هشدارهای مرتبط با SSL به واحد نظیر بکار می‌رود. همانند سایر کاربردهائی که از SSL استفاده می‌کنند، پیام‌های هشدار فشرده‌سازی و رمزنگاری می‌شوند.



شکل ۴-۷ فرمت SSL Record

هر پیام در این پروتکل از دو بایت تشکیل می‌شود(شکل ۵-۷ب). بایت اول یا اندازهٔ 1 (warning) و یا اندازهٔ 2 (fatal) را داراست که اهمیت پیام را نشان می‌دهد. اگر اندازه برابر fatal باشد، SSL بلافاصله اتصال را خاتمه می‌دهد. سایر اتصالات مربوط به این اجلاس ممکن است ادامه یابند اما هیچ اتصال جدیدی در این اجلاس نمی‌تواند برقرار گردد. بایت دوم شامل یک کُد است که هشدار خاصی را مشخص می‌سازد. در ابتدا هشدارهائی که همیشه fatal هستند را بیان می‌کنیم (تعاریف از مشخصه‌های SSL اقتباس شده‌اند):

- **unexpected_message**: یک پیام نامربوط دریافت شده است.
- **bad_record_mac**: یک MAC ناصحیح دریافت شده است.
- **decompression_failure**: تابع معکوس فشرده‌سازی یک ورودی غیرصحیح را دریافت کرده است (مثلاً قادر نیست تا آنچه را دریافت کرده از فشرده‌سازی خارج کرده و یا در صورت این کار پیام از طول ماکزیمم مجاز بیشتر خواهد شد).
- **handshake_failure**: فرستنده قادر نبوده است تا روی یک مجموعهٔ قابل قبول از پارامترهای امنیتی با توجه به امکانات موجود توافق حاصل نماید.
- **illegal_parameter**: یک میدان در یک پیام handshake خارج از محدوده بوده و یا با سایر میدان‌های موجود همخوانی نداشته است.



شکل ۵-۷ محمولهٔ SSL Record Protocol

بقیهٔ هشدارها چنین‌اند:

- **close_notify**: به گیرنده اطلاع می‌دهد که فرستنده پیام‌های دیگری را روی این اتصال نخواهد فرستاد. هر یک از طرفین ارتباط موظف‌اند تا یک هشدار close_notify را قبل از بستن مرحلهٔ نوشتن پیام در یک اتصال ارسال دارند.
- **no_certificate**: ممکن است در پاسخ به درخواست یک گواهی، در صورتی که گواهی‌نامهٔ مطلوب در دسترس نباشد، ارسال شود.
- **bad_certificate**: گواهی دریافت شده مشکل داشته است (مثلاً شامل امضائی بوده است که مورد تأئید نیست).
- **unsupported_certificate**: نوع گواهی دریافت شده قابل قبول نیست.
- **certificate_revoked**: یک گواهی بتوسط امضاءکنندهٔ آن باطل شده است.
- **certificate_expired**: اعتبار یک گواهی تمام شده است.
- **certificate_unknown**: یک مسألهٔ غیرمشخص در مرحلهٔ پذیرش گواهی پیش آمده است که آن را غیرقابل پذیرش نموده است.

پروتکل Handshake

پیچیده‌ترین بخش SSL مربوط به پروتکل Handshake است. این پروتکل به سِرور و کلاینت اجازه می‌دهد تا هویت یکدیگر را تأئید نموده و راجع به یک الگوریتم رمزنگاری و MAC و همچنین کلیدهای رمزنگاری که قرار است دیتای ارسال شده در یک رکورد SSL را محافظت کنند، توافق نمایند. پروتکل Handshake قبل از اینکه هیچگونه دیتای کاربردی انتقال یابد، اجرا می‌شود.

پروتکل Handshake شامل یک سری پیام‌های ردوبدل شده بین کلاینت و سِرور است. تمام این پیام‌ها فرمت نشان داده شده در شکل ۵-۷ج را دارند. هر پیام شامل سه میدان است:

- **نوع (۱ بایت)**: نمایشگر یکی از ۱۰ نوع پیام متفاوت است. جدول ۲-۷ انواع پیام‌های مرتبط را نشان داده است.
- **طول (۳ بایت)**: طول پیام بر حسب بایت.
- **محتوا (0 بایت ≤)** : پارامترهای مرتبط با این پیام. این پارامترها در جدول ۲-۷ لیست شده‌اند.

شکل ۶-۷ مبادلات اولیه برای برقراری یک اتصال منطقی بین کلاینت و سِرور را نشان می‌دهد. این مبادلات را می‌توان شامل چهار فاز دانست.

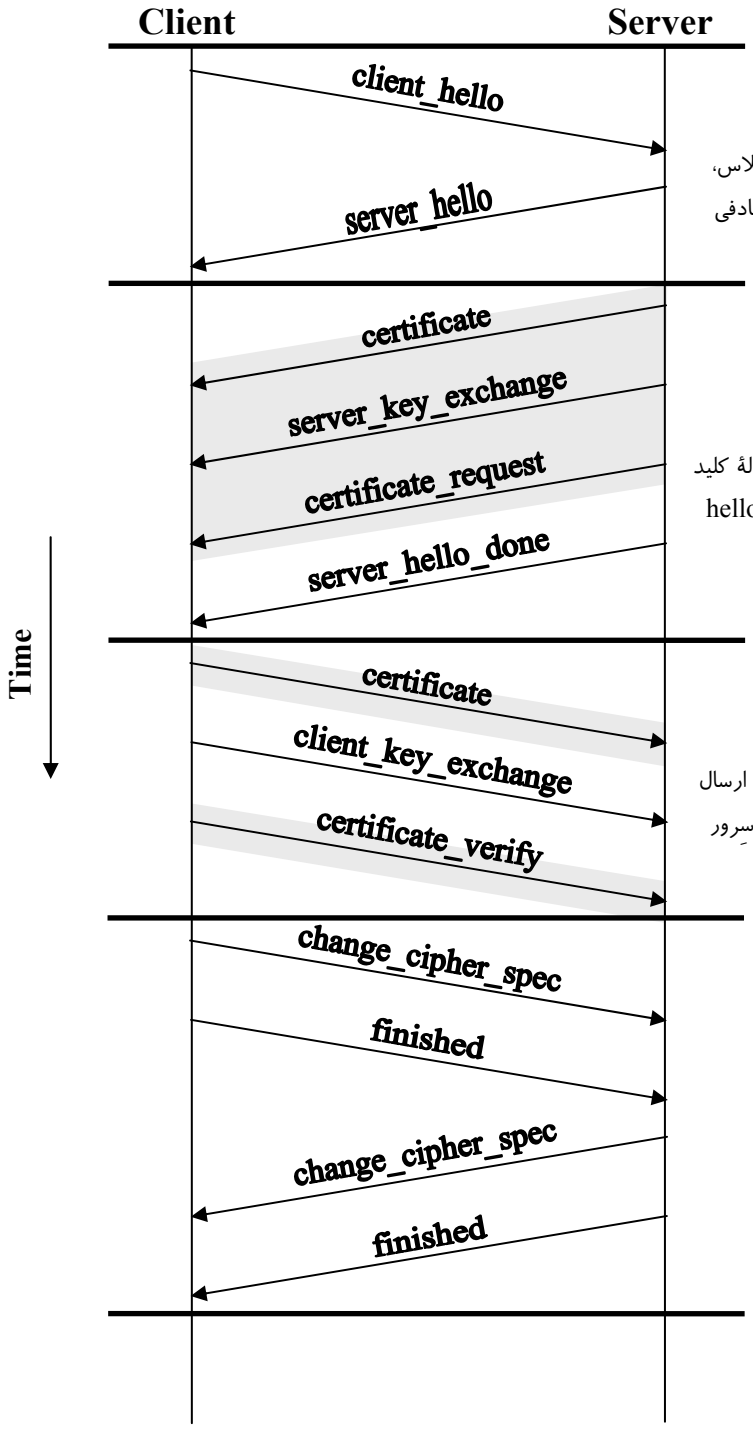
فاز اول- استقرار توانائی‌های امنیتی

این فاز برای شروع یک اتصال منطقی و استقرار توانائی‌های امنیتی مرتبط با آن بکار می‌رود. مبادلهٔ پیام‌ها از طرف کلاینت شروع می‌شود که یک **پیام client_hello** که شامل پارامترهای زیر است را ارسال می‌کند:

- **Version**: بالاترین نسخهٔ SSL که بتوسط کلاینت قابل فهم است.
- **Random**: یک ساختار تصادفی تولیدشده بتوسط کلاینت که شامل یک برچسب زمانی ۳۲- بیتی و همچنین ۲۸ بایت تولید شده بتوسط یک تولیدکنندهٔ اعداد تصادفی امن است. این مقادیر بعنوان nonceهای مختلف عمل نموده و در خلال مبادلهٔ کلید برای جلوگیری از حملات بازخوانی(replay) بکار می‌روند.
- **Session ID**: یک شناسهٔ اجلاس با طول متغیر. یک مقدار غیر صفر نشان می‌دهدکه کلاینت می‌خواهد پارامترهای موجود را بروزرسانی کرده و یا یک اتصال جدید در این اجلاس را برقرار نماید. یک مقدار صفر نمایش این است که کلاینت تمایل دارد تا یک اتصال جدید و یا یک اجلاس جدید را ایجاد کند.
- **CipherSuite**: این لیستی شامل مجموعه‌ای از الگوریتم‌های رمزنگاری است که مورد حمایت کلاینت بوده و بر حسب ترجیح نزولی مرتب شده است. هر عنصر لیست (هرمجموعهٔ رمز) هم الگوریتم تبادل کلید و هم یک CipherSpec را تعریف می‌کند که هردوی اینها بعداً توضیح داده خواهد شد.
- **Compression Method**: لیستی از روش‌های فشرده‌سازی مورد حمایت کلاینت است.

جدول ۲-۷ انواع پیام‌های پروتکل SSL Handshake

نوع پیام	پارامترها
hello_request	null
client_hello	version, random, session id, cipher suite, compression method
server_hello	version, random, session id, cipher suite, compression method
certificate	chain of X.509v3 certificates
server_key_exchange	parameters, signature
certificate_request	type, authorities
server_done	null
certificate_verify	signature
client_key_exchange	parameters, signature
finished	hash value



فاز اول

توانائی‌های امنیتی که شامل نسخهٔ پروتکل، ID اجلاس، مجموعهٔ رمزنگاری، روش فشرده‌سازی و اعداد تصادفی اولیه است استقرار می‌یابند.

فاز دوم

سِرور ممکن است گواهی خود را ارسال کرده، مبادلهٔ کلید نموده و گواهی درخواست نماید. سِرور پایان پیام hello را اعلام می‌دارد.

فاز سوم

کلاینت گواهی خود را در صورت درخواست سِرور ارسال می‌دارد. کلاینت ممکن است تأئید دریافت گواهی سِرور را نیز ارسال نماید.

فاز چهارم

مجموعهٔ رمزنگاری تعویض شده و پروتکل handshake خاتمه می‌یابد.

توجه:
مبادلاتی که با سایه علامت‌گذاری شده‌اند اختیاری و یا وابسته به حالات خاص بوده و همیشه ارسال نمی‌شوند.

شکل ۶-۷ عملیات پروتکل Handshake

پس از ارسال پیام client_hello، کلاینت منتظر **پیام server_hello** می‌ماند که شامل همان پارامترهای پیام client_hello است. میدان Version شامل پائین‌ترین نسخهٔ پیشنهادشده بتوسط کلاینت و بالاترین نسخهٔ حمایت شده بتوسط سِرور است. میدان Random بتوسط سِرور تولید شده و مستقل از میدان Random کلاینت است. اگر میدان

SessionID کلاینت صفر نبوده است همان مقدار بتوسط سِرور نیز مورد استفاده قرار می‌گیرد. در غیر اینصورت میدان SessionID سِرور شامل مقدار تازه‌ای برای یک اجلاس جدید خواهد بود. میدان CipherSuite شامل یک مجموعهٔ رمزنگاری انتخاب شده بتوسط سِرور از بین آنهائی است که بتوسط کلاینت پیشنهاد شده‌اند. میدان Compression شامل متد فشرده‌سازی انتخاب شده بتوسط سِرور از بین آنهائی است که بتوسط کلاینت پیشنهاد شده‌اند.

اولین عنصر از پارامترهای CipherSuite متد مبادلهٔ کلید است (یعنی روشی که بتوسط آن کلیدهای رمزنگاری سنّتی و MAC مبادله می‌شوند). روش‌های زیر برای مبادلهٔ کلید مورد حمایت‌اند:

- **RSA**: کلید سرّی بتوسط کلید عمومی RSA گیرنده رمزنگاری می‌گردد. یک گواهی‌نامهٔ کلید- عمومی برای کلید گیرنده بایستی اخذ شود.
- **Fixed Diffie-Hellman**: این یک روش مبادلهٔ کلید Diffie-Hellman است که در آن گواهی‌نامهٔ سِرور شامل پارامترهای عمومی Diffie-Hellman بوده که بتوسط مسئول گواهی (CA) امضاء شده است. یعنی گواهی‌نامهٔ کلید- عمومی شامل پارامترهای کلید عمومی Diffie-Hellman است. کلاینت پارامترهای کلید عمومی Diffie-Hellman خود را یا در یک گواهی عرضه می‌نماید (اگر اعتبارسنجی کلاینت مورد نیاز باشد)، و یا اینکه آنها را در یک مبادلهٔ کلید عرضه می‌دارد. این روش به ایجاد یک کلید سرّی مشترک، مبتنی بر محاسبات Diffie-Hellman با استفاده از کلیدهای عمومی، بین دو واحد نظیر منجر می‌گردد.
- **Ephemeral Diffie-Hellman**: این تکنیک برای خلق کلیدهای سرّی ephemeral (موقت- یکبارمصرف) بکار می‌رود. در این مورد کلیدهای عمومی Diffie-Hellman مبادله می‌شوند که بتوسط کلیدهای خصوصی RSA فرستنده و یا کلید DSS امضاء شده‌اند. گیرنده می‌تواند از کلید عمومی متناظر با کلید خصوصی برای تأئید امضاء استفاده نماید. گواهی‌نامه‌ها برای اعتبارسنجی کلیدهای عمومی بکار می‌روند. بنظر می‌رسدکه این روش امن‌ترین روش از بین سه روش Diffie-Hellman باشد زیرا بواسطهٔ آن کلید موقت اعتبارسنجی شده ایجاد می‌گردد.
- **Anonymous Diffie-Hellman**: الگوریتم پایهٔ Diffie-Hellman بدون اعتبارسنجی مورد استفاده قرار می‌گیرد. یعنی هر طرف پارامترهای کلید عمومی Diffie-Hellman خود را برای دیگری ارسال کرده بدون اینکه هیچگونه احراز هویتی صورت پذیرد. این روش به حملات man-in-the-middle، که درآن حمله‌کننده بصورت ناشناس در وسط قرار گرفته و با طرفین ارتباط برقرار می‌کند، آسیب‌پذیر است.
- **Fortezza**: تکنیک تعیین شده در روش Fortezza.

بدنبال تعریف یک روش مبادلهٔ کلید، CipherSpec است که شامل میدان‌های زیر است:

- **CipherAlgorithm**: هر یک از الگوریتم‌هائی که قبلاً از آنها یاد شد مانند: DES40, IDEA, Foretzza 3DES ,DES ,RC2 ,RC4
- **MACAlgorithm**: MD5 یا SHA-1
- **CipherType**: قالبی یا دنباله‌ای
- **IsExportable**: صحیح یا غلط
- **HashSize**: صفر یا ۱۶ بایت (برای MD5) و یا ۲۰ بایت (برای SHA-1).
- **Key Material**: دنباله‌ای از بایت‌ها که شامل داده‌های بکار رفته در تولید کلیدهای نوشتن است.
- **IV Size**: اندازهٔ Initialization Value برای رمزنگاری CBC (Cipher Block Chaining)

فاز دوم – تصدیق هویت سِرور و مبادلهٔ کلید

سِرور این فاز را، اگر لازم است که هویتش تصدیق شود، با ارسال گواهی‌نامهٔ خود آغاز می‌کند. پیام شامل یک یا زنجیره‌ای از گواهی‌های X.509 است. **پیام certificate** برای هریک از انواع روش‌های توافق شده مبادلهٔ کلید بجز anonymous Diffie-Hellman مورد نیاز است. توجه شود که اگر از fixed Diffie-Hellman استفاده شود این پیام گواهی بعنوان پیام مبادلهٔ کلید سِرور عمل می‌کند زیرا شامل پارامترهای عمومی Diffie-Hellman سِرور است.

سپس یک **پیام server_key_exchange** در صورت لزوم ارسال می‌شود. در دو حالت این پیام مورد لزوم نیست: (۱) سِرور یک گواهی با پارامترهای fixed Diffie-Hellman فرستاده است. (۲) از مبادلهٔ کلید RSA استفاده شود. پیام server_key_exchange برای موارد زیر مورد نیاز است:

- **Anonymous Diffie-Hellman**: محتوای پیام شامل دو مقدار عمومی Diffie-Hellman (یک عدد اول و ریشهٔ اولیهٔ آن عدد) و علاوه بر آن کلید عمومی Diffie-Hellman سِرور می‌باشد (به مبحث مربوط مراجعه شود).
- **Ephemeral Diffie-Hellman**: محتوای پیام شامل سه پارامتر Diffie-Hellman فراهم شده برای anonymous Diffie-Hellman بعلاوهٔ یک امضاء برای آن پارامترهاست.
- **RSA Key exchange که در آن سِرور از RSA استفاده می‌کند اما تنها یک کلید RSA برای امضاء دارد**: در اینجا کلاینت نمی‌تواند به تنهائی یک کلید سرّی که با کلید عمومی سِرور رمزنگاری شده است را ارسال نماید. در عوض سِرور بایستی یک جفت کلید RSA عمومی/خصوصی موقتی خلق کرده و از پیام server_key_exchange برای ارسال کلید عمومی استفاده نماید. محتوای پیام شامل دو پارامتر کلید عمومی موقت RSA (مدول و آرگومان) بعلاوهٔ امضاء آن پارامترهاست.
- **Foretzza**

جزئیات اضافی بیشتری در مورد امضاءها مورد تعهد است. همانند قبل یک امضاء بتوسط محاسبهٔ اندازهٔ hash یک پیام و رمزنگاری آن با کلید خصوصی فرستنده خلق می‌شود. در این مورد hash بصورت زیر تعریف می‌شود

hash(ClientHello.random || ServerHello.random || ServerParams)

بنابراین hash نه تنها پارامترهای Diffie-Hellman یا RSA را می‌پوشاند بلکه دو nonce از پیام‌های اولیهٔ hello را هم پوشش می‌دهد. این امر از حملات بازخوانی و یا نمایش غلط جلوگیری می‌کند. در مورد یک امضاء DSS، hash با استفاده از الگوریتم SHA-1 انجام می‌شود. در مورد یک امضاء RSA، هم hash مربوط به MD5 و هم hash مربوط به SHA-1 محاسبه می‌شود و جمع رشته‌ای این دو hash (۳۶ بایت) بتوسط کلید خصوصی سِرور رمزنگاری می‌شود.

در مرحلهٔ بعد یک سِرور غیرناشناس (سِروری که از anonymous Diffie-Hellman استفاده نمی‌کند)، می‌تواند از کلاینت درخواست یک گواهی‌نامه نماید. پیام certificate_request_message شامل دو پارامتر است: certificate_type و certificate_authorities. certificate_type نشان‌دهندهٔ الگوریتم رمزنگاری کلید- عمومی و کاربردهای آن است:

- RSA، فقط امضاء
- DSS، فقط امضاء
- RSA برای fixed Diffie-Hellman. در این مورد امضاء فقط برای اعتبارسنجی، با ارسال یک گواهی که بتوسط RSA امضاء شده است، بکار می‌رود
- DSS برای fixed Diffie-Hellman که باز هم فقط برای اعتبارسنجی استفاده می‌شود
- RSA برای ephemeral Diffie-Hellman
- DSS برای ephemeral Diffie-Hellman
- Fortezza

پارامتر دوم در پیام certificate_request یک لیست از نام‌های شناخته شده از CAهای مورد پذیرش است.

پیام آخر فاز دوم که همیشه مورد نیاز است، **پیام server_done** است که بتوسط سِرور ارسال می‌شود تا نشان دهد که پایان hello سِرور و پیام‌های مرتبط با آن است. پس از ارسال این پیام، سِرور منتظر پاسخ کلاینت می‌شود. پیام server_done شامل هیچ پارامتری نیست.

فاز سوم – تصدیق هویت کلاینت و مبادلهٔ کلید

پس از دریافت پیام server_done، کلاینت بایستی اگر لازم است تحقیق کند که سِرور یک گواهی معتبر را ارسال نموده و همچنین کنترل نماید که پارامترهای server_hello قابل قبول‌اند. اگر همه چیز بر وفق مراد باشد، کلاینت یک یا چند پیام را بسمت سِرور خواهد فرستاد.

اگر سِرور درخواست یک گواهی‌نامه نموده باشد، کلاینت این فاز را با ارسال یک **پیام certificate** آغاز می‌کند. اگر هیچ گواهی مناسبی وجود نداشته باشد، کلاینت بجای آن یک هشدار no_certificate را ارسال خواهد کرد.

سپس در این فاز **پیام client_key_exchange** بایستی ارسال گردد. محتوای پیام وابسته به نوع مبادلهٔ کلید بشرح زیر است:

- **RSA**: کلاینت یک دیتای ۴۸- بایتی *pre-master sercert* را تولید کرده و آن را با کلید عمومی گواهی‌نامهٔ سِرور و یا کلید موقت RSA از پیام server_key_exchange رمزنگاری می‌کند. کاربرد آن برای محاسبهٔ یک *master secret* بعداً توضیح داده خواهد شد.
- **Ephemeral or Anonymous Diffie-Hellman**: پارامترهای عمومی Diffie-Hellman کلاینت ارسال می‌شوند.
- **Fixed Diffie-Hellman**: پارامترهای عمومی Diffie-Hellman کلاینت در یک گواهی‌نامه ارسال شده بود و بنابراین محتویات این پیام خالی است.
- **Fortezza**: پارامترهای Fortezza کلاینت ارسال می‌شوند.

بالاخره در این فاز، کلاینت ممکن است یک **پیام certificate_verify** ارسال نماید تا تأئید صریح یک گواهی کلاینت را فراهم کند. این پیام تنها بدنبال هر گواهی‌نامهٔ کلاینت که دارای قابلیت امضاء باشد ارسال می‌شود (یعنی تمام گواهی‌نامه‌ها بجز آنهائی که دارای پارامترهای fixed Diffie-Hellman هستند). این پیام یک کُد hash را که مبتنی بر پیام‌های قبل است امضاء می‌کند و چنین تعریف می‌شود:

CertificateVerify.signature.md5_hash

MD5(master_secret || pad_2 || MD5(handshake_messages || master_secret || pad_1));

Certificate.signature.sha_hash

SHA(master_secret || pad_2 || SHA(handshake_messages || master_secret || pad_1));

که در آن pad_1 و pad_2 مقادیری هستند که قبلاً برای MAC تعریف شده‌اند، handshake_messages به تمام پیام‌های پروتکل Handshake که در شروع client_hello ارسال و دریافت شده‌اند (ولی شامل این پیام نیستند) اشاره می‌کند، و master_secret یک مقدار سرّی محاسبه شده است که نحوهٔ ساخت آن را بعداً در همین بخش خواهیم دید. اگر کلید خصوصی کاربر DSS باشد، آنگاه از آن برای رمزنگاری hash مربوط به SHA-1 استفاده خواهد شد. اگر کلید خصوصی کاربر RSA باشد، از آن برای رمزنگاری جمع رشته‌ای hashهای مربوط به MD5 و SHA-1 استفاده می‌شود. در هریک از دو مورد، مقصود تأئید مالکیت کلید خصوصی کلاینت در گواهی کلاینت است. حتی اگر کسی از گواهی‌نامهٔ کلاینت سوء استفاده نماید، او قادر نخواهد بود که این پیام را ارسال نماید.

فاز چهارم - خاتمه

این فاز برقراری یک اتصال امن را کامل می‌کند. کلاینت یک **پیام change_cipher_spec** را ارسال کرده و CipherSpec موقت را در وضع فعلی CipherSpec کپی می‌کند. توجه شود که این پیام بعنوان بخشی از پروتکل Handshake تلقی نشده بلکه با استفاده از پروتکل Change Cipher Spec ارسال می‌شود. بعد از آن کلاینت بدون فوت وقت **پیام finished** را تحت الگوریتم‌های جدید، کلیدها و مقادیر سرّی ارسال می‌کند. پیام finished تأئید می‌کند که مبادلهٔ کلید و مراحل اعتبارسنجی موفقیت‌آمیز بوده است. محتویات پیام finished جمع رشته‌ای دو مقدار hash زیر است:

MD5(master_secret || pad2 || MD5(handshake_messages|| Sender || master_secret || pad1))

SHA(master_secret || pad2 || SHA(handshake_messages || Sender || master_secret || pad1))

که در آن Sender کُدی است که مشخص می‌کند ارسال‌کننده کلاینت می‌باشد و handshake_messages کلیهٔ داده‌های مربوط به پیام‌های handshake تا این پیام، ولی نه شامل این پیام، است.

در پاسخ به این دو پیام، سِرور پیام change_cipher_spec خود را ارسال کرده و مقادیر موقت را به CipherSpec انتقال می‌دهد. سِرور سپس پیام finished را ارسال می‌کند. در این هنگام، دستداد کامل شده و کلاینت و سِرور می‌توانند به مبادلهٔ داده‌های لایهٔ کاربرد اقدام نمایند.

محاسبات رمزنگاری

دو مقولهٔ جالب توجه دیگر وجود دارند: خلق یک master secret مشترک بتوسط مبادلهٔ کلید و تولید پارامترهای رمزنگاری از روی master secret.

خلق Master Secret

master secret مشترک، یک مقدار ۴۸- بایتی (۳۸۴ بیت) یکبار مصرف است که برای این اجلاس بتوسط مبادلهٔ امن کلید تولید شده است. خلق این مقدار در دو مرحله انجام می‌شود. در ابتدا یک pre_master_secret مبادله می‌شود. سپس یک master_secret بتوسط هر دو طرف محاسبه می‌گردد. برای مبادلهٔ pre_master_secret دو امکان وجود دارد

- **RSA**: یک pre_master_secret ۴۸- بایتی بتوسط کلاینت تولید شده، بتوسط کلید RSA سِرور رمزنگاری شده و برای سِرور ارسال می‌شود. سِرور متن رمزشده را با استفاده از کلید خصوصی خود رمزگشائی کرده تا pre_master_secret را بازیابی نماید.
- **Diffie_Hellman**: هر دوی کلاینت و سِرور یک کلید عمومی Diffie-Hellman تولید می‌کنند. پس از اینکه این کلیدهای عمومی رد و بدل شدند، هریک از دو طرف محاسبات Diffie-Hellman را برای خلق pre_master_secret انجام می‌دهند.

دو طرف master_secret را چنین محاسبه می‌کنند:

master_secret = MD5(pre_master_secret || SHA('A' || pre_master_secret ||
ClientHello.random || ServerHello.random)) ||
MD5(pre_master_secret || SHA('BB' || pre_master_secret ||
ClientHello.random || ServerHello.random)) ||
MD5(pre_master_secret || SHA('CCC' || pre_master_secret ||
ClientHello.random || ServerHello.random))

که در آن ClientHello.random و ServerHello.random دو مقدار nonce هستند که در پیام‌های hello اولیهٔ مبادله شده‌اند.

تولید پارامترهای رمزنگاری

CipherSpec نیاز به یک client write MAC secret، یک server write MAC secret، یک client write key، یک server write key، یک client write IV و یک server write IV دارد که از روی master secret بهمان ترتیب محاسبه می‌شوند. این پارامترها از روی master secret و با محاسبهٔ hash آن در دنباله‌ای از بایت‌های امن با طول مناسب برای همهٔ پارامترهای لازم تولید می‌شوند.

تولید اقلام کلید از master secret از همان فرمت تولید master secret از pre-master secret تبعیت می‌کند:

```
Key_block = MD5(master_secret || SHA('A' || master_secret ||
        ServerHello.random || ClientHello.random)) ||
    MD5(master_secret || SHA('BB' || master_secret ||
        ServerHello.random || ClientHello.random)) ||
    MD5(master_secret  || SHA('CCC' || master_secret ||
        ServerHello.random || ClientHello.random)) || …
```

تا اینکه خروجی کافی تولید شده باشد. نتیجهٔ این ساختار الگوریتمی، یک تابع شبه تصادفی است. master_secret را می‌توان بذر(seed) این تابع تولید اعداد شبه تصادفی دانست. اعداد تصادفی کلاینت و سِرور را می‌توان بعنوان مقادیر salt برای پیچیده‌سازی حملات ممکن به رمز در نظر گرفت (برای بحث در مورد استفاده از salt به فصل ۹ مراجعه شود).

امنیت لایهٔ حمل و نقل (TLS)

TLS برگرفته از استانداردسازی IETF است که هدف اولیهٔ آن تولید نسخهٔ استاندارد اینترنتی SSL بوده است. TLS بعنوان یک استاندارد پیشنهادی اینترنت در RFC 2246 تعریف شده است. RFC 2246 شباهت زیادی به SSLv3 دارد. در این بخش این اختلافات را مورد بررسی قرار می‌دهیم.

Version Number

TLS Record Format همانند SSL Record Format (شکل۴-۷) بوده و میدان‌های موجود در سرآیند همان مفاهیم را دارند. تنها اختلاف در اندازهٔ version می‌باشد. برای نسخهٔ جاری TLS، MajorVersion برابر 3 و MinorVersion برابر 1 است.

کُد اعتبارسنجی پیام (MAC)

دو اختلاف بین روش‌های SSLv3 و TLS MAC وجود دارد: الگوریتم واقعی و منظر محاسبات MAC. TLS از الگوریتم HMAC که در RFC 2104 تعریف شده است استفاده می‌کند. از آنچه در فصول قبل گفته شد بخاطر آورید که HMAC چنین تعریف می‌شود:

$$HMAC_K(M) = H[(K^+ \oplus \text{opad}) \| H[(K^+ \oplus \text{ipad}) \| M]]$$

که در آن

H = تابع hash درونی (برای TLS یا MD5 و یا SHA-1 است)

M = پیام ورودی به HAMC

K^+ = کلید سرّی با 0 های اضافه شده به سمت چپ آن بطوری که نتیجه برابر طول بلوک کُد hash باشد (برای MD5 و SHA-1 طول بلوک = ۵۱۲ بیت)

ipad = بایت 0110 0011 (36 هگزادسیمال) که ۶۴ بار تکرار شود (۵۱۲ بیت)

opad = بایت 1100 0101 (5C هگزادسیمال) که ۶۴ بار تکرار شود (۵۱۲ بیت)

SSLv3 از همین الگوریتم استفاده می‌کند بجز اینکه بایت‌های لائی (padding) با کلید سرّی جمع رشته‌ای می‌شود نه اینکه با کلید سرّی که به اندازهٔ طول بلوک طویل شده است، XOR گردد. سطح امنیت هر دو روش تقریباً یکسان است.

برای TLS، محاسبات MAC میدان‌های نشان داده شده در عبارت زیر را شامل می‌شوند:

HMAC_hash(MAC_write_secret,seq_num || TLSCompressed.type ||
TLSCompressed.version || TLSCompressed.length || TLSCompressed.fragment))

محاسبهٔ MAC تمام میدان‌های پوشانیده شده بتوسط SSLv3 را پوشش داده و علاوه بر آن میدان TLSCompressed.version، که نسخهٔ پروتکل بکار گرفته شده است، را نیز می‌پوشاند.

تابع شبه‌تصادفی

TLS از یک تابع شبه‌تصادفی که PRF نامیده می‌شود برای بسط مقادیر سرّی به بلوک‌های دیتا، برای مقاصد تولید کلید یا تأئید کلید، استفاده می‌کند. هدف این است که از یک مقدار سرّی نسبتاً کوچک برای تولید بلوک‌های بزرگتر دیتا بنحوی استفاده شود که از انواع حملاتی که روی توابع hash و MAC می‌شوند در امان باشد. PRF مبتنی بر تابع بسط‌دهندهٔ دیتا بصورت زیر است (شکل ۷- ۷):

P_hash(secret,seed) = HMAC_hash(secret,A(1) || seed) ||
HMAC_hash(secret,A(2) || seed) ||
HMAC_hash(secret,A(3) || seed) || …

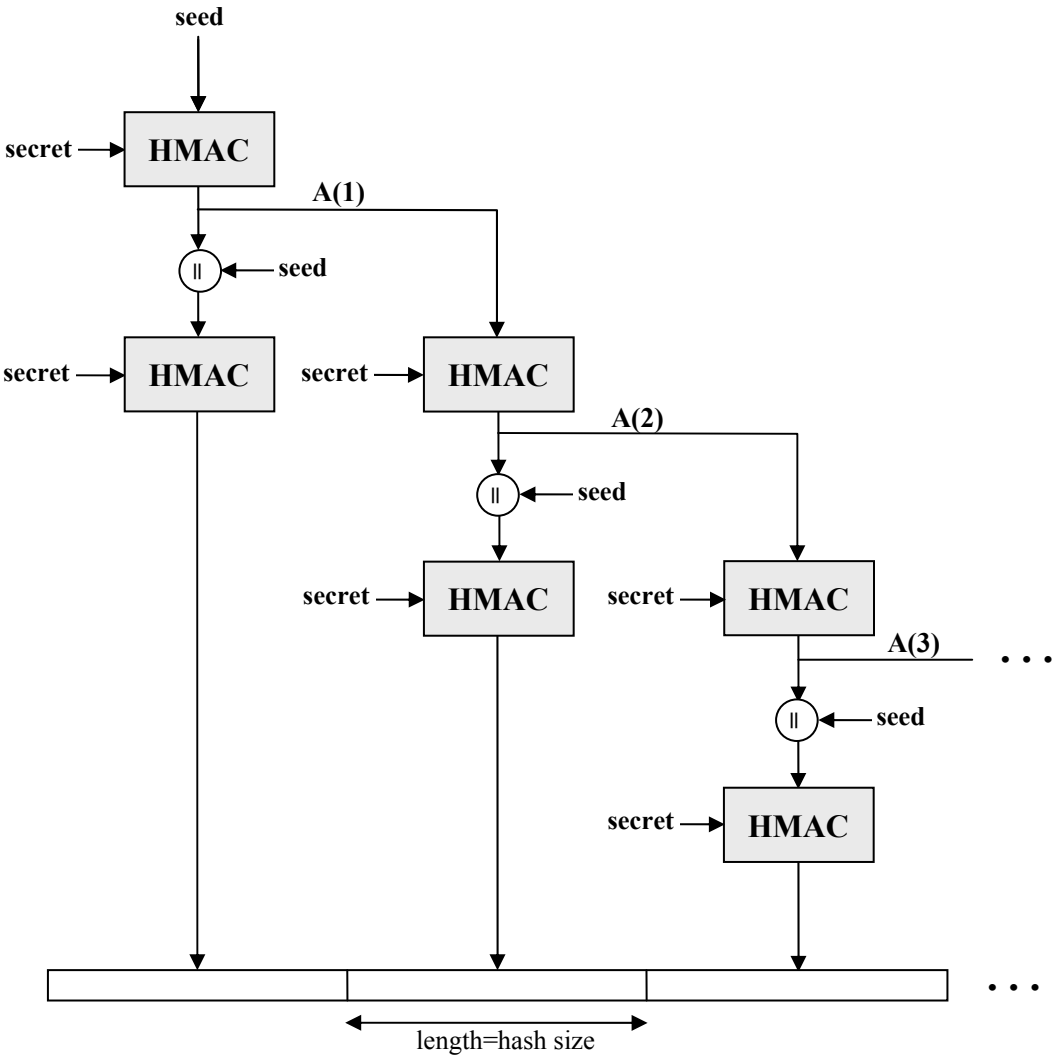
که در آن ( )A چنین تعریف می‌شود

A(0) = seed
A(i) = HMAC_hash(secret,A(i-1))

تابع بسط‌دهندهٔ دیتا از الگوریتم HMAC استفاده می‌کند که MD5 یا SHA-1 تابع hash درونی آن هستند. همانطور که می‌توان مشاهده کرد P_hash می‌تواند هرچند بار که لازم است تکرار گردد تا مقدار لازم دیتا تولید شود. برای مثال اگر از P_SHA-1 برای تولید ۶۴ بایت دیتا استفاده شود، لازم است تا چهاربار تکرار شده تا ۸۰ بایت دیتا را درست کرده که ۱۶ بایت آخر آن معدوم می‌شوند. در همین مثال P_MD5 بایستی چهار بار تکرار گردد که دقیقاً ۶۴ بایت را تولید نماید. توجه شود که هر تکرار شامل دوبار اجرای HMAC است که هرکدام بنوبهٔ خود شامل دوبار اجرای الگوریتم hash مورد استفاده است.

برای اینکه PRF تا حد امکان امن باشد از دو الگوریتم hash بنحوی استفاده می‌کند که در صورت امن ماندن هر یک از دو الگوریتم، امنیت آن تضمین شده باشد. PRF چنین تعریف می‌شود

PRF(secret, label, seed) = P_MD5(S1, label || seed) ⊕ P_SHA-1(S2, label || seed)



شکل ۷-۷ تابع P_hash (secret,seed) پروتکل TLS

PRF بعنوان ورودی یک مقدار سرّی، یک برچسب شناسائی و یک مقدار seed را گرفته و یک خروجی با طول دلخواه را تولید می‌کند. خروجی بتوسط نصف کردن مقدار سرّی به دو نیمهٔ (S1 و S2) و محاسبهٔ P_hash روی هر نیمه با استفاده از MD5 روی یک نیمه و SHA-1 روی نیمهٔ دیگر انجام می‌شود. دو نتیجه باهم XOR شده تا خروجی را درست کند. برای این منظور P_MD5 معمولاً باید بیشتر از P_SHA-1 تکرار شود تا مقدار برابری از دیتا برای ورودی تابع XOR را فراهم آورد.

کُدهای Alert

TLS تمام هشدارهای تعریف شده در SSLv3 بجز no_certificate را حمایت می‌کند. تعدادی هشدارهای اضافی نیز در TLS تعریف می‌شود که از میان آنها کُدهای زیر همیشه fatal هستند:

- **decryption_failed**: یک متن رمزشده که به روش غیرمعتبری رمزگشائی شود. یا طول متن رمزشده مضرب زوجی از طول بلوک نبوده و یا اندازهٔ لائی آن وقتی کنترل شده است صحیح نبوده است.
- **record_overflow**: یک رکورد TLS با بار آن (متن رمزشده)، دریافت شده که طول آن از ۲,۰۴۸ + $۲^{۱۴}$ بایت تجاوز کرده و یا یک متن رمزشده به طولی بیش از ۲,۰۴۸ + $۲^{۱۴}$ بایت رمزگشائی شده است.
- **unknown_ca**: یک زنجیره و یا بخشی از زنجیرهٔ گواهی‌نامه‌ها دریافت شده ولی پذیرفته نشده است، زیرا یا CA صادرکنندهٔ گواهی شناسائی نشده و یا گواهی با یک CA مورد اعتماد شناخته شده مرتبط نبوده است.
- **access_denied**: یک گواهی معتبر دریافت شده است ولی وقتی کنترل دست‌یابی به آن اعمال شده است فرستنده تصمیم گرفته است که به مذاکرات توافقی ادامه ندهد.
- **decode_error**: یک پیام نتوانسته است از کُد خارج شود زیرا یک میدان، از محدودهٔ تعیین شده خارج گشته و یا طول پیام ناصحیح بوده است.
- **export_restriction**: یک توافق که متناقض با محدودیت‌های طول کلید است، تشخیص داده شده است.
- **protocol_version**: نسخهٔ پروتکلی که کلاینت برای توافق روی آن تلاش می‌کند شناخته شده ولی مورد حمایت نیست.
- **insufficient_security**: این هشدار بجای handshake_failure وقتی ارسال می‌شود که سِرور به رمزهائی امن‌تر از آنچه بتوسط کلاینت حمایت می‌شود نیاز دارد.
- **internal_error**: یک خطای داخلی، غیرمرتبط با واحد نظیر و یا غیرمرتبط با صحت پروتکل، امکان ادامهٔ کار را نمی‌دهد.

بقیهٔ هشدارهای جدید چنین‌اند:

- **decrypt_error**: یکی از عملیات رمزنگاری دستداد موفق نبوده است که می‌تواند ناتوانی در تأئید یک امضاء، ناتوانی در رمزگشائی یک مبادلهٔ کلید و یا ناتوانی در تأئید اختتام موفقیت‌آمیز یک پیام باشد.
- **user_canceled**: این دستداد به دلیلی غیرمرتبط با مشکلات پروتکلی، ملغی شده است.
- **no_renegotiation**: بتوسط یک کلاینت در پاسخ به یک hello_request و یا بتوسط یک سِرور در پاسخ به client_hello پس از دستداد اولیه ارسال می‌شود. هریک از این دو پیام قاعدتاً منجر به توافق مجدد خواهد شد اما این هشدار نشان می‌دهد که فرستنده قادر به توافق مجدد نیست. این پیام همیشه یک اخطار است.

مجموعه‌های رمزنگاری

چند اختلاف کوچک بین مجموعه‌های رمزنگاری موجود تحت SSLv3 و TLS وجود دارد:

- **مبادلهٔ کلید**: TLS تمام تکنیک‌های مبادلهٔ کلید SSLv3 بجز Fortezza را می‌پذیرد.
- **الگوریتم‌های رمزنگاری متقارن**: TLS شامل تمام الگوریتم‌های رمزنگاری متقارن موجود در SSLv3 بجز Fortezza است.

انواع گواهی‌های کلاینت

TLS انواع گواهی‌های زیر که می‌تواند در یک پیام certificate_request تقاضا شود را تعریف می‌کند: rsa_sign ,dss_sign ,rsa_fixed_dh ,dss_fixed_dh. همه اینها در SSLv3 تعریف شده‌اند. علاوه بر آن SSLv3 شامل rsa_ephemeral_dh ،dss_ephemeral_dh و fortezza_kea می‌باشد. Ephemeral Diffie-Hellman شامل امضاء پارامترهای Diffie-Hellman با RSA یا DSS است. در TLS از rsa_sign و dss_sign برای این منظور استفاده می‌شود و یک نوع امضاء مجزا برای امضاء پارامترهایDiffie-Hellman مورد نیاز نیست. TLS روش Fortezza را شامل نمی‌شود.

پیام‌های Certificate_Verify و Finished

در پیام certificate_verify مربوط به TLS، hashهای MD5 و SHA-1 تنها روی پیام‌های handshake_messages انجام می‌شود. بیاد آورید که برای SSLv3 محاسبات hash همچنین شامل master secret و padها هم بودند. احساس شده است که این میدان‌های اضافی چیزی به امنیت اضافه نمی‌کنند.

همانند پیام finished در SSLv3، پیام finished در TLS یک مقدار hash مبتنی بر master secret مشترک، پیام‌های قبلی handshake و یک برچسب است که کلاینت یا سِرور را مشخص می‌نمایند. محاسبات قدری متفاوت‌اند. برای TLS داریم:

PRF(master_secret,finished_label,MD5(handshake_messages) || SHA-1(handshake_messages))

که در آن finished_label دنبالهٔ "client finished" برای کلاینت و "server finished" برای سِرور است.

محاسبات رمزنگاری

pre_master_secret برای TLS بهمان صورت SSLv3 محاسبه می‌شود. همانند SSLv3، master_secret در TLS بصورت یک تابع hash با ورودی‌های pre_master_secret و دو عدد تصادفی hello محاسبه می‌گردد. فرم محاسبات TLS نسبت به SSLv3 متفاوت بوده و چنین تعریف می‌شود:

```
master_secret =
    PRF(pre_master_secret, "master secret ", ClientHello.random || ServerHello.random)
```

الگوریتم آنقدر اجرا می‌شود تا ۴۸ بایت شبه‌تصادفی خروجی تولید شود. محاسبات اقلام کلید (MAC secret keys، session encryption keys، و IVها) بصورت زیر تعریف می‌شوند:

```
key_block =
        PRF(master_secret, "key expansion",
            SecurityParameters.server_random  || SecurityParameters.client_random)
```

تا خروجی کافی تولید شده باشد. همانند SSLv3 ، key_block تابعی از master_secret و اعداد تصادفی کلاینت و سِرور است ولی برای TLS الگوریتم‌ها متفاوت‌اند.

لائی (padding)

در SSL، لائی اضافه شده قبل از رمزنگاری دیتای کاربر، حداقل مقدار لازم برای کامل کردن اندازهٔ کل دیتائی که باید رمز شود، به مضربی از طول بلوک رمز است. در TLS، لائی می‌تواند هر مقدار حداکثر تا ۲۵۵ بایت باشد که کل دیتا را مضربی از طول بلوک رمز نماید. بعنوان مثال اگر متن ساده (یا متن فشرده‌سازی شده در صورت استفاده از فشرده‌سازی) بعلاوه MAC بعلاوهٔ بایت مربوط به طول لائی، ۷۹ بایت باشد، آنگاه طول لائی می‌تواند ۱ و ۹ و ۱۷ و غیره تا ۲۴۹ بایت باشد. از یک طول متغیر لائی ممکن است برای پیچیده نمودن حملاتی که مبتنی بر تجزیه و تحلیل طول‌های پیام‌های مبادله شده است، استفاده شود.

۷-۳ معاملهٔ الکترونیکی اَمن (Secure Electronic Transaction)

SET یک مدل رمزنگاری و امنیتی برای حفاظت معاملات مرتبط با کارت‌های اعتباری روی اینترنت است. نسخهٔ فعلی آن یعنی SETv1 به درخواست MasterCard و Visa برای ایجاد استانداردهای امنیتی، در فوریهٔ ۱۹۹۶ میلادی، بوجود آمد. شرکت‌های زیادی در تعیین مشخصه‌های اولیهٔ SET دخالت داشتندکه از آن جمله IBM، Microsoft، Netscape، RSA، Terisa و Verisign را می‌توان نام برد. فعالیت‌های مربوط که از سال ۱۹۹۶ آغاز شده بود پس از تست‌های فراوان باعث گردید تا در سال ۱۹۹۸ اولین موج محصولات مرتبط با SET به بازار عرضه گردد.

SET به خودی‌خود یک سیستم پرداخت نیست. بلکه SET مجموعه‌ای از پروتکل‌های امنیتی و فرمت‌هائی است که کاربران را قادر می‌سازد تا زیرساخت موجود پرداخت از طریق کارت‌های اعتباری را، از طریق یک شبکهٔ باز مثل اینترنت بصورت امن انجام دهند. فی‌الجمله SET سه سرویس را فراهم می‌آورد:

- یک کانال ارتباطی امن بین تمام طرفین درگیر در یک معامله ایجاد می‌کند.
- با استفاده از گواهی‌نامه‌های دیجیتال X.509v3، اعتماد را فراهم می‌سازد.
- محرمانگی را تضمین می‌نماید، زیرا اطلاعات تنها در زمان و مکان لازم در اختیار طرفین قرار می‌گیرد.

SET دارای مشخصه‌های پیچیده‌ای است که تعاریف آنها در ماه می ۱۹۹۷ در سه کتاب منتشر گردید:

- **کتاب اول** : توصیف کسب و کار (۸۰ صفحه)
- **کتاب دوم** : راهنمای برنامه‌نویسان (۶۲۹ صفحه)
- **کتاب سوم** : تعریف رسمی پروتکل : (۲۶۲ صفحه)

تمام اینها ۹۷۱ صفحه مشخصات را شامل می‌شود. در مقایسه، مشخصه‌های SSLv3 در ۶۳ صفحه و مشخصه‌های TLS در ۸۰ صفحه قرار دارند. بهمین مناسبت تنها خلاصه‌ای از این مشخصات همه جانبه در این بخش آورده شده است.

مروری بر SET

روش مناسبی برای شروع بحث در مورد SET این است که به انتظارات کسب و کار از SET، مشخصه‌های کلیدی آن و افراد درگیر در یک گردش کاری SET نگاهی بیندازیم.

انتظارات

کتاب اول از مشخصه‌های SET، نیازهای تجاری معاملات امن با کارت‌های اعتباری روی اینترنت و سایر شبکه‌ها را بصورت زیر به لیست درآورده است:

- **ایجاد محرمانگی در اطلاعات مربوط به سفارش کالا و پرداخت پول**: لازم است دارندگان کارت‌های اعتباری را مطمئن نمود که این نوع اطلاعات آنان محرمانه مانده و تنها در دسترس گیرنده معین خاص قرار می‌گیرد. محرمانه‌سازی اطلاعات همچنین خطر تقلب از سوی هر یک از طرفین معامله و یا شخص ثالث بداندیش را کاهش می‌دهد. برای محرمانه‌کردن اطلاعات، SET از رمزنگاری استفاده می‌کند.
- **اطمینان از صحت داده‌های انتقال یافته**: یعنی بایستی اطمینان داده شود که در خلال انتقال پیام‌های SET هیچگونه تغییری در محتوای داده‌ها بوجود نمی‌آید. برای حفظ صحت اطلاعات از امضاء دیجیتال استفاده می‌شود.
- **احراز هویت بمنظور اطمینان از اینکه یک دارندهٔ کارت اعتباری، صاحب قانونی آن است**: مکانیسمی که دارندهٔ کارت را به یک شماره حساب مشخص مرتبط می‌سازد، از موارد تقلب و هزینهٔ تمام شدهٔ پردازش پرداخت‌ها جلوگیری می‌کند. از امضاءهای دیجیتال و گواهی‌نامه‌ها برای تأئید این مطلب که دارندهٔ کارت همان صاحب قانونی یک حساب معتبر است استفاده می‌شود.
- **اعتبارسنجی یک فروشنده برای اینکه مشخص شود که او از طریق یک مؤسسهٔ مالی و اعتباری قادر به پذیرش معاملات کارت اعتباری است**: این مورد مکمل مورد قبل است. دارندگان کارت لازم است بتوانند بازرگانانی را که می‌خواهند با آنها معاملات امن داشته باشند شناسائی نمایند. بازهم در اینجا از امضاء دیجیتال و گواهی‌نامه‌ها استفاده می‌شود.
- **اطمینان استفاده از بهترین عملیات امنیتی و تکنیک‌های طراحی سیستم برای حفاظت همهٔ طرف‌های قانونی درگیر در یک معاملهٔ تجاری الکترونیک**: SET مجموعه‌ای است که بر مبنای الگوریتم‌ها و پروتکل‌های رمزنگاری بسیار امن بنا شده و امتحان خود را در این زمینه پس داده است.
- **خلق یک پروتکل که نه به مکانیسم‌های امنیت حمل و نقل وابسته بوده و نه از استفاده چنین مکانیسم‌هائی جلوگیری نماید**: SET می‌تواند بصورت امن روی یک پشته TCP/IP «خام» سوار شود. با وجود این SET در صورت استفاده از مکانیسم‌های امنیتی دیگر مثل IPSec و SSL/TLS دخالتی در کار آنها نمی‌کند.
- **تعامل بین نرم‌افزار و شبکه را تسهیل و تشویق نماید**: فرمت‌ها و پروتکل‌های SET مستقل از نوع سخت‌افزار، نوع سیستم عامل، و نرم‌افزار وِب می‌باشند.

مشخصه‌های کلیدی SET

برای رفع انتظاراتی که در بالا بیان گردید، SET تسهیلات زیر را فراهم کرده است:

- **محرمانگی اطلاعات**: اطلاعات مربوط به حساب و پرداخت‌های صاحب کارت در عبور از عرض شبکه امن می‌مانند. یک جنبهٔ قابل توجه و مهم SET این است که اجازه نمی‌دهد فروشنده از شمارهٔ کارت اعتباری خریدار مطلع گردد و این اطلاعات فقط به بانک صادرکنندهٔ کارت عرضه می‌شود. برای محرمانه‌سازی اطلاعات از رمزنگاری مرسوم DES استفاده می‌شود.

- **صحت داده‌ها**: اطلاعات پرداخت که از سوی خریدار برای فروشنده ارسال می‌شود شامل اطلاعات درخواست کالا، داده‌های شخصی و دستورات پرداخت است. SET تضمین می‌نماید که محتویات این پیام‌ها در عبور از شبکه عوض نشوند. امضاءهای دیجیتال RSA با استفاده از کُدهای hash نظیر SHA-1، صحت پیام را فراهم می‌آورد. علاوه بر آن برخی پیام‌ها بتوسط HMAC و با استفاده از SHA-1 نیز حفاظت می‌گردند.
- **اعتبارسنجی حساب دارندهٔ کارت**: SET فروشندگان را قادر می‌سازد تا تحقیق نمایند که دارندهٔ کارت، یک کاربر قانونی و صاحب یک حساب کارتی معتبر است. SET از گواهی‌های دیجیتال X.509v3 که برای این منظور دارای امضاءهای RSA هستند استفاده می‌کند.
- **تأئید هویت فروشنده**: SET دارندگان کارت‌های اعتباری را قادر می‌سازد تا هویت فروشنده را از این نظر که با یک مؤسسهٔ مالی مجاز به پذیرش پرداخت‌های کارتی در رابطه است، تعیین نمایند. SET برای این کار از گواهی‌های دیجیتال X.509v3 که برای این منظور دارای امضاءهای RSA هستند استفاده می‌کند.

توجه کنید که برخلاف IPSec و SSL/TLS، SET برای هر الگوریتم رمزنگاری فقط یک انتخاب دارد. این یک امر منطقی بوده زیرا SET یک کاربرد منفرد با مجموعهٔ خاصی از نیازهاست در حالی که IPSec و SSL/TLS برای حمایت از کاربردهای متعددی طراحی شده‌اند.

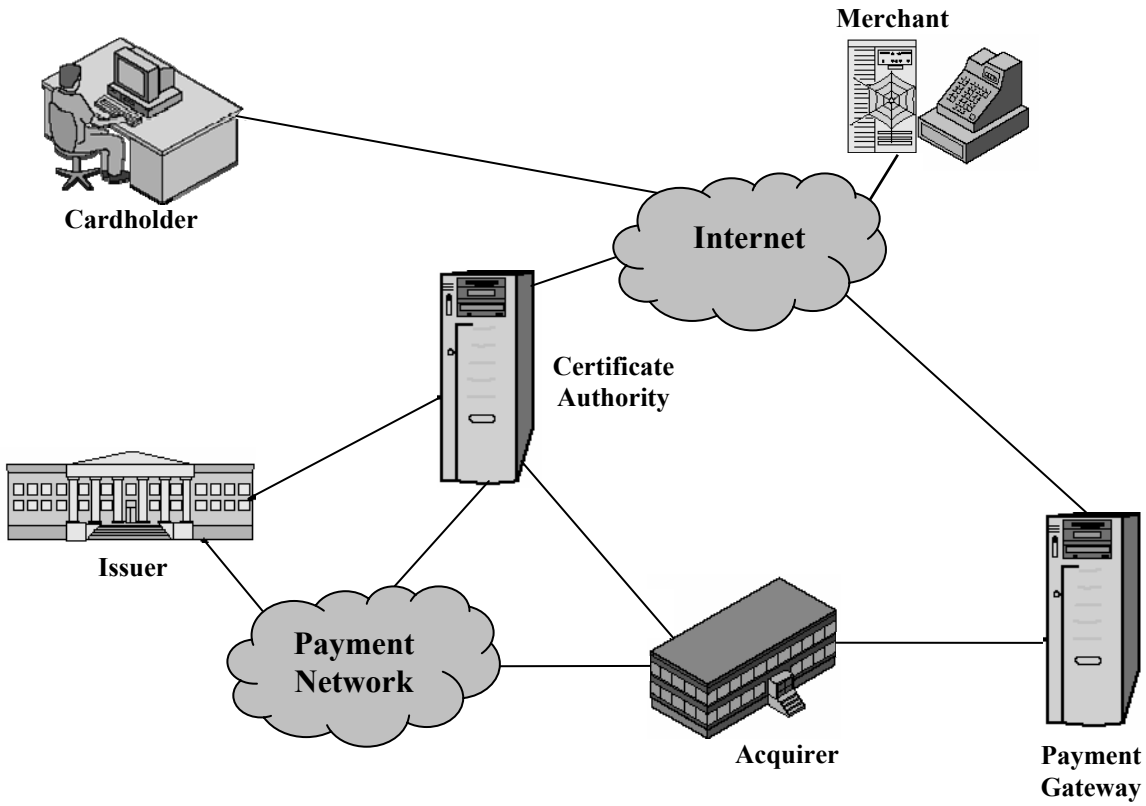
طرف‌های درگیر در SET

شکل ۸-۷ طرف‌های درگیر در یک سیستم SET را نشان می‌دهد که شامل عناصر زیراند:

- **دارندهٔ کارت**: در یک محیط الکترونیکی، مصرف‌کنندگان و خریداران با فروشندگان و تاجران از طریق کامپیوترهای شخصی متصل به اینترنت مرتبط‌اند. یک دارندهٔ کارت قاعدتاً صاحب قانونی یک کارت اعتباری (مثل Visa یا MasterCard) است که از طرف یک مؤسسهٔ مالی صادر شده است.
- **فروشنده یا تاجر**: فروشنده یک شخص و یا یک سازمان است که کالا و خدمات را برای فروش به دارندگان کارت عرضه می‌نماید. معمولاً این کالاها و سرویس‌ها از طریق یک وِب سایت و یا از طریق پست الکترونیک عرضه می‌شوند. فروشنده‌ای که پرداخت‌های کارتی را قبول می‌کند بایستی با یک مباشر ارتباط داشته باشد.
- **صادرکنندهٔ کارت**: این یک مؤسسهٔ مالی، مثل بانک، است که کارت اعتباری را برای دارندهٔ کارت صادر می‌کند. معمولاً تقاضای بازکردن چنین حساب‌های اعتباری یا حضوراً و یا از طریق پست صورت می‌پذیرد. در نهایت این صادرکنندهٔ کارت است که مسئول بازپرداخت دیون دارندهٔ کارت خواهد بود.
- **مباشر (Acquirer)**: این یک مؤسسهٔ مالی است که حسابی را با یک فروشنده برقرار کرده و مسئولیت پرداخت‌ها و تأئید حساب‌های اعتباری را بعهده دارد. تجار معمولاً بیش از یک نوع کارت اعتباری را می‌پذیرند ولی نمی‌خواهند تا با سازمان‌های متعدد بانکی و یا صادرکنندگان کارت‌های مختلف در تماس باشند. مباشر این اطمینان را برای فروشنده بوجود می‌آورد که حساب یک کارت عرضه شده معتبر و فعال بوده و میزان خرید صاحب کارت از اعتبار او تجاوز نمی‌نماید. مباشر همچنین موجبات انتقال الکترونیک وجوه پرداخت شده به حساب فروشنده را فراهم می‌آورد. به دنبال آن صادرکنندهٔ کارت مبلغ هزینه شده را بنحوی از طریق یک شبکهٔ پرداخت الکترونیک می‌پردازد.

- **دروازهٔ پرداخت**: این وظیفه‌ای است که بتوسط مباشر و یا شخص ثالث دیگری که پیام‌های مربوط به دریافت و پرداخت فروشنده را فراهم می‌آورد، انجام می‌شود. دروازهٔ پرداخت بین SET و شبکهٔ پرداخت کارتی موجود واسطه‌ای را ایجاد می‌کند که عملیات مجاز پرداخت‌ها را حمایت نماید. فروشنده پیام‌های SET را با دروازهٔ پرداخت رد و بدل کرده و دروازهٔ پرداخت از طریق یک لینک مستقیم و یا شبکه‌ای به سیستم پردازش مالی مباشر متصل است.
- **مقام مسئول گواهی‌کننده (CA)**: واحدی است که برای صدور گواهی‌نامه‌های کلید- عمومی X.509v3 برای دارندهٔ کارت، فروشنده و دروازهٔ پرداخت مورد اعتماد قرار گرفته است. موفقیت SET منوط به حضور یک زیرساخت CA برای این منظور است. همانطور که در فصول قبل بیان شد، یک ساختار سلسله مراتبی CA مورد استفاده قرار گرفته تا طرف‌های درگیر لزومی به تأئید از طرف بالاترین مقام CA را نداشته باشند.

حال بطور مختصر گردش کار یک معاملهٔ الکترونیک را بررسی می‌کنیم. سپس به برخی از جزئیات رمزنگاری SET نگاهی خواهیم انداخت.



شکل ۸-۷ عوامل معاملهٔ الکترونیکی امن

۱- **مشتری یک حساب باز می‌کند**. مشتری یک حساب کارت اعتباری، همانند Visa و یا MasterCard، در یک بانک که پرداخت الکترونیک و SET را حمایت می‌کند باز می‌کند.

۲- **مشتری یک گواهی‌نامه دریافت می‌کند**. پس از تصدیق هویت مشتری به روش مناسب، وی یک گواهی‌نامهٔ دیجیتال X.509v3 که بتوسط بانک امضاء شده است را دریافت می‌دارد. گواهی، کلید عمومی RSA مشتری و تاریخ انقضای آن را تعیین کرده است. این گواهی همچنین یک ارتباط تضمین شده بتوسط بانک بین جفت کلید مشتری و کارت اعتباری او برقرار می‌سازد.

۳- **فروشندگان گواهی‌نامه‌های خود را دارند**. فروشنده‌ای که نوع مخصوصی از کارت اعتباری را می‌پذیرد بایستی دو گواهی‌نامهٔ مختلف برای دو کلید عمومی که در اختیار اوست داشته باشد که یکی برای امضاء پیام‌ها و دیگری برای تبادل کلید است. فروشنده همچنین نیاز به یک نسخه از گواهی کلید- عمومی دروازهٔ پرداخت را دارد.

۴- **مشتری سفارش می‌دهد**. در این مرحله ممکن است لازم باشد که در ابتدا مشتری وِب سایت فروشنده را مرور نموده و قیمت کالا را بدست آورد. سپس او یک لیست از اقلامی که تمایل به خرید آنها را دارد برای فروشنده ارسال کرده و فروشنده در مقابل فرم سفارش کالا که شامل لیست اقلام، قیمت آنها، قیمت کل و شمارهٔ سفارش است را برای او برمی‌گرداند.

۵- **فروشنده تأئید می‌شود**. علاوه بر فرم سفارش، فروشنده یک نسخه از گواهی نامهٔ خود را برای خریدار ارسال می‌کند تا خریدار بتواند تأئید کند که با یک فروشندهٔ مجاز و معتبر روبروست.

۶- **سفارش و پرداخت آن تواماً ارسال می‌گردند**. مشتری سفارش خود و پرداخت مرتبط با آن را برای فروشنده ارسال می‌کند و بهمراه آن گواهی‌نامهٔ مشتری نیز فرستاده می‌شود. سفارش، خرید اقلامی را که در فرم ارسالی فروشنده آمده بود تأئید می‌کند. پرداخت شامل جزئیات مربوط به کارت اعتباری است. اطلاعات پرداخت طوری رمزنگاری می‌شودکه فروشنده قادر به خواندن آنها نباشد. گواهی‌نامهٔ مشتری، فروشنده را قادر می‌سازد تا اعتبار مشتری را تحقیق کند.

۷- **فروشنده تأئید پرداخت را درخواست می‌کند**. فروشنده اطلاعات پرداخت را به دروازهٔ پرداخت می‌فرستد و از او تأئید می‌گیرد که اعتبار مشتری برای خرید فعلی کافی است.

۸- **فروشنده سفارش را تأئید می‌کند**. فروشنده تأئید سفارش را برای مشتری ارسال می‌کند.

۹- **فروشنده محصول و یا سرویس را فراهم می‌آورد**. فروشنده،کالا را برای مشتری ارسال کرده و یا سرویس درخواستی او را فراهم می‌سازد.

۱۰- **فروشنده درخواست پرداخت می‌نماید**. این درخواست برای دروازهٔ پرداخت ارسال شده و تمام پردازش‌های مرتبط با پرداخت در آنجا انجام می‌شود.

امضاء دوگانه (Dual Signature)

قبل از اینکه به جزئیات پروتکل SET بپردازیم، اجازه دهید تا به یک نوآوری مهم که در SET فراهم آورده شده و به امضاء دوگانه معروف است اشاره نمائیم. هدف امضاء دوگانه این است که دو پیام که به مقصد دو دریافت کنندهٔ متفاوت ارسال

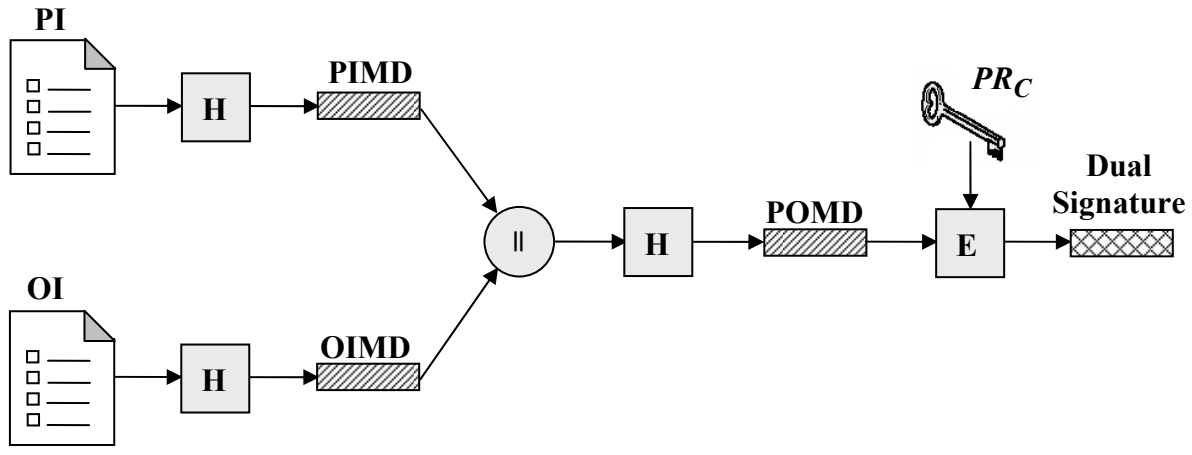
می‌شوند را بهم پیوند دهد. در این مورد، مشتری می‌خواهد که اطلاعات مربوط به سفارش کالا OI (Order Information) را برای فروشنده، و اطلاعات مربوط به قیمت آن کالا PI (Payment Information) را برای بانک ارسال نماید. فروشنده لازم نیست که شمارهٔ کارت اعتباری خریدار را بداند، و بانک هم نیازی به دانستن جزئیات سفارش کالای خریدار ندارد. از نظر خصوصی ماندن و حفاظت اطلاعات، بهتر است که این دو قلم از هم جدا باشند. از سوی دیگر این دو قلم اطلاعات بایستی طوری باهم مرتبط شوند که اگر لازم شد بتوان دعاوی آینده را پاسخ داد. این پیوند بایستی طوری باشد که مشتری بتواند اثبات کند که این پرداخت برای این سفارش، و نه سفارش یا سرویس دیگری بوده است.

برای ملاحظهٔ نیاز به این پیوند، فرض کنید که مشتریان دو پیام برای فروشنده ارسال نمایند- یک OI امضاء شده و یک PI امضاء شده - و فروشنده PI را برای بانک بفرستد. اگر فروشنده بتواند OI دیگری از خریدار دریافت کند، ممکن است ادعا نماید که این OI جدید به همراه PI بوده است و OI قدیم را نادیده انگارد. پیوند ذکرشده از این امر جلوگیری می‌کند.

شکل ۹-۷ استفاده از امضاء دوگانه (DS) برای رفع نیاز فوق‌الذکر را نشان می‌دهد. مشتری hash مرتبط با PI (با استفاده از SHA-1) و hash مرتبط با OI را استخراج می‌کند. این دو hash سپس با هم جمع رشته‌ای شده و hash نتیجهٔ آنها گرفته می‌شود. در انتها مشتری hash نهائی را با کلید خصوصی امضاء خود رمزنگاری کرده و امضاء دوگانه را تولید می‌کند. عملیات را می‌توان چنین خلاصه نمود

$$DS = E(PR_C, [H(H(PI) \,||\, H(OI)])$$

که درآن PR_C کلید خصوصی امضاء مشتری است. حال فرض کنید که فروشنده امضاء دوگانه (DS)، OI و چکیدهٔ پیام برای PI (PIMD) را در اختیار دارد. فروشنده همچنین کلید عمومی مشتری را که در گواهی نامهٔ او ذکر شده است می‌داند. بنابراین فروشنده قادر است تا دو کمیت زیر را محاسبه نماید:



PI = Payment Information
OI = Order Information
H = Hash function (SHA-1)
|| = Concatenation
PIMD = PI message digest
OIMD = OI message digest
POMD = Payment Order message digest
E = Encryption (RSA)
PR_c = Customer's private signature key

شکل ۹-۷ نحوهٔ ساخت امضاء دوگانه

$$D\ (PU_C\ ,DS)\ \text{و}\ H(PIMD \parallel H[OI])$$

که در آن PU_C کلید عمومی امضاء مشتری است. اگر این دو کمیت با هم برابر باشند، فروشنده امضاء مشتری را تأئید کرده است. بهمین ترتیب اگر بانک DS، PI، چکیدهٔ پیام برای OI (OIMD) و گواهی‌نامهٔ کلید- عمومی مشتری را داشته باشد آنگاه بانک می‌تواند دو کمیت زیر را حساب کند:

$$D\ (PU_C\ ,\ DS)\ \text{و}\ H(H[PI] \parallel OIMD)$$

بازهم اگر این دو کمیت برابر باشد، آنگاه بانک امضاء مشتری را تأئید کرده است. بطور خلاصه،

۱- فروشنده OI را دریافت کرده و امضاء را گواهی نموده است.
۲- بانک PI را دریافت کرده و امضاء را گواهی نموده است.
۳- مشتری PI و OI را بهم پیوند داده و می‌تواند ارتباط این دو با هم را اثبات کند.

بعنوان مثال، فرض کنید که فروشنده می‌خواهد به نفع خود OI دیگری را برای این پرداخت جا بزند. در این صورت او مجبور خواهد بود تا OI دیگری که تابع hash آن با OIMD موجود یکسان باشد را پیدا کند. با استفاده از SHA-1 این کار امکان‌پذیر نخواهد بود. بنابراین فروشنده نمی‌تواند OI دیگری را به این PI مرتبط سازد.

عملیات پرداخت

جدول ۳-۷ انواع گردش اسناد مرتبط با SET را نشان می‌دهد. در آنچه ذیلاً خواهد آمد، به جزئیات سه گردش مالی زیر خواهیم پرداخت:

- درخواست خرید
- تأئید پرداخت
- برداشت پول

درخواست خرید

قبل از اینکه درخواست خرید ارسال شود، دارندهٔ کارت اعتباری در وِب جستجو کرده، کالای مورد نظر خود را انتخاب نموده و سفارش را آماده کرده است. این فاز ابتدائی وقتی پایان می‌یابد که فروشنده یک فرم سفارش کالا (پیش فاکتور) را برای خریدار ارسال کرده باشد. تمام این مراحل بدون استفاده از SET صورت می‌پذیرد.

درخواست خرید شامل چهار پیام است: Initiate Request، Initiate Response، Purchase Request و Purchase Response.

برای ارسال پیام‌های SET به فروشنده، دارندهٔ کارت بایستی یک نسخه از گواهی‌نامه‌های فروشنده و دروازهٔ پرداخت را داشته باشد. خریدار در پیام **Initiate Request** که برای فروشنده ارسال می‌شود گواهی‌نامه‌ها را درخواست می‌نماید. در این پیام نوع کارت اعتباری که خریدار مایل به استفاده از آن است ذکر می‌شود. پیام همچنین شامل یک ID تخصیص داده شده برای این زوج درخواست/ پاسخ و یک nonce برای اطمینان از بهنگام بودن آن است.

جدول ۳-۷ انواع گردش اسناد SET

Cardholder registeration	دارندگان کارت قبل از این که بتوانند پیام‌های SET را برای فروشندگان بفرستند، بایستی در یک CA ثبت نام شده باشند.
Merchant registeration	فروشندگان قبل از این که بتوانند پیام‌های SET را با مشتریان و دروازه‌های پرداخت مبادله نمایند، بایستی در یک CA ثبت نام شده باشند.
Purchase Request	پیامی که از جانب مشتری برای فروشنده ارسال شده و شامل OI برای فروشنده و PI برای بانک است.
Payment authorization	مبادلهٔ بین فروشنده و دروازهٔ پرداخت تا مقدار مشخصی پول برای یک خرید را از طریق یک کارت اعتبار بخشد.
Payment capture	به فروشنده اجازه می‌دهد تا از دروازهٔ پرداخت تقاضای پول نماید.
Certificate inquiry and status	اگر CA نتواند یک درخواست گواهی را سریعاً پردازش نموده و پاسخ دهد، با ارسال یک پیام به دارندهٔ کارت اعتباری و یا فروشنده خبر می‌دهد که بعداً تماس بگیرند. دارندهٔ کارت یا فروشنده با ارسال پیام *Certificate Inquiry* از وضعیت تقاضای خود با خبر شده و اگر درخواست آنان تأئید شده باشد گواهی را دریافت خواهند کرد.
Purchase inquiry	به دارندهٔ کارت اجازه می‌دهد تا پس از دریافت پاسخ خرید، از وضعیت سفارش خود مطلع گردد. توجه شود که این پیام شامل اطلاعاتی همانند کالای فراهم شده نیست بلکه نشان‌دهندهٔ اعتبارسنجی، برداشت پول و پردازش‌های اعتباری است.
Authorization reversal	به یک فروشنده اجازه می‌دهد تا درخواست‌های قبلی خود را تصحیح کند. اگر سفارش کامل نشود، فروشنده تمام اعتبارسنجی را از سر خواهد گرفت. اگر بخشی از سفارش کامل نگردد، فروشنده آن بخش را از سر خواهد گرفت.
Capture reversal	به یک فروشنده اجازه می‌دهد تا اشتباهات احتمالی در یک درخواست پرداخت پول را که ممکن است ناشی از خطای یک منشی باشد تصحیح کند.
Credit	به یک فروشنده اجازه می‌دهد تا در صورت برگشت کالا و یا مثلاً فاسد شدن آن پولی را به حساب دارندهٔ کارت واریز نماید. توجه شود که پیام *Credit* در SET همیشه از سوی فروشنده و نه از سوی دارندهٔ کارت ارسال می‌شود. تمام ارتباطات بین دارندهٔ کارت و فروشنده که منجر به پردازش اعتبار می‌گردد خارج از SET واقع می‌شود.
Credit reversal	به یک فروشنده اجازه می‌دهد تا یک اعتبار درخواست شدهٔ قبلی را تصحیح نماید.
Payment gateway certificate request	به یک فروشنده اجازه می‌دهد تا از دروازهٔ پرداخت استعلام کرده و یک کپی از مبادلهٔ کلید جاری دروازه و گواهی امضاءها را دریافت کند.
Batch administration	به یک فروشنده اجازه می‌دهد تا اطلاعات مرتبط با معاملات او را به دروازهٔ پرداخت بفرستد.
Error message	نشان می‌دهد که یک پاسخ‌دهنده بعلت اشتباه در فرمت و یا اعتبار یک پیام از پاسخ به آن خودداری نموده است.

فروشنده یک پاسخ را تهیه کرده و آن را با کلید خصوصی امضاء خود امضاء می‌کند. پاسخ شامل nonce مشتری، nonce دیگری برای مشتری در ارسال پیام بعدی و یک ID معاملاتی مربوط به این خرید است. علاوه بر پاسخ امضاء شده، پیام **Initiate Response** شامل گواهی‌نامه امضاء فروشنده و گواهی‌نامهٔ مبادلهٔ کلید دروازهٔ پرداخت است.

دارندهٔ کارت، گواهی‌نامه‌های فروشنده و دروازه را بتوسط امضاءهای CA مرتبط با آنها تأئید کرده و آنگاه OI و PI را فراهم می‌آورد. ID معامله که بتوسط فروشنده به این معامله تخصیص یافته است، هم در OI و هم در PI منظور خواهند شد. OI بطور صریح داده‌های مربوط به سفارش مانند تعداد اقلام و قیمت هرقلم را ذکر نمی‌کند، بلکه به یک شمارهٔ سفارش اشاره می‌نماید که قبلاً در مبادلات بین خریدار و فروشنده (پیش فاکتور) ذکر شده است (قبل از اینکه وارد اولین پیام SET شویم). در مرحلهٔ بعد دارندهٔ کارت پیام **Purchase Request** (شکل ۱۰-۷) را تولید و ارسال می‌کند. برای این منظور دارندهٔ کارت یک کلید رمزنگاری متقارن یکبارمصرف K_S را تولید می‌کند. پیام شامل اطلاعات زیر است:

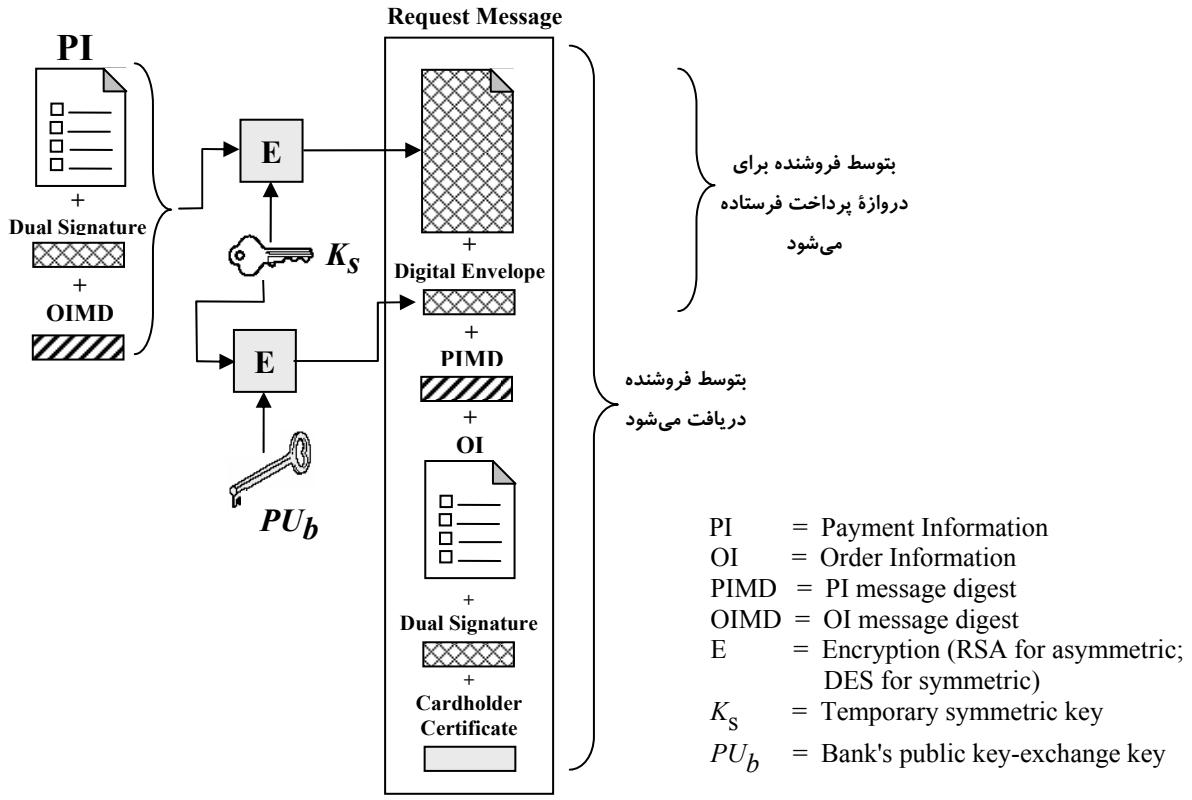
۱- **اطلاعات مرتبط با خرید**. این اطلاعات از طرف فروشنده به سمت دروازهٔ پرداخت ارسال شده و شامل اقلام زیر است

- PI
- امضاء دوگانه که از OI و PI محاسبه شده و بتوسط کلید خصوصی امضاء مشتری، امضاء شده است.
- چکیدهٔ پیام OI (OIMD)

OIMD بتوسط دروازهٔ پرداخت مورد نیاز بوده تا همانطور که قبلاً بیان شد امضاء دوگانه را تأئید نماید. تمام این اقلام با K_S رمزنگاری می‌شوند. آخرین قلم عبارت است از

- پاکت دیجیتال. این قلم با رمزنگاری K_S بتوسط کلید مبادلهٔ کلید- عمومی دروازهٔ پرداخت انجام می‌شود. آن را پاکت دیجیتال گویند زیرا قبل از اینکه سایر اقلامی که قبلاً به آنها اشاره شد خوانده شوند بایستی این پاکت باز شود (رمزگشائی شود).

اندازهٔ K_S در اختیار فروشنده قرار نمی‌گیرد. بنابراین فروشنده نمی‌تواند هیچ یک از اطلاعات مربوط به پرداخت را بخواند.



شکل ۱۰-۷ دارندهٔ کارت اعتباری درخواست خرید می‌کند

۲- **اطلاعات مرتبط با سفارش**. این اطلاعات مورد نیاز فروشنده بوده و شامل اقلام زیر است:
- OI
- امضاء دوگانه که از OI و PI محاسبه شده و بتوسط کلید خصوصی امضاء مشتری امضاء شده است.
- چکیدهٔ پیام PI (PIMD)

PIMD مورد نیاز فروشنده است تا بتواند امضاء دوگانه را تأئید نماید. توجه شود که OI بصورت رمزنشده ارسال می‌شود.

۳- **گواهی‌نامهٔ دارندهٔ کارت اعتباری**. این شامل کلید عمومی امضاء دارندهٔ کارت است که هم مورد نیاز فروشنده و هم مورد نیاز دروازهٔ پرداخت است.

وقتی فروشنده پیام Purchase Request را دریافت کرد عملیات زیر را انجام می‌دهد (شکل ۱۱-۷):

۱- گواهی‌نامه‌های دارندهٔ کارت را بتوسط امضاءهای CA تأئید می‌کند.
۲- امضاء دوگانه را با استفاده از کلید عمومی امضاء مشتری تأئید می‌کند. این مسأله این اطمینان را ایجاد می‌کند که سفارش در زمان ترانزیت دستکاری نشده و بتوسط کلید خصوصی امضاء دارندهٔ کارت امضاء شده است.
۳- پردازش مربوط به سفارش را انجام داده و اطلاعات پرداخت را جهت تأئید به دروازهٔ پرداخت می‌فرستد(بعداً توضیح داده خواهد شد).
۴- یک Purchase Response برای دارندهٔ کارت می‌فرستد.

پیام **Purchase Response** شامل یک بلوک پاسخ است که سفارش را تأئید نموده و به شمارهٔ مأخذ معامله اشاره می‌کند. این بلوک بتوسط فروشنده و با استفاده از کلید خصوصی او امضاء می‌شود. بلوک و امضاء آن به همراه گواهی امضاء فروشنده برای خریدار ارسال می‌شود.

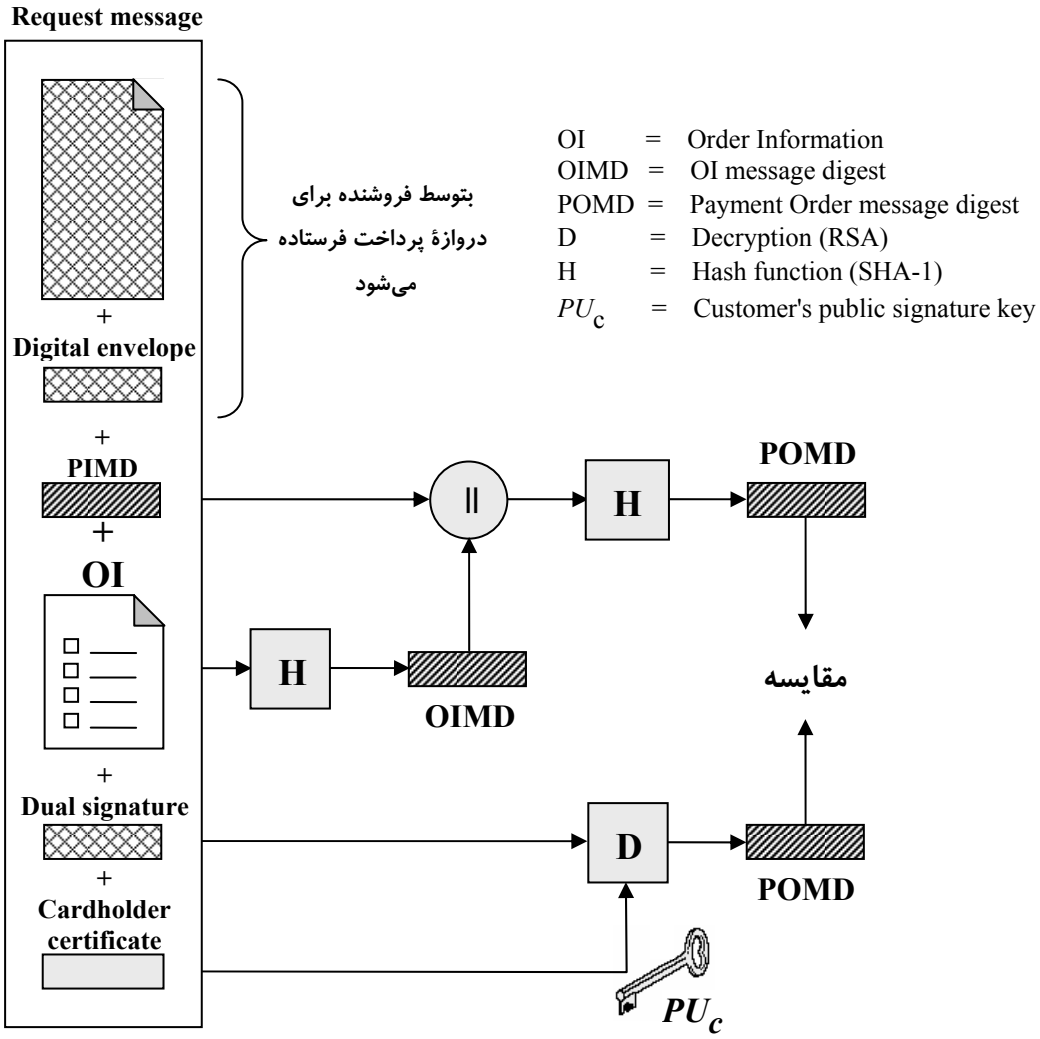
وقتی نرم‌افزار دارندهٔ کارت، پیام پاسخ خرید را دریافت کرد، گواهی‌نامهٔ فروشنده را تأئیدکرده و سپس امضاء بلوک پاسخ را تأئید می‌کند. بالاخره بر اساس پاسخ، عملیاتی همانند نمایش یک پیام برای کاربر و یا بروزرساندن یک پایگاه داده با وضعیت سفارش انجام می‌پذیرد.

تأئید پرداخت

در خلال انجام عملیات مربوط به سفارش کالای یک دارندهٔ کارت، فروشنده ضمن تماس با دروازهٔ پرداخت، معامله را تأئید می‌کند. تأئید پرداخت این اطمینان را ایجاد می‌کند که پرداخت بتوسط صادرکنندهٔ کارت مورد پذیرش است. این تأئید تضمین می‌نماید که فروشنده پول خود را دریافت خواهد کرد و بر این اساس سرویس و یا کالای لازم برای مشتری را فراهم می‌نماید. عملیات تأئید پرداخت شامل دو پیام است. درخواست تأئید و پاسخ تأئید.

فروشنده یک پیام **درخواست تأئید(Authorization Request)** را به دروازهٔ پرداخت می‌فرستدکه شامل اقلام زیر است:

۱- **اطلاعات مرتبط با خرید**. این اطلاعات از مشتری دریافت شده و شامل اقلام زیر است:
- PI
- امضاء دوگانه که از OI و PI محاسبه شده و بتوسط کلید خصوصی امضاء مشتری امضاء شده است.
- چکیدهٔ پیام OI (OIMD)
- پاکت دیجیتال



شکل ۱۱-۷ فروشنده درخواست خرید مشتری را تأئید می‌کند

۲- **اطلاعات مرتبط با تأئید**. این اطلاعات بتوسط فروشنده تولید شده و شامل:

- یک بلوک تأئید که شامل ID معامله است که بتوسط کلید خصوصی امضاء فروشنده امضاء شده و با یک کلید متقارن یکبارمصرف که بتوسط فروشنده تولید شده، رمزنگاری شده است.
- یک پاکت دیجیتال. این قلم با رمزنگاری کلید یکبارمصرف بتوسط کلید مبادلهٔ کلید عمومی دروازهٔ پرداخت تهیه شده است.

۳- **گواهی‌نامه‌ها**. فروشنده گواهی‌نامهٔ کلید امضاء دارندهٔ کارت (که برای امضاء دوگانه بکار می‌رود)، گواهی‌نامهٔ کلید امضاء فروشنده (که برای تأئید امضاء فروشنده بکار می‌رود) و گواهی‌نامهٔ مبادلهٔ کلید فروشنده (که در پاسخ دروازهٔ پرداخت لازم است) را بهمراه اقلام فوق الذکر ارسال می‌دارد.

دروازهٔ پرداخت وظایف زیر را انجام می‌دهد:

۱- همهٔ گواهی‌نامه‌ها تأئید می‌کند.
۲- پاکت دیجیتال بلوک تأئید را رمزگشائی کرده تا کلید متقارن را بدست آورده و آنگاه بلوک تأئید را رمزگشائی می‌نماید.
۳- امضاء فروشنده در بلوک تأئید را، تأئید می‌نماید.
۴- پاکت دیجیتال بلوک پرداخت را رمزگشائی نموده تا کلید متقارن را بدست آورده و آنگاه بلوک پرداخت را رمزگشائی می‌نماید.
۵- امضاء دوگانهٔ بلوک پرداخت را تأئید می‌نماید.
۶- ID معاملهٔ دریافت شده از فروشنده را با آنچه که از طرف مشتری (بطور غیرمستقیم) در PI است مقایسه و در صورت تطبیق تأئید می‌کند.
۷- از صادرکنندهٔ کارت تقاضای تأئید اعتبار نموده و آن را دریافت می‌دارد.

پس از دریافت تأئید از طرف صادرکنندهٔ کارت، دروازهٔ پرداخت یک پیام **پاسخ تأئید (Authorization Response)** را برای فروشنده ارسال می‌کند. این پیام شامل عناصر زیر است:

۱- **اطلاعات مرتبط با تأئید**. شامل یک بلوک تأئید است که بتوسط کلیدخصوصی امضاء دروازه، امضاء شده و بتوسط یک کلید متقارن یکبارمصرف که بتوسط دروازه تولید شده است، رمزنگاری شده است. این کلید همچنین شامل یک پاکت دیجیتال است که شامل کلید یکبارمصرف بوده و با کلید مبادلهٔ کلید عمومی فروشنده رمزنگاری شده است.
۲- **اطلاعات مرتبط با قبض پرداخت**. این اطلاعات در آینده برای امر پرداخت مورد استفاده قرار خواهد گرفت. این بلوک به همان فرم بند (۱) بوده یعنی یک قبض پرداخت شده رمزنگاری شدهٔ امضاءشده بهمراه یک پاکت دیجیتال است. این قبض بتوسط فروشنده مورد پردازش قرار نمی‌گیرد، بلکه بایستی با درخواست پرداخت برگردانده شود.
۳- **گواهی‌نامه**. گواهی‌نامهٔ کلید امضاء دروازهٔ پرداخت.

با تأئید موارد بتوسط دروازهٔ پرداخت، فروشنده می‌تواند کالا را برای خریدار فراهم نموده و یا سرویس مورد نیاز او را تأمین نماید.

برداشت پول

برای دریافت پول، فروشنده دروازهٔ پرداخت را در یک گردش برداشت پول درگیر کرده که شامل یک درخواست برداشت و یک پیام پاسخ برداشت است.

برای پیام **درخواست برداشت(Capture Request)**، فروشنده یک بلوک درخواست برداشت را تولید کرده، امضاء نموده و رمزنگاری می‌کند. این بلوک شامل مبلغ برداشت و ID معامله است. پیام همچنین شامل قبض برداشت رمزشده که قبلاً دریافت شده بود (در پاسخ تأئید) برای این معامله و همچنین کلید امضاء فروشنده و گواهی‌های مبادلهٔ کلید است.

وقتی دروازهٔ پرداخت، پیام درخواست برداشت را دریافت می‌کند، آن را رمزگشائی کرده و بلوک درخواست برداشت را تأئید نموده و بلوک قبض برداشت را رمزگشائی و تأئید می‌نماید. آنگاه تطابق بین درخواست برداشت با قبض برداشت را وارسی می‌نماید. سپس یک درخواست کِلر (اصطلاح بانکی معمول) تولید کرده که از طریق شبکهٔ خصوصی پرداخت، برای صادرکنندهٔ کارت ارسال می‌شود. این تقاضا باعث می‌شود که پول به حساب فروشنده واریز شود.

دروازه پرداخت آنگاه فروشنده را از پرداخت پول بتوسط یک پیام **پاسخ برداشت(Capture Response)** آگاه می‌سازد. پیام شامل یک بلوک پاسخ بوده که دروازه آن را امضاء نموده و رمزنگاری می‌کند. پیام همچنین شامل گواهی‌نامهٔ کلید امضاء دروازه می‌باشد. نرم‌افزار فروشنده، پاسخ برداشت را ذخیره کرده تا در رفع اختلاف با مباشر مورد استفاده قرار گیرد.

۷-۴ منابع مطالعاتی

[RESC01] مرور کاملی بر SSL و TLS را فراهم می‌آورد.

بهترین مأخذ مطالعهٔ جزئیات SET کتاب اول مشخصه‌ها است که در MasterCard SET Web site در دسترس است. [MACG97] نیز مطلب را بصورت عالی بررسی کرده است. [DREW99] نیز یک منبع مطالعاتی خوب است.

DREW99 Drew, G. *Using SET for Secure Electronic Commerce*. Upper Saddle River, NJ: Prentice Hall, 1999.

MACG97 Macgregor, R.; Ezvan, C.; Liguori, L.; and Han, J. *Secure Electronic Transactions: Credit Card Payment on the Web inTheory and Practice.* IBM RedBook SG244978-00, 1997. Available at **www.redbooks.ibm.com**.

RESC01 Rescorla, E. *SSL and TLS: Designing and Building Secure Systems*. Reading, MA: Addison-Wesely, 2001.

وِب سایت‌های مفید

- **Netscape's SSL Page**: شامل مشخصه‌های SSL است.
- **Transport Layer Security Charter**: آخرین RFC ها و پیش نویس‌های اینترنت در مورد TLS.
- **OpenSSL Project**: پروژهٔ تولید نرم‌افزارهای SSL و TLS. سایت شامل اسناد و لینک‌هائی در این مقوله است.

۷-۵ واژه‌های کلیدی، سؤالات مرورکنندهٔ بحث و مسائل

واژه‌های کلیدی

دروازهٔ پرداخت	payment gateway	مباشر	acquirer
معاملهٔ الکترونیکی امن	Secure Electronic Transaction (SET)	دارندهٔ کارت اعتباری	cardholder
لایهٔ سوکت امن	Secure Socket Layer (SSL)	مقام مسئول صدورگواهی‌نامه	certification authority (CA)
امنیت لایهٔ حمل و نقل	Transport Layer Security (TLS)	امضاء دوگانه	dual signature
		صادرکننده کارت اعتباری	issuer
		تاجر یا فروشندهٔ کالا	merchant

سؤالات مرورکننده بحث

۷-۱ مزایای هریک از سه روش نشان داده شده در شکل ۷-۱ کدامند؟

۷-۲ کدام پروتکل‌ها، SSL را می‌سازند؟

۷-۳ اختلاف بین یک اتصال SSL با یک اجلاس SSL چیست؟

۷-۴ پارامترهائی که حالت اجلاس SSL را تعریف می‌کنند لیست کرده و مختصراً توضیح دهید.

۷-۵ پارامترهائی که حالت اتصال SSL را تعریف می‌کنند لیست کرده و مختصراً توضیح دهید.

۷-۶ پروتکل SSL Record چه سرویس‌هائی را فراهم می‌آورد؟

۷-۷ انتقال پروتکل SSL Record شامل چه قدم‌هائی است؟

۷-۸ گروه‌های اصلی شرکت‌کننده در SET را لیست نموده و مختصراً تعریف کنید.

۷-۹ امضاء دوگانه چیست و اهداف آن کدام است؟

مسائل

۷-۱ در SSL و TLS چرا بجای اینکه تنها یک پیام change_cipher_spec در پروتکل Handshake وجود داشته باشد، یک پروتکل Change Cipher Spec جداگانه فراهم شده است؟

۷-۲ تهدیدهای زیر در مورد امنیت web را در نظر گرفته و توصیف کنید که چگونه با هر یک از آنها بتوسط یکی از مشخصه‌های SSL مقابله می‌شود.

الف- حملهٔ همه جانبهٔ شکستن رمز: یک جستجوی کامل فضای کلید برای یک الگوریتم رمزنگاری متقارن

ب- حملهٔ لغت نامه‌ای با دانستن متن ساده: بسیاری از پیام‌ها شامل متون سادهٔ قابل پیش‌بینی هستند (همانند فرمان HTTP GET). یک مهاجم یک لغت‌نامه که شامل متون رمزشدهٔ همه متن‌های سادهٔ ممکن است را تولید می‌کند. وقتی که یک پیام رمزشده مورد شنود قرار می‌گیرد، مهاجم بخشی را که شامل متن سادهٔ معلوم رمزنگاری شده است گرفته و در لغت نامه به دنبال متن رمزشده می‌گردد. متن رمزشده بایستی با یکی از اقلام موجود در لغت‌نامه که با همان کلید سّری رمزنگاری شده است تطبیق کند. اگر تطبیق در چند مورد صورت پذیرد، هر مورد با کل متن رمزشده مقایسه شده تا نتیجهٔ صحیح به دست آید. این حمله علی‌الخصوص در مورد اندازهٔ کوچک کلیدها (مثلاً کلید ۴۰- بیتی) مؤثر است.

ج- حملهٔ بازخوانی: پیام‌های handshake قدیمی دوباره اجرا شوند.

د - حملهٔ Man-in-the-Middle: یک مهاجم در هنگام مبادلهٔ کلید، خود را در مسیر ارتباطات قرار داده و برای سِرور بصورت کلاینت و برای کلاینت بصورت سِرور ظاهر می‌شود.

هـ- password sniffing: کلمات عبور در HTTP و یا سایر کاربردها مورد استراق سمع قرار می‌گیرد.

و- IP Spoofing: با استفاده از آدرس‌های IP تقلیدی یک میزبان را گول زده تا دیتای عوضی را بپذیرد.

ز- IP Hijacking: یک اتصال فعال معتبر بین دو میزبان، گسسته شده و حمله‌کننده جای یکی از طرفین را اشغال می‌کند.

ح- SYN Flooding: یک مهاجم پیام‌های TCP SYN را ارسال کرده تا یک اتصال را درخواست نماید ولی به پیام انتهائی جواب نمی‌دهد تا اتصال بطور کامل برقرار شود. مدول TCP مورد تهاجم معمولاً حدود چند دقیقه «اتصال نیمه باز» را نگاه می‌دارد. پیام‌های تکراری SYN می‌تواند مدول TCP را مسدود کند.

۷-۳ بر اساس آنچه در این فصل آموخته‌اید، آیا در SSL ممکن است که گیرنده، بلوک‌های SSL record را که خارج از نظم وارد می‌شوند به نظم درآورد. اگر چنین است توضیح دهید که چگونه چنین چیزی ممکن است؟ اگر نه چرا نه؟

فصل ۸

امنیت مدیریت شبکه

۸-۱ مفاهیم اساسی SNMP

معماری مدیریت شبکه
معماری پروتکل مدیریت شبکه
پروکسی‌ها
SNMPv2

۸-۲ تسهیلات جامعه‌ای SNMPv1

جوامع و نام‌های جوامع
سرویس اعتبارسنجی
خط‌مشی دست‌یابی
سرویس پروکسی

۸-۳ SNMPv3

معماری SNMP
پردازش پیام و مدل امنیتی کاربر
کنترل دست‌یابی مبتنی بر منظر

۸-۴ منابع مطالعاتی

۸-۵ واژه‌های کلیدی، سؤالات مرورکنندهٔ بحث و مسائل

واژه‌های کلیدی
سؤالات مرورکنندهٔ بحث
مسائل

شبکه‌ها و سیستم‌های عملیاتی توزیع شده از اهمیت حیاتی و فزاینده‌ای در کسب و کار، دولت و سایر سازمان‌ها برخوردارند. در داخل یک سازمان بخصوص، رَوَند کار به سمت شبکه‌های پیچیده‌تر که کاربردها و کاربران بیشتری را پشتیبانی می‌کنند در حرکت است. همین‌طور که این شبکه‌ها رشد بیشتری می‌یابند، دو واقعیت ملموس‌تر میشود:

- شبکه و منابع مرتبط با آن و کاربردهای توزیع شده از ضروریات لاینفک سازمان می‌شوند.
- رویدادهای بیشتری ممکن است باعث خطا شده و شبکه یا بخشی از شبکه را از کار انداخته و یا عملکرد آن را به سطح غیرقابل قبولی تنزل دهد.

یک شبکهٔ بزرگ نمی‌تواند تنها بتوسط تلاش‌های انسانی، جمع و جور و مدیریت شود. پیچیدگی چنین سیستمی، استفاده از ابزارهای خودکار مدیریتی را ایجاب می‌کند. اگر شبکه شامل تجهیزاتی از سازندگان مختلف باشد، نیاز به چنین ابزارهائی افزایش یافته و تهیهٔ این ابزارها نیز مشکل می‌شود. در پاسخ به این نیازها، استانداردهائی که مرتبط با مدیریت شبکه هستند تهیه شده که سرویس‌ها، پروتکل‌ها و پایگاه‌های اطلاعات مدیریتی را پوشش می‌دهند. تا زمان حاضر پراستفاده‌ترین استانداردی از این نوع، پروتکل سادهٔ مدیریت شبکه SNMP (Simple Network Management Protocol) بوده است.

از زمان انتشار آن در سال ۱۹۸۸ میلادی، SNMP در تعداد روزافزونی از شبکه‌ها و محیط‌های پیچیده مورد استفاده واقع شده است. همین‌طور که استفاده از SNMP گسترش یافت، نیاز به کارآئی‌های جدید برای پوشش نیازهای جدید نیز هویدا گشت. همچنین اهمیت ایجاد یک قابلیت امنیتی بعنوان بخشی از مدیریت شبکه آشکارتر گردید. در جهت ایجاد کارآئی بیشتر، نسخهٔ دوم SNMP تعریف شد. قابلیت‌های امنیتی فراگیرتر در SNMPv3 فراهم آمد.

این فصل تسهیلات امنیتی مقدماتی در SNMPv1 را توصیف کرده و سپس به خصوصیات امنیتی بسیار گسترده‌تری که در SNMPv3 وجود دارد می‌پردازد.

۸-۱ مفاهیم اساسی SNMP

این بخش مروری بر چهارچوب اصلی SNMP دارد.

معماری مدیریت شبکه

یک سیستم مدیریت شبکه، مجموعه‌ای از ابزارها برای پائیدن و کنترل شبکه است که از جنبه‌های زیر یکپارچه‌اند:

- یک واسط اپراتوری منفرد، با مجموعه‌ای از فرامین قوی ولی آشنا با کاربر، برای انجام اکثر وظایف مدیریتی شبکه.
- میزان حداقلی از تجهیزات مجزا. یعنی بیشتر سخت‌افزارها و نرم‌افزارهای لازم برای مدیریت شبکه در داخل تجهیزات موجود کاربر جای داده شده‌اند.

یک سیستم مدیریت شبکه، شامل برخی سخت‌افزارها و نرم‌افزارهای اضافی است که به مؤلفه‌های موجود شبکه اضافه شده‌اند. نرم‌افزار استفاده شده برای انجام وظایف مدیریت شبکه، در دل کامپیوترهای میزبان و عوامل ارتباطی (مثل پردازشگرهای خط اول، کنترل‌کنندهٔ ترمینال‌ها) جای دارد. یک سیستم مدیریت شبکه طوری طراحی شده است که تمام شبکه را بصورت یکپارچه نگریسته، هر نقطهٔ از آن را با آدرس‌ها و برچسب‌هایش شناخته و صفات آن نقطه و ارتباطش با کل شبکه را درک کند. عناصر فعال شبکه، یک بازخورد منظم از اطلاعات مربوط به وضعیت شبکه را برای مرکز کنترل شبکه فراهم می‌آورند.

مدل مدیریت شبکه که در SNMP از آن استفاده می‌شود شامل عناصر کلیدی زیر است:

- ایستگاه مدیریت
- عامل مدیریت
- پایگاه اطلاعات مدیریت
- پروتکل مدیریت شبکه

ایستگاه مدیریت (management station) معمولاً یک دستگاه متکی به خود است، ولی ممکن است یک قابلیت پیاده‌سازی شده در یک سیستم اشتراکی باشد. در هر صورت، ایستگاه مدیریت واسط بین مدیریت انسانی شبکه با سیستم مدیریت شبکه است. ایستگاه مدیریت حداقل دارای مؤلفه‌های زیر است:

- یک مجموعه از برنامه‌های کاربردی مدیریتی برای تحلیل داده‌ها، بازیابی از خطا و غیره
- یک واسط که بتوسط آن مدیر شبکه بتواند شبکه را کنترل کرده و بپاید
- قابلیت ترجمه نیازهای مدیر شبکه به پایش واقعی و کنترل عناصر دور در شبکه
- یک پایگاه داده از اطلاعات استخراج شده تمام واحدهای مدیریت شده در شبکه

تنها دو مؤلفهٔ آخر، موضوع استانداردسازی SNMP را تشکیل می‌دهند.

عنصر فعال دیگر در سیستم مدیریت شبکه، **عامل مدیریت** (management agent) است. تجهیزات کلیدی مانند میزبان‌ها، پل‌ها، مسیریاب‌ها و هاب‌ها ممکن است با SNMP طوری تجهیز شوند که بتوان از یک ایستگاه مدیریت آنها را اداره نمود. عامل مدیریت به سؤالات اطلاعاتی یک ایستگاه مدیریت جواب داده، به درخواست‌های عملیاتی ایستگاه مدیریت واکنش نشان داده و ممکن است بطور غیرهمزمان اطلاعات درخواست نشده ولی مهم را برای ایستگاه مدیریت ارسال نماید.

برای مدیریت منابع در شبکه، هر منبع بصورت یک موضوع (object) نشان داده می‌شود. یک موضوع نوعاً یک متغیر اطلاعاتی است که وجهی از عامل مدیریت شده را نشان می‌دهد. به مجموعهٔ موضوعات، **پایگاه اطلاعات مدیریت** (management information base (MIB می‌گویند. MIB بصورت مجموعه‌ای از نقاط دست‌یابی در عامل مدیریت، برای ایستگاه مدیریت عمل می‌کند. موضوعات در عرض سیستم‌های یک کلاس بخصوص استاندارد شده‌اند (مثلاً همهٔ پل‌ها یک نوع موضوعات مدیریتی را حمایت می‌کنند). یک ایستگاه مدیریت، عمل پائیدن شبکه را با اخذ موضوعات MIB انجام می‌دهد. یک ایستگاه مدیریت می‌تواند باعث شود که عملی در یک عامل مدیریت صورت پذیرفته و یا می‌تواند پیکربندی یک عامل را، با تغییر اندازهٔ موضوعات مشخص، تغییر دهد.

ایستگاه مدیریت و عوامل مدیریت بتوسط **پروتکل مدیریت شبکه** (network management protocol) بهم پیوند می‌خورند. پروتکل استفاده شده برای مدیریت شبکه‌های TCP/IP، پروتکل سادهٔ مدیریت شبکه SNMP است. این پروتکل شامل قابلیت‌های کلیدی زیر است:

- **Get**: ایستگاه مدیریت را قادر می‌سازد تا اندازه‌های موضوعات در عامل مدیریت را بدست آورد.
- **Set**: ایستگاه مدیریت را قادر می‌سازد تا اندازه‌های موضوعات در عامل مدیریت را تنظیم کند.
- **Notify**: یک عامل مدیریت را قادر می‌سازد تا ایستگاه مدیریت را از پیشامدهای قابل توجه خبردار سازد.

معماری پروتکل مدیریت شبکه

در سال ۱۹۸۸ میلادی، مشخصه‌هایSNMP منتشر گردید و بسرعت بصورت استاندارد غالب مدیریت شبکه درآمد. تعدادی از فروشندگان، ایستگاه‌های کاری منفرد مدیریت شبکه مبتنی بر SNMP را عرضه نموده و بیشتر فروشندگان پل‌ها، مسیریاب‌ها، ایستگاه‌های کاری و PCها، بسته‌های نرم‌افزاری عامل SNMP که محصولات آنان را قادر به اعمال مدیریت از سوی ایستگاه مدیریت می‌کند، به مشتریان عرضه می‌دارند.

همانطور که از نام آن پیداست، SNMP یک ابزار ساده برای مدیریت شبکه است. این ابزار یک پایگاه اطلاعاتی مدیریت(MIB) از متغیرهای عددی و جداول دوبعدی که محدود و بسهولت قابل پیاده‌سازی است را تعریف می‌کند و همچنین یک پروتکل روان برای اینکه یک مدیر بتواند متغیرهای MIB را بدست آورده و تنظیم کند، و همچنین یک عامل بتواند یادآوری‌های درخواست نشده بنام *traps* صادر کند را بوجود می‌آورد. قدرت SNMP در این سهولت نهفته است. SNMP بسهولت پیاده‌سازی شده و از منابع شبکه و پردازش‌گر در حد متوسط استفاده می‌کند. همچنین ساختار پروتکل و MIB بحد کافی سرراست بوده و باعث می‌شود که بین ایستگاه‌های مدیریت و نرم‌افزارهای عامل سازندگان مختلف، تعامل خوبی برقرار باشد.

سه مشخصهٔ زیربنائی عبارتند از:

- **ساختار و شناسائی اطلاعات مدیریت برای شبکه‌های مبتنی برTCP/IP (RFC 1155)**: نحوهٔ تعریف موضوعات مدیریت شده در MIB را توصیف می‌کند.
- **پایگاه اطلاعات مدیریت برای اینترنت‌های مبتنی بر TCP/IP: MIB-II (RFC 1213)**: موضوعات مدیریت شده در MIB را توصیف می‌کند.
- **پروتکل سادهٔ مدیریت شبکه (RFC 1157)**: پروتکل استفاده شده برای مدیریت این موضوعات را تعریف می‌کند.

SNMP بصورت یک پروتکل سطح کاربرد طراحی گردیده که بخشی از مجموعهٔ پروتکلی TCP/IP است. هدف از طراحی، عملکرد SNMP در بالای پروتکل UDP (User Datagram Protocol) بوده است که در RFC 768 تعریف شده است. برای یک ایستگاه مدیریتی منفرد، یک پروسهٔ مدیریتی، دست‌یابی به MIB مرکزی در ایستگاه مدیریت را کنترل کرده و یک واسط برای مدیریت شبکه را فراهم می‌آورد. پروسهٔ مدیریتی، مدیریت شبکه را با استفاده از SNMP که در بالای UDP، IP و پروتکل‌های نظیر وابسته به شبکه (مثل Ethernet, FDDI, X.25) هستند انجام می‌دهد.

هر عامل نیز بایستی SNMP، UDP و IP را پیاده‌سازی نماید. علاوه برآن یک پروسهٔ عامل وجود دارد که پیام‌های SNMP را تعبیر نموده و MIB عامل را کنترل می‌نماید. برای هر دستگاه عامل که سایر کاربردها همانند FTP را پشتیبانی می‌کند، هم TCP و هم UDP مورد نیاز است.

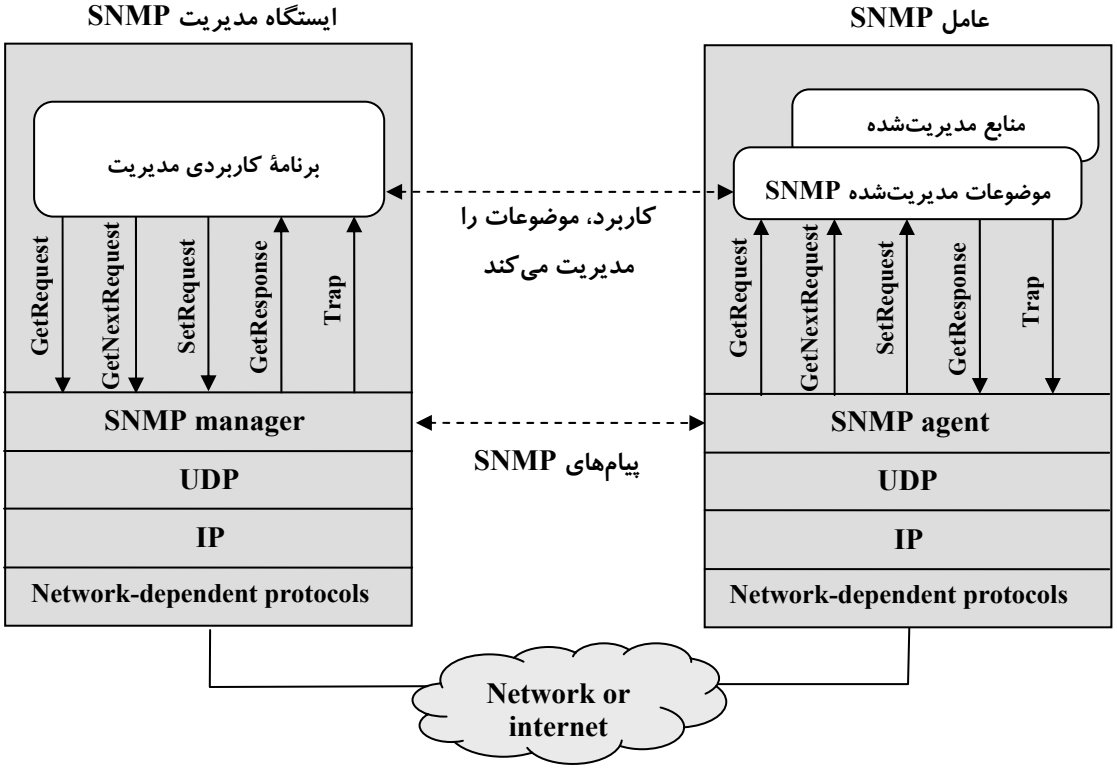
شکل ۱-۸ بستر پروتکل SNMP را نشان می‌دهد. از یک ایستگاه مدیریت، سه نوع پیام SNMP از جانب کاربردهای مدیریتی صادر می‌شود: GetRequest، GetNextRequest و SetRequest. دو پیام اول نوعی از تابع get هستند. هرسه نوع پیام بتوسط عامل و با پیام GetResponse تأئید می‌گردند که به کاربرد مدیریتی بالاتر ارجاع می‌شود. علاوه برآن، یک عامل ممکن است که یک پیام trap در پاسخ به پیشامدی که MIB و منابع مدیریت شده زیرمجموعه را تحت تأثیر قرار می‌دهد صادر کند.

چون SNMP به UDP که یک پروتکل غیراتصالی است متکی است، خود SNMP نیز غیراتصالی است. هیچ اتصالی بین یک ایستگاه مدیریت و عامل‌های آن برقرار نمی‌شود، بلکه هر مبادله یک ارتباط مجزا بین یک ایستگاه مدیریت و عامل آن است.

پروکسی‌ها

در SNMPv1 تمام عامل‌ها و همچنین خود ایستگاه‌های مدیریت بایستی UDP و IP را پشتیبانی نمایند. این امر مدیریت مستقیم بر بعضی دستگاه‌ها را محدود نموده و دستگاه‌های دیگری همچون بعضی پل‌ها و مُودم‌ها که هیچکدام از بخش‌های TCP/IP را پشتیبانی نمی‌نمایند از دُور خارج می‌سازد. علاوه برآن ممکن است سیستم‌های کوچک بسیاری (کامپیوترهای شخصی، ایستگاه‌های کاری، کنترل‌کننده‌های قابل برنامه‌ریزی) وجود داشته باشند که برای کارهای خود، TCP/IP را پیاده‌سازی کرده ولی تمایلی ندارند که بار اضافی SNMP، منطق عامل و نگهداری MIB را بعهده داشته باشند.

برای پشتیبانی دستگاه‌هائی که پیاده‌سازی SNMP را در خود ندارند، مقولهٔ پروکسی(proxy) خلق گردید. در این روش یک عامل SNMP بصورت پروکسی (وکیل) برای یک یا چند دستگاه عمل می‌کند. یعنی عامل SNMP به وکالت از طرف دستگاه‌های پروکسی شده، رفتار می‌کند.



شکل ۱- ۸ نقش SNMP

شکل ۲-۸ نوع معماری پروتکل درگیر را نشان می‌دهد. ایستگاه مدیریت سؤالاتی که در مورد دستگاه خاصی دارد را به عامل پروکسی خود می‌فرستد. عامل پروکسی، هر سؤال را به پروتکل مدیریتی بکار رفته بتوسط آن دستگاه تبدیل می‌کند. وقتی عامل پروکسی پاسخ سؤال را دریافت کرد، آن را برای ایستگاه مدیریت ارسال می‌دارد. بطریق مشابه اگر یک یادآوری یا اخطار از هر نوع از سوی دستگاه به پروکسی انتقال یابد، پروکسی آن را بصورت یک پیام trap برای ایستگاه مدیریت خواهد فرستاد.

SNMPv2 نه تنها استفاده از مجموعهٔ پروتکلی TCP/IP بلکه استفاده از سایر انواع پروتکل‌ها را نیز اجازه می‌دهد. SNMPv2 علی‌الخصوص برای اجرا روی بستهٔ پروتکلی OSI طراحی شده است. بنابراین SNMPv2 می‌تواند تعداد متنوع‌تری از پیکربندی‌های شبکه را مدیریت نماید. در رابطه با پروکسی‌ها، هر دستگاهی که پیاده‌سازی SNMPv2 را ندارد تنها از طریق یک پروکسی می‌تواند مدیریت شود. این امر حتی شامل دستگاه‌های SNMPv1 هم می‌شود. یعنی اگر یک دستگاه، نرم‌افزار عامل SNMPv1 را در پیاده‌سازی خود داشته باشد، تنها از طریق یک دستگاه پروکسی که عامل SNMPv2 و نرم‌افزار مدیریت SNMPv1 را داشته باشد می‌تواند به مدیر SNMPv2 دسترسی یابد.

مواردی که در بالا به آنها اشاره شد را روابط پروکسی خارجی در SNMPv2 گویند. علاوه برآن SNMPv2 از یک ارتباط پروکسی بومی که در آن دستگاه‌های پروکسی شده، SNMPv2 را حمایت می‌کنند پشتیبانی می‌کند. در مورد اخیر، یک مدیر SNMPv2 با یک گره SNMPv2 که بعنوان یک عامل کار می‌کند ارتباط برقرار می‌کند. این گره آنگاه بصورت یک مدیر برای دسترسی به دستگاه پروکسی شده عمل می‌کند که بعنوان عامل SNMPv2 کار خواهدکرد. علت پشتیبانی از چنین ارتباط غیرمستقیمی این است که کاربران را قادر سازد تا سیستم مدیریت شبکه‌های سلسله مراتبی و غیرمتمرکز را، چنان که بعداً توضیح داده خواهد شد، پیکربندی نمایند.

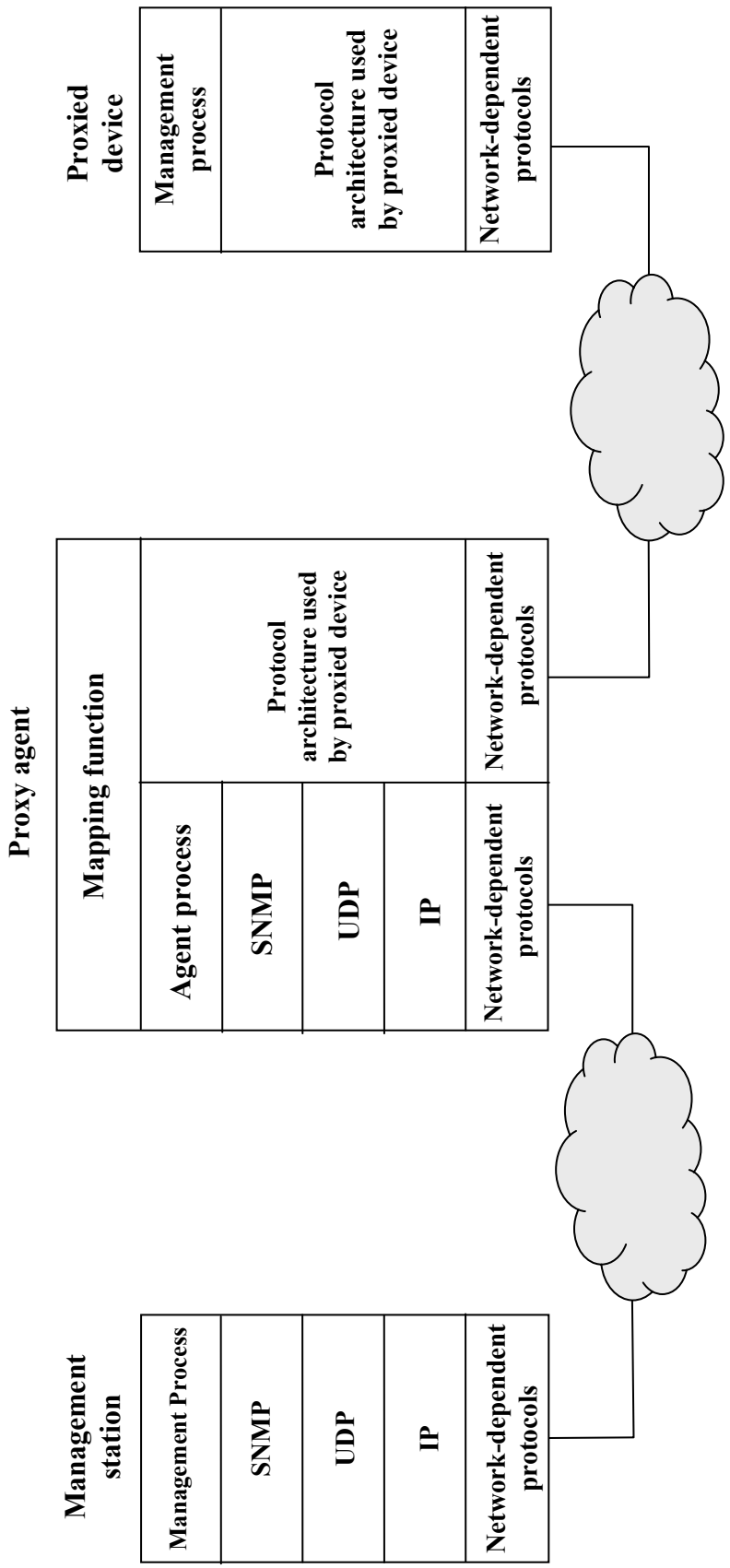
SNMPv2

قدرت SNMP در سادگی آن است. SNMP یک مجموعه پایه از ابزارهای مدیریت شبکه در یک بستهٔ نرم‌افزاری که بسهولت پیاده‌سازی و پیکربندی می‌شود را فراهم می‌آورد. از سوی دیگر چون کاربران برای مدیریت شبکه‌هائی که روز به روز توسعه یافته و بار کاری آنها افزایش می‌یابد هر روز بیشتر از دیروز به SNMP روآورده‌اند، کمبودهای آن کاملاً آشکار شده است. این کمبودها در سه گروه قرار دارند:

- عدم پشتیبانی از مدیریت توزیع‌شدهٔ شبکه
- نواقص عملیاتی
- نواقص امنیتی

کمبودهای اول و دوم در SNMPv2 مورد توجه قرار گرفتند که در سال ۱۹۹۳ منتشر شد و نسخهٔ تجدیدنظرشدهٔ آنهم در سال ۱۹۹۶ تهیه گردید (در حال حاضر RFCهای 1901, 1904 تا 1908، 2578 و 2579). SNMPv2 بسرعت مورد استقبال قرار گرفت و تعدادی از فروشنگان تنها چندماه پس از انتشار آن، محصولات منطبق با این استاندارد را به بازار فرستادند. نواقص امنیتی در SNMPv3 مورد توجه قرار گرفته است.

در بقیهٔ این بخش بطور مختصر مشخصه‌های جدید فراهم آمده بتوسط SNMPv2 را بررسی می‌کنیم. خصوصیات امنیتی SNMPv1 و SNMPv3 در بخش‌های بعدی توصیف خواهند شد.



شکل ۲-۸ پیکربندی پروکسی

مدیریت توزیع‌شدهٔ شبکه

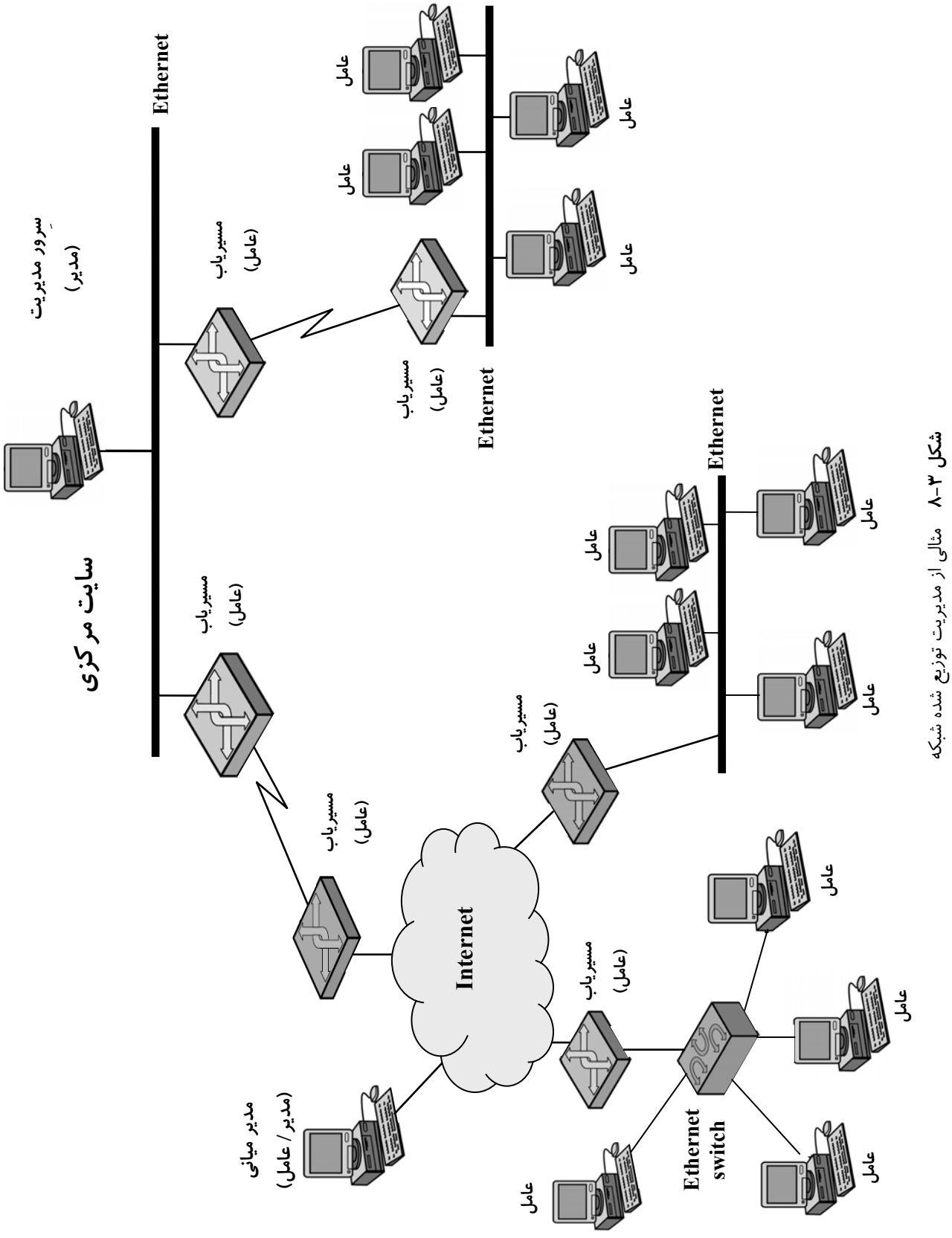
در روش سنّتی مدیریت متمرکز شبکه، یکی از میزبان‌ها در پیکربندی شبکه نقش ایستگاه مدیریت را دارد. ممکن است یک یا دو ایستگاه مدیریت دیگر نیز بعنوان پشتیبان ایستگاه اصلی عمل نمایند. مابقی تجهیزات شبکه دارای نرم‌افزارهای عامل و یک MIB هستند که پایش و کنترل از جانب ایستگاه مدیریت را امکان‌پذیر می‌نمایند. وقتی اندازهٔ شبکه‌ها بزرگ شده و بار ترافیکی آنها افزایش می‌یابد، چنین سیستم متمرکزی کارآئی نخواهد داشت. در این حالت فشار زیادی روی ایستگاه مدیریت وارد آمده و ترافیک پرحجمی ایجاد می‌شود که ناشی از گزارشات عوامل مدیریت بوده که تلاش می‌کنند تا سراسر شبکه را طی کرده و خود را به محل استقرار ایستگاه مدیریت برسانند. در چنین وضعیتی یک روش غیرمتمرکز توزیع شده، عملکرد بهتری دارد (شکل ۳-۸). در یک روش غیرمتمرکز مدیریت شبکه، ممکن است ایستگاه‌های مدیریتی سطح بالای متعددی وجود داشته باشند که به آنها سِرورهای مدیریت گویند. هر یک از چنین سِرورهائی ممکن است بطور مستقیم بخشی از عوامل را اداره نمایند ولی برای بسیاری از عامل‌ها، سِرور مدیریت مسئولیت خود را به یک مدیر میانی می‌سپارد. مدیر میانی، وظیفهٔ مدیریت برای پایش و کنترل عوامل تحت مسئولیت خود را دارد. مدیر میانی برای مدیر بالادست نقش یک عامل را بازی کرده، اطلاعات مورد نیاز او را فراهم نموده و دستورات او را می‌پذیرد. این نوع معماری مشکلات پردازش‌های مدیریتی را کم کرده و ترافیک کل شبکه را کاهش می‌دهد.

SNMPv2، هم از یک استراتژی کاملاً متمرکز و هم از یک استراتژی توزیع‌شده پشتیبانی می‌کند. در مورد اخیر بعضی سیستم‌ها هم نقش مدیر و هم نقش عامل را دارند. در نقش عامل، چنین سیستمی فرامین را از یک سیستم مدیریت مافوق می‌پذیرد. برخی از این فرامین مربوط به MIB محلی در عامل هستند. سایر فرامین به این امر نیاز دارند که عامل بعنوان یک پروکسی دستگاه‌های دور عمل کند. در این مورد، عامل پروکسی نقش مدیر برای دست‌یابی به اطلاعات یک عامل دور را داشته و آنگاه نقش یک عامل را از جهت ردکردن اطلاعات به مدیر مافوق خواهد داشت.

ارتقاء قابلیت‌های عملیاتی

جدول ۱-۸ قابلیت‌های ارتقاء یافتهٔ در SNMPv2 را نشان می‌دهد. هر دو پروتکل SNMP برحسب یک سری فرامین تعریف شده‌اند که بصورت واحدهای دیتای پروتکلی (PDU) منتقل می‌گردند. در مورد SNMPv1 پنج فرمان وجود دارد. یک مدیر، فرمان Get را برای یک عامل می‌فرستد تا اندازه موضوعات در MIB را برای او بفرستد. فرمان GetNext از این حقیقت استفاده می‌کند که موضوعات در یک MIB بصورت درختی سازمان داده شده‌اند. وقتی در یک فرمان GetNext از یک موضوع نام برده می‌شود، عامل، موضوع بعدی در درخت را پیدا کرده و اندازهٔ آن را برای مدیر برمی‌گرداند. GetNext از این جهت مفید است که به یک مدیر اجازه می‌دهد تا یک درخت، در محل عامل را، جهت یافتن موضوعات «به پیماید». فرمان Set به مدیر اجازه می‌دهد تا اندازه‌های موجود در عامل را به روزرسانی کرده و برای خلق و حذف ردیف‌های جداول از آن استفاده نماید. یک عامل از فرمان GetResponse برای پاسخ دادن به فرمان یک مدیر استفاده می‌کند. بالاخره فرمان Trap یک عامل را قادر می‌سازد تا بدون اینکه منتظر درخواست مدیریت شود، اطلاعات را برای مدیر بفرستد. بعنوان مثال یک عامل می‌تواند طوری پیکربندی شود تا در صورت پاره شدن یک لینک و یا افزایش ترافیک از آستانهٔ مشخصی، یک فرمان trap ارسال کند.

SNMPv2 تمام فرامین موجود در SNMPv1 را داشته و علاوه برآن دارای دو فرمان اضافی است. فرمان مهم‌تر، فرمان Inform است. این فرمان بتوسط یک ایستگاه مدیریت برای یک ایستگاه دیگر ارسال شده و همانند trap، اطلاعات مربوط به حالت‌ها یا پیشامدهای ایجادشده در یک فرستنده است. حسن فرمان Inform این است که از آن می‌توان در ساخت یک پیکربندی، برای تقسیم وظایف مدیریتی بین چند مدیر، در یک شبکهٔ بزرگ استفاده کرد.



شکل ۳-۸ مثالی از مدیریت توزیع شده شبکه

جدول ۱-۸ مقایسهٔ PDUهای SNMPv1 و SNMPv2

SNMPv1 PDU	SNMPv2 PDU	جهت ارسال	توصیف
GetRequest	GetRequest	مدیر به عامل	درخواست اندازه برای هر موضوع لیست شده
GetNextRequest	GetNextRequest	مدیر به عامل	درخواست اندازهٔ بعدی برای هر موضوع لیست شده
ــــــ	GetBulkRequest	مدیر به عامل	درخواست اندازه‌های متعدد
SetRequest	SetRequest	مدیر به عامل	تنظیم اندازه برای هر موضوع لیست شده
ــــــ	InformRequest	مدیر به مدیر	انتقال اطلاعات درخواست نشده
GetResponse	Response	عامل به مدیر یا مدیربه مدیر (SNMPv2)	پاسخ به درخواست مدیر
Trap	SNMPv2-Trap	عامل به مدیر	انتقال اطلاعات درخواست نشده

فرمان جدید دیگر، GetBulk است که به یک مدیر اجازه می‌دهد تا بلوک بزرگی از داده‌ها را درخواست نماید. فرمانGetBulk علی‌الخصوص برای انتقال تمام جداول، بتوسط یک فرمان منفرد، طراحی شده است.

یک اختلاف نهائی: فرمان Get در مورد SNMPv1 یکپارچه بوده در حالی که در مورد SNMPv2 یکپارچه نیست. اگر فرمان Get در SNMPv1 شامل لیستی از موضوعات باشد که اندازهٔ آنها درخواست شده است و حداقل یکی از این موضوعات در محل عامل موجود نباشد، تمام فرمان پذیرفته نخواهد شد. برای SNMPv2 چنین نبوده و بخشی از نتایج می‌تواند بازگشت داده شود. فرمان Get غیریکپارچه، استفادهٔ بهره‌ورتری از ظرفیت شبکه بتوسط مدیر را اجازه می‌دهد.

۸-۲ تسهیلات جامعه‌ای SNMPv1

SNMPv1 برابر آنچه که در RFC 1157 تعریف شده است، تنها شامل یک امکان امنیتی ابتدائی مبتنی برمفهوم جامعه (community) است. این امکان، سطح معینی از امنیت را ایجاد کرده ولی به حملات مختلف آسیب‌پذیر است [CERT02,JIAN02].

جوامع و نام‌های جوامع

همانند سایر کاربردهای توزیع شده، مدیریت شبکه از تعامل تعدادی موجودیت‌های کاربردی که از سوی یک پروتکل کاربردی حمایت می‌شوند، تشکیل می‌شود. در مورد SNMP، موجودیت‌های کاربردی مدیرها و عامل‌هائی هستندکه از SNMP استفاده می‌کنند.

مدیریت شبکه SNMP دارای چندین مشخصه است که شبیه کاربردهای توزیع شدهٔ دیگر نیستند. SNMP شامل یک رابطهٔ یک- به- چند بین یک مدیر و مجموعه‌ای از عوامل است: مدیر قادر است که موضوعات موجود در عوامل را بدست آورده و تنظیم کند. او همچنین قادر است که trapها را از عوامل دریافت دارد. بنابراین از نظر عملیاتی یا کنترلی، مدیر تعدادی از عوامل را «مدیریت می‌کند». همچنین ممکن است تعدادی مدیر وجود داشته باشدکه هریک از آنها تمام و یا زیرمجموعه‌ای از عوامل را پیکربندی و مدیریت نمایند. این زیرمجموعه‌ها ممکن است هم‌پوشانی داشته باشند.

بهمین ترتیب بایستی قادر باشیم تا شبکهٔ مدیریت SNMP را بصورت یک رابطهٔ یک- به- چند از طرف یک عامل با چند مدیر نگاه کنیم. هر عامل، MIB محلی خود را کنترل کرده و بایستی قادر باشد استفاده از آن MIB بتوسط تعدادی مدیر را کنترل کند. این کنترل دارای سه جنبه است:

- **سرویس اعتبارسنجی**: عامل ممکن است علاقه‌مند باشد تا دست‌یابی به MIB را تنها به مدیریت‌های معتبر اجازه دهد.
- **خط‌مشی دست‌یابی**: عامل ممکن است علاقه‌مند باشد تا امتیازات دست‌یابی متفاوتی به مدیران مختلف تخصیص دهد.
- **سرویس پروکسی**: یک عامل ممکن است بعنوان وکیل (پروکسی) سایر عوامل عمل نماید. این امر ممکن است شامل پیاده‌سازی سرویس اعتبارسنجی و/ یا خط‌مشی دست‌یابی برای سایر عوامل، در سیستم پروکسی باشد.

تمام این جنبه‌ها مرتبط با مسائل امنیتی هستند. در محیطی که مسئولیت مؤلفه‌های شبکه تقسیم‌بندی شده‌اند (مثلاً بین تعدادی واحدهای مدیریتی)، عامل‌ها نیازمند حفاظت خود و MIBهای خود از دست‌یابی‌های ناخواسته/ غیرمعتبر می‌باشند. SNMP برابر آنچه در RFC 1157 تعریف شده است، تنها یک امکان ابتدائی و محدود از چنین امنیتی با نام جامعه را فراهم می‌آورد.

یک **جامعهٔ SNMP (community)**، یک رابطهٔ بین یک عامل SNMP با جمعی از مدیران SNMP است که اعتبارسنجی، کنترل دست‌یابی و مشخصه‌های پروکسی را تعریف می‌کند. جامعه یک مفهوم محلی است که در محل یک عامل تعریف می‌شود. عامل برای هر ترکیب مطلوب از اعتبارسنجی، کنترل دست‌یابی و مشخصه‌های پروکسی، یک جامعه را تعریف می‌کند. به هر جامعه، یک نام یکتا (در درون این عامل) داده شده و مدیران متعلق به این جامعه بایستی از نام این جامعه در تمام فرامین get و set خود استفاده نمایند. یک عامل می‌تواند چندین جامعه را تشکیل دهد بطوری که عضویت مدیران در این جوامع هم‌پوشانی هم داشته باشند.

چون جوامع بطور محلی در یک عامل تعریف می‌شوند، عوامل مختلف ممکن است از نام واحدی استفاده کنند. یکسان بودن این نام‌ها مهم نبوده و نشان‌دهندهٔ هیچ شباهتی بین جوامع تعریف شده نیستند. بنابراین یک مدیر بایستی سابقهٔ نام و یا نام‌های مرتبط با هر عاملی که مایل به تماس با آن است را داشته باشد.

سرویس اعتبارسنجی

هدف سرویس اعتبارسنجی SNMPv1 این است که به گیرنده اطمینان دهد که یک پیام SNMPv1 از همان منبعی صادر شده است که ادعا می‌کند. SNMPv1 تنها یک روش ابتدائی برای احراز هویت را فراهم می‌آورد. هر پیام (تقاضای get یا put) از یک مدیر به یک عامل، شامل نام یک جامعه است. این نام بصورت یک کلمهٔ عبور عمل کرده و اگر فرستنده کلمهٔ عبور را بداند، معتبر تلقی خواهد شد.

با چنین فرم محدودی از اعتبارسنجی، بسیاری از مدیران شبکه حاضر نیستند که چیزی بجز پایش شبکه، یعنی عملیات get و trap را اجازه دهند. کنترل شبکه از طریق عمل set، طبیعتاً امر حساس‌تری است. نام جامعه می‌تواند آغازکنندهٔ یک رَویهٔ اعتبارسنجی باشد بشرط اینکه از نام فقط بصورت ابزار اولیه جستجوی کلمهٔ عبور استفاده شود. رَویهٔ اعتبارسنجی می‌تواند شامل مراحل پیچیده‌تری مثل رمزنگاری/رمزگشائی برای تأمین امنیت بیشتر باشد. این مسائل ورای قابلیت‌های RFC 1157 قرار دارد.

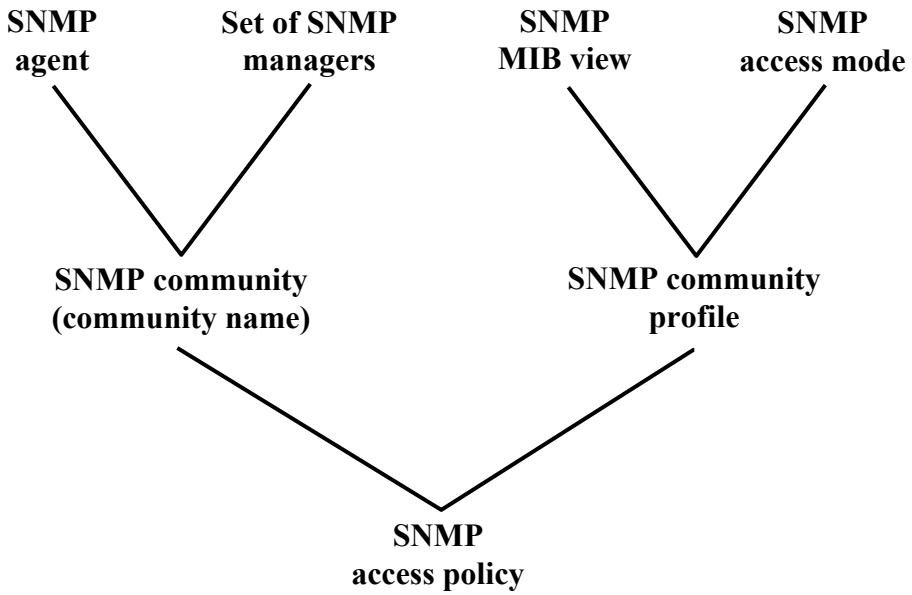
خط‌مشی دست‌یابی

با تعریف یک جامعه، یک عامل، دست‌یابی به MIB خود برای تعدادی از مدیران را محدود می‌کند. با استفاده از بیش از یک جامعه، عامل می‌تواند گروه‌های دست‌یابی مختلف برای مدیران مختلف تعریف کند. این کنترل دست‌یابی دو جنبه دارد:

- **منظر** MIB **در** SNMP: منظر(view) زیرمجموعه‌ای از موضوعات، در درون یک MIB است. منظرهای MIB مختلف ممکن است برای هر جامعه تعریف شود. مجموعهٔ موضوعات در یک منظر نیازی نیست که به یک زیرشاخهٔ منفرد MIB تعلق داشته باشد.
- **مُود دست‌یابی SNMP**: یک عنصر از مجموعه {READ-ONLY,READ-WRITE}است. یک مُود دست‌یابی برای هر جامعه تعریف می‌شود.

ترکیب یک منظرMIB و مُود دست‌یابی را یک **پروفایل جامعهٔ SNMP** گویند. بنابراین پروفایل یک جامعه شامل یک زیرمجموعهٔ تعریف‌شده از MIB در یک عامل، باضافهٔ یک مُود دست‌یابی به آن موضوعات است. مُود دست‌یابی SNMP بطور یکنواخت به تمام موضوعات در MIB view اعمال می‌گردد. بنابراین اگر مُود دست‌یابی READ-ONLY انتخاب شود، به تمام موضوعات منظر اعمال شده و مدیران را محدود می‌سازد: دسترسی به این منظر فقط READ-ONLY است.

با هر جامعهٔ تعریف شده بتوسط یک عامل، یک پروفایل جامعه مرتبط است. ترکیب یک جامعهٔ SNMP و یک پروفایل جامعهٔ SNMP را **خط‌مشی دست‌یابی SNMP** خوانند. شکل۴-۸ مفاهیمی را که مورد بحث قرار گرفت نشان می‌دهد.



شکل ۴-۸ مفاهیم مدیریتی SNMPv1

سرویس پروکسی

مفهوم جامعه در پشتیبانی سرویس پروکسی نیز مفید است. بخاطر آورید که یک پروکسی یک عامل SNMP است که به وکالت از طرف سایر دستگاه‌ها عمل می‌کند. معمولاً سایر دستگاه‌ها بیگانه هستند، یعنی TCP/IP و SNMP را پشتیبانی نمی‌کنند. در برخی موارد سیستم‌های پروکسی شده ممکن است SNMP را پشتیبانی کنند ولی پروکسی برای بحداقل رساندن تعامل بین دستگاه پروکسی شده و سیستم‌های مدیریت شبکه بکار می‌رود

برای هر دستگاهی که پروکسی وکیل آن است، یک خط‌مشی دست‌یابی SNMP تعریف می‌شود. بنابراین پروکسی می‌داند که کدام موضوعات MIB می‌توانند برای مدیریت سیستم‌های پروکسی شده (MIB view) و مُودهای دست‌یابی آن بکار روند.

۸-۳ SNMPv3

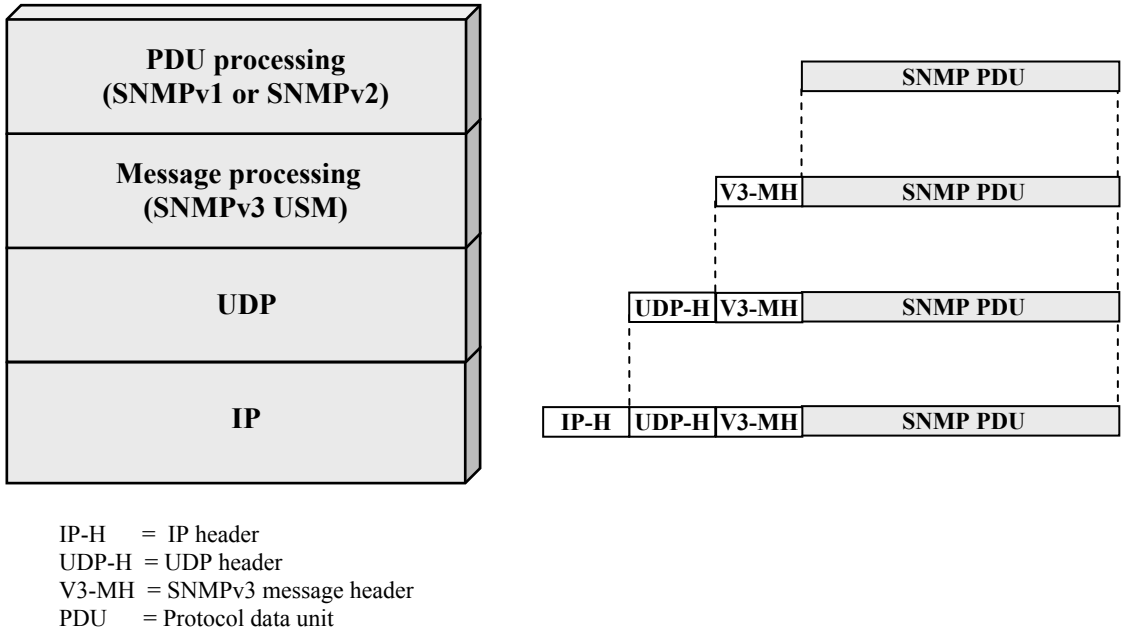
در سال ۱۹۹۸ میلادی، گروه کاری IETF SNMPv3 یک مجموعه از استانداردهای پیشنهادشده برای اینترنت که در حال حاضر RFC 2570 تا RFC 2576 را تشکیل می‌دهند فراهم نمود. این مجموعهٔ اسناد، یک چهارچوب برای بکارگیری قابلیت‌های امنیتی در مشخصه‌های کلی عملیات SNMPv1 یا SNMPv2 را شامل می‌شود. علاوه براین، اسناد یک مجموعه از قابلیت‌ها برای امنیت شبکه و کنترل دست‌یابی را فراهم می‌آورند.

توجه به این نکته مهم است که SNMPv3 یک جانشین برای SNMPv1 و/یا SNMPv2 نیست. SNMPv3 یک قابلیت امنیتی را تعریف می‌کند که می‌تواند بهمراه SNMPv2 (ترجیحاً) و یا SNMPv1 بکار رود. علاوه برآن، RFC 2571 یک معماری که در آن تمام نسخه‌های جاری و آتی SNMP می‌گنجند را توصیف می‌کند. RFC 2575 نیز یک امکان کنترل دست‌یابی را تعریف می‌کند که هدف آن این است که بطور مستقل از قابلیت هستهٔ SNMPv3 عمل کند. در این بخش یک نگاه کلی به RFC های 2570 تا 2576 نموده و توانائی‌های تعریف شده در آنها را بررسی می‌کنیم.

شکل ۵-۸ رابطهٔ میان نسخه‌های مختلف SNMP از نظر فرمت بکار رفته را نشان می‌دهد. اطلاعات بین یک ایستگاه مدیریت و یک عامل مدیریت به شکل یک پیام SNMP مبادله می‌شود. پردازش‌های مرتبط با امنیت در سطح پیام انجام می‌شود. بعنوان مثال SNMPv3 یک مدل امنیتی کاربر(User Security Model (USM که از میدان‌های موجود در سرآیند پیام استفاده می‌کند را مشخص می‌نماید. محمولهٔ یک پیام SNMP، یک PDU از SNMPv1 یا SNMPv2 است. یک PDU نمایش‌دهندهٔ یک عمل مدیریتی (مانند get یا set در مورد یک موضوع تحت مدیریت) است که با آن لیستی از نام متغیرهای مربوط به آن عمل همراه است.

RFCهای 2570 تا 2576 معماری کلی بعلاوهٔ ساختار پیام‌های خاص و خصوصیات امنیتی آنها را توصیف کرده ولی فرمت جدیدSNMP PDU را تعریف نمی‌کند. در نتیجه در داخل معماری جدید بایستی از فرمت‌های PDU نسخه‌های SNMPv1 یا SNMPv2 استفاده کرد. یک نوع پیاده‌سازی که از آن با عنوان SNMPv3 یاد می‌شود، شامل خصوصیات امنیتی و معماری تعریف شده در RFCهای 2570 تا 2576 بعلاوهٔ فرمت PDU و کارآئی‌های تعریف شده در اسناد SNMPv2 است. این موضوع در RFC 2570 چنین ذکر شده است: «SNMPv3 را می‌توان یک SNMPv2 باضافهٔ قابلیت‌های امنیتی و مدیریتی اضافی دانست.»

بقیهٔ این بخش چنین سازمان‌دهی شده است. ابتدا معماری SNMP که در RFC 2571 تعریف شده است را بطور مختصر معرفی می‌کنیم. سپس امکانات محرمانگی و اعتبارسنجی فراهم آمده بتوسط مدل امنیتی کاربر(USM) در SNMPv3 توصیف می‌گردد. بالاخره کنترل دست‌یابی و مدل کنترل دست‌یابی (VACM) را معرفی خواهیم کرد.



شکل ۵-۸ معماری پروتکل SNMP

معماری SNMP

معماری SNMP، برابر آنچه در RFC 2571 توصیف شده است، شامل مجموعه‌ای از موجودیت‌های توزیع‌شده و متعامل SNMP است. هر موجودیت، بخشی از قابلیت‌های SNMP را ایجاد کرده و ممکن است بصورت یک گرهٔ عامل، یک گرهٔ مدیریت و یا ترکیبی از این دو عمل نماید. هر موجودیت SNMP، شامل یک مجموعه از مدول‌هائی است که با هم تعامل کرده تا سرویس‌ها را فراهم آورند. این تعامل‌ها را می‌توان بصورت مجموعه‌ای از متغیرهای ابتدائی و پارامترهای انتزاعی مدل نمود.

RFC 2571 یک نیاز کلیدی طراحی برای SNMPv3 را مشخص می‌کند: یک معماری پودمانی طوری طراحی کنید که (۱) پیاده‌سازی روی محدودهٔ وسیعی از محیط‌های عملیاتی را اجازه دهد که برخی از آنها نیاز به عملکرد حداقل و ارزان قیمت داشته و برخی دیگر ممکن است دارای قابلیت‌های بیشتر برای مدیریت شبکه‌های بزرگ باشند، (۲) در مسیر استانداردسازی امکان انتقال بخش‌هائی از معماری به جلو، در صورت عدم تطبیق همهٔ بخش‌ها با هم، وجود داشته باشد و (۳) قابل سازگاری با مدل‌های امنیتی دیگر باشد.

موجودیت SNMP

هر موجودیت(entity) SNMP شامل یک موتور (engine) SNMP است. یک موتور SNMP عملیات ارسال و دریافت پیام‌ها، اعتبارسنجی و رمزنگاری/ رمزگشائی پیام‌ها و کنترل دست‌یابی به موضوعات مدیریت شده را پیاده‌سازی می‌نماید. این عملیات برای سرویس دادن به یک یا چند کاربرد، بتوسط موتور SNMP، پیکربندی شده و یک موجودیت SNMP را تشکیل می‌دهند.

هم موتور SNMP و هم کاربردهای پشتیبانی شده بتوسط این موتور، بصورت مجموعه‌ای از مدول‌های گسسته تعریف شده‌اند. این نوع معماری مزایای متعددی دارد. اول این که نقش موجودیت SNMP بتوسط مدول‌هائی که در آن پیاده‌سازی شده‌اند تعریف می‌شود. تعداد مشخصی از مدول‌ها، مورد نیاز یک عامل SNMP بوده در حالیکه تعداد مشخص دیگری (ممکن است هم‌پوشانی هم وجود داشته باشد) مورد نیاز یک مدیر SNMP هستند. ثانیاً ساختار پودمانی (مدولار) مشخصه‌ها باعث می‌شود که بتوان نسخه‌های مختلفی از هر مدول را تعریف کرد. این بنوبهٔ خود باعث می‌شود که (۱) قابلیت جایگزین کردن و یا ارتقاء توانائی‌ها برای جنبه‌های معینی از SNMP، بدون نیاز به عبور به نسخهٔ استاندارد شدهٔ بالاتر (مثلاً SNMPv4)، ایجاد گردد و (۲) بطور روشنی روش‌های هم‌زیستی، و استراتژی‌های عبور تعیین شود(RFC 2576).

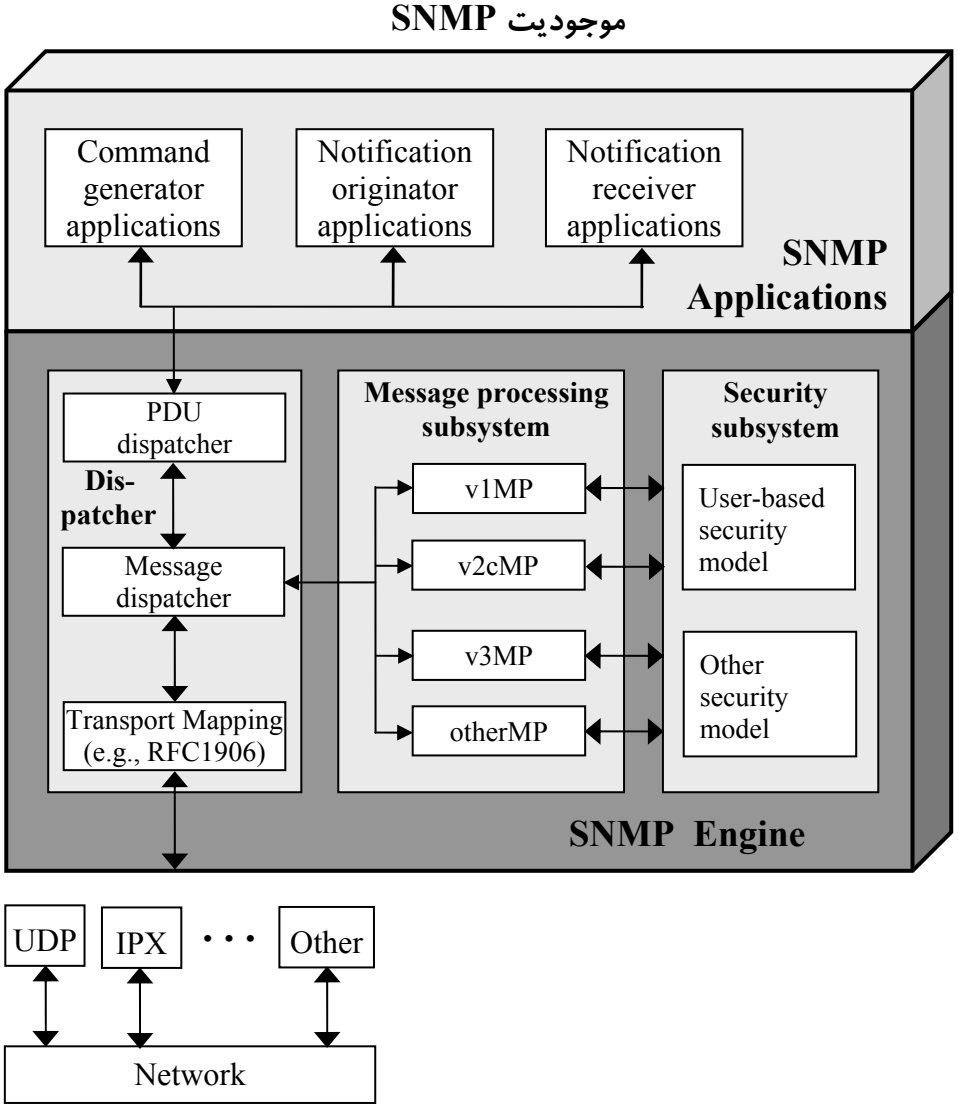
برای درک بهتر نقش هر مدول و رابطهٔ آن با مدول‌های دیگر، بهترین روش این است که به نحوهٔ استفاده از آنها در مدیرها و عامل‌ها در SNMP سنّتی توجه کنیم. اصطلاح *سنّتی(traditional)* که معادل *خالص(pure)* است از این جهت بکار رفته تا به این واقعیت اشاره کند که یک پیاده‌سازی لزومی ندارد حتماً کار یک مدیر خالص و یا یک عامل خالص را انجام دهد بلکه می‌توان مدول‌هائی را در کنار هم جمع کرد که هم وظایف مدیر و هم وظایف عامل را انجام دهند.

شکل ۶-۸ که برپایهٔ تصویر ارائه شده در RFC 2571 بناشده است بلوک دیاگرام یک **مدیر سنّتی یا خالص SNMP** را نشان می‌دهد. یک مدیر خالص SNMP با صدور فرمان‌هائی(get و set) با عوامل SNMP تعامل نموده و پیام‌های trap را دریافت می‌دارد. مدیر همچنین می‌تواند با سایر مدیران با صدور Inform Request PDU تعامل نموده، هشدارها (alerts) را فراهم کرده و یا فرامین Inform Response PDU را دریافت کند که در حقیقت تأئید دریافت فرامین Inform Request می‌باشند. در فرهنگ اصطلاحات SNMPv3، یک مدیرخالص SNMP شامل سه گروه از کاربردهاست.کاربرد مولد فرمان، داده‌های مدیریت در عامل‌های دور را پائیده و تغییرات لازم در آنها را بوجود می‌آورد. آنها از PDUهای SNMPv1 و/یا SNMPv2 که شامل Get، GetNext، GetBulk و Set است استفاده می‌کنند. یک کاربرد مولد اخطار، آغازگر پیام‌های آسنکرون است. در مورد یک مدیر خالص، Inform Request PDU برای این منظور بکار می‌رود. یک کاربرد گیرندهٔ اخطار، پیام‌های آسنکرون ورودی را پردازش می‌کند. اینها شاملPDUهای Inform Request، SNMPv2-Trap و SNMPv1-Trap هستند.

تمام کاربردهائی که در بالا توصیف شدند از سرویس‌های فراهم شده بتوسط موتور SNMP برای این موجودیت استفاده می‌کنند. موتور SNMP دو وظیفهٔ جمعی را انجام می‌دهد:

- PDUهای خارج‌شونده از کاربردهای SNMP را می‌پذیرد، پردازش‌های لازم را که شامل واردکردن کُدهای اعتبارسنجی و رمزنگاری است انجام داده و سپس PDUها را برای انتقال بصورت یک پیام، کپسولی می‌نماید.
- پیام‌های SNMP واردشونده را از لایهٔ حمل‌ونقل تحویل می‌گیرد، پردازش‌های لازم شامل اعتبارسنجی و رمزگشائی را انجام داده و سپس PDUها را از پیام استخراج کرده و آنها را به واحدهای کاربردی استفاده‌کننده از SNMP تحویل می‌دهد.

در یک مدیر خالص، موتور SNMP شامل یک حمل‌کننده (Dispatcher)، یک زیرسیستم پردازش پیام (Processing Subsystem) و یک زیرسیستم امنیت (Security Subsystem) است. حمل‌کننده بسادگی یک مدیر ترافیک است. برای PDUهای خارج‌شونده، حمل‌کننده PDUها را از کاربردها تحویل گرفته و کارهای زیر را انجام می‌دهد. برای هرPDU، حمل‌کننده نوع پردازش لازم برای پیام را تعیین کرده (یعنیSNMPv1، SNMPv2c و SNMPv3) و PDU را به مدول مناسب در زیرسیستم پردازش پیام عبور می‌دهد. به دنبال آن زیرسیستم پردازش پیام، یک پیام که شامل آن PDU و سرآیندهای مناسب هستند را باز می‌گرداند. حمل‌کننده آنگاه این پیام را برای انتقال به لایهٔ حمل‌ونقل می‌دهد.



شکل ۶-۸ مدیر سنّتی SNMP

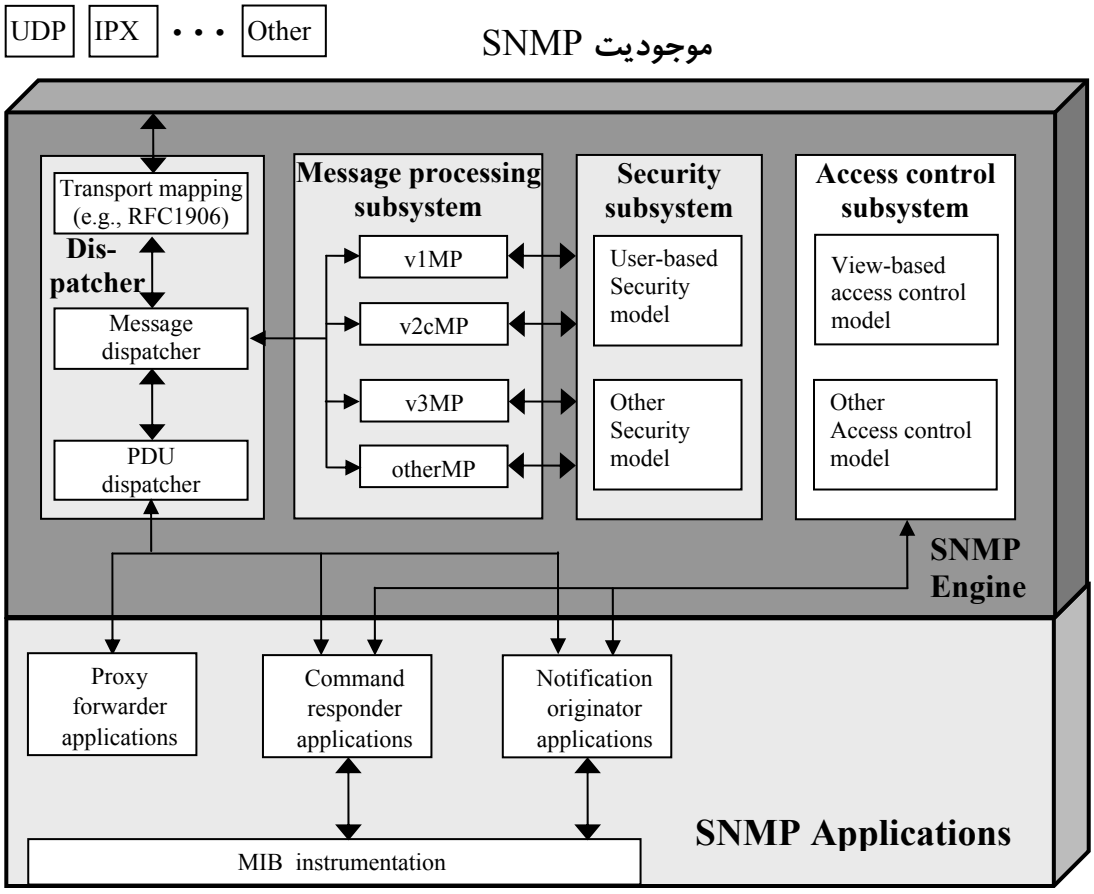
برای پیام‌های واردشونده، حمل‌کننده پیام‌ها را از لایهٔ حمل‌ونقل تحویل گرفته و عملیات زیر را انجام می‌دهد. حمل‌کننده، هر پیام را به مسیر مناسب که منتهی به مدول مناسب پردازش پیام است راهنمائی می‌کند. به دنبال آن زیرسیستم پردازش پیام، PDU موجود در پیام را بازپس می‌دهد. حمل‌کننده این PDU را به کاربرد مناسب عبور می‌دهد.

زیرسیستم پردازش پیام، PDUهای خارج شونده از حمل‌کننده را پذیرفته، آنها را با سرآیندهای مناسب تجهیز کرده و سپس به حمل‌کننده بازپس می‌دهد. زیرسیستم پردازش پیام همچنین پیام‌های ورودی را از حمل‌کننده قبول کرده، سرآیند هر پیام را پردازش نموده و PDU حمل‌شده در پیام را به حمل‌کننده برمی‌گرداند. یک پیاده‌سازی زیرسیستم پردازش پیام ممکن است تنها یک فرمت خاص پیام که مرتبط با یک نسخهٔ خاص SNMP (SNMPv1 ، SNMPv2c و یا SNMPv3) است را پشتیبانی کرده و یا ممکن است شامل تعدادی مدول باشد که هرکدام نسخهٔ خاصی از SNMP را پشتیبانی نمایند.

زیرسیستم امنیت، وظایف مربوط به اعتبارسنجی و رمزنگاری را انجام می‌دهد. هر پیام خارج‌شونده از زیرسیستم پردازش پیام، به زیرسیستم امنیت وارد می‌شود. بسته به سرویس مورد نیاز، زیرسیستم امنیت ممکن است PDU موجود در پیام و احتمالاً بخشی از میدان‌های سرآیند پیام را رمزنگاری کرده و ممکن است یک کُد اعتبارسنجی پیام تولید نموده و آن را در سرآیند پیام وارد کند. پیام پردازش‌شده آنگاه برای زیرسیستم پردازش پیام برگردانده می‌شود. به طریق مشابه، هر پیام واردشونده از زیرسیستم پردازش پیام به زیرسیستم امنیت عبور داده می‌شود. اگر لازم باشد، زیرسیستم امنیت کُد اعتبارسنجی را کنترل کرده، رمزگشائی را انجام داده و سپس پیام پردازش شده را به زیرسیستم پردازش پیام برمی‌گرداند. پیاده‌سازی زیرسیستم امنیت ممکن است یک یا چند مدل امنیتی خاص را پوشش دهد. تا کنون تنها مدل امنیتی تعریف شده User-Based Security Model (USM) برای SNMPv3 است که در RFC 2574 توصیف شده است.

شکل ۷-۸ که بر پایهٔ تصویر ارائه شده در RFC 2571 بنا شده است، بلوک دیاگرام یک **عامل سنّتی یا خالص SNMP** را نشان می‌دهد.

عامل سنّتی یا خالص ممکن است شامل سه نوع کاربرد باشد. کاربردهای پاسخ‌دهنده به فرامین، دست‌یابی به داده‌های مدیریت شده را فراهم می‌سازند. این کاربردها به درخواست‌های ورودی با گردآوری و/یا تنظیم موضوعات مدیریت شده و سپس صدور یک Response PDU پاسخ می‌دهند. یک کاربرد مولد اخطار، آغازگر پیام‌های آسنکرون می‌باشد. در مورد یک عامل خالص، PDUهای SNMPv2-Trap و SNMPv1-Trap برای این منظور مورد استفاده قرار می‌گیرند. یک کاربرد جلوبرندهٔ پروکسی، پیام‌ها را بین موجودیت‌ها، به جلو می‌راند.



شکل ۷-۸ عامل سنّتی SNMP

موتور SNMP برای یک عامل خالص، دارای تمام مؤلفه‌های موجود در موتور SNMP برای یک مدیر خالص باضافهٔ یک زیرسیستم کنترل دست‌یابی(Access Control Subsystem) است. این زیرسیستم وظیفهٔ شناسائی دست‌یابی‌های مجاز به MIB جهت خواندن و تنظیم موضوعات مدیریتی را داراست. این سرویس‌ها بر مبنای محتویات PDUها انجام می‌شوند. یک پیاده‌سازی زیرسیستم امنیتی، ممکن است از یک یا چند مدول کنترل دست‌یابی پشتیبانی نماید. تا کنون تنها مدل امنیتی تعریف شده View-Based Access Control Model (VACM) است که در RFC 2575 توصیف شده است.

توجه کنید که عملیات مرتبط با امنیت در دو زیرسیستم مجزا سازمان‌دهی شده‌اند: امنیت و کنترل دست‌یابی. این یک مثال عالی از طراحی پودمانی است، زیرا دو زیرسیستم وظایف کاملاً مشخصی را انجام داده و بنابراین منطقی خواهد بود که استانداردسازی این دو مقوله بطور مستقل از هم انجام شود. زیرسیستم امنیت، وظیفهٔ کنترل سرّی بودن و معتبر بودن را بعهده داشته و روی پیام‌های SNMP عمل می‌کند. زیرسیستم کنترل دست‌یابی، وظیفهٔ کنترل دست‌یابی مجاز به اطلاعات مدیریتی را داشته و روی PDUهای SNMP عمل می‌کند.

فرهنگ واژه‌ها

جدول ۲-۸ بطور خلاصه بعضی از واژه‌هائی که در RFC 2571 معرفی شده‌اند را تعریف می‌کند. با هر موجودیت SNMP یک snmpEngineID یکتا مرتبط است. برای اهداف کنترل دست‌یابی، فرض می‌شود که هر موجودیت SNMP، مدیریت تعـــدادی از مقـــوله‌های(context) اطلاعات مدیریت‌شده را بعهده دارد که هرکدام از آنها دارای یک contextName هستند که در آن موجودیت، یکتاست. برای تأکید بر اینکه در هر موجودیت یک مدیر واحد مقوله‌ها وجود دارد، هر موجودیت دارای یک contextEngineID یکتای مرتبط با آن است. چون یک ارتباط یک- به- یک بین موتور context و موتور SNMP در این موجودیت وجود دارد، contextEngineID از نظر اندازه برابر snmpEngineID است. کنترل دست‌یابی بتوسط مقوله‌های مشخصی که برای دست‌یابی به آنها تلاش شده است و هویت کاربر متقاضی، مدیریت می‌گردد. این کاربر که ممکن است یک فرد، یک کاربرد و یا گروهی از افراد یا کاربردها باشند، را رئیس(principal) خوانند.

سایر واژه‌های دارای اهمیت، مربوط به پردازش پیام‌ها می‌باشند. snmpMessageProcessingModel فرمت پیام و نسخهٔ SNMP پردازش پیام را مشخص می‌کند. snmpSecurityModel تعیین می‌کند که از کدام مدل امنیتی باید استفاده شود. snmpSecurityLevel تعیین می‌کند که کدام سرویس‌های امنیتی برای این کار مشخص درخواست شده‌اند. کاربر ممکن است فقط اعتبارسنجی، یا اعتبارسنجی بعلاوهٔ محرمانگی(رمزنگاری) و یا هیچکدام را درخواست کند.

کاربردهای SNMPv3

سرویس‌های بین مدول‌ها در یک موجودیت SNMP، در RFCها، برحسب فرامین ابتدائی(primitives) و پارامترها (parameters) تعریف شده‌اند. یک فرمان ابتدائی، عملی که باید انجام شود را مشخص کرده و از پارامترها برای عبور دادن دیتا و اطلاعات کنترلی استفاده می‌شود. این فرامین اولیه و پارامترها را می‌توان یک روش فرموله شده برای تعریف سرویس‌های SNMP دانست. فرم واقعی یک فرمان ابتدائی مستقل از پیاده‌سازی است و مثالی از آن احضار یک procedure است. در بحثی که به دنبال خواهد آمد، استفاده از شکل ۸-۸ که بر اساس تصویری در RFC 2571 بنا شده است ممکن است مفید واقع گردد تا ببینیم چگونه تمام این فرامین اولیه با هم جفت و جور می‌شوند. شکل ۸-۸الف دنباله‌ای از وقایعی را نشان می‌دهد که درآن یک کاربرد مولد فرمان یا مولد اخطار، درخواست ارسال یک PDU را می‌نماید و به دنبال آن یک پاسخ منطبق با این درخواست برای آن کاربرد پس فرستاده می‌شود. این پیشامدها در یک مدیر SNMP واقع

می‌شود. شکل ۸-۸ب پیشامدهای نظیر در یک عامل SNMP را نشان می‌دهد. در شکل نشان داده شده است که چگونه یک پیام ورودی منجر به تحویلPDU موجود در آن به یک کاربرد گشته و چگونه پاسخ کاربرد، منتج به یک پیام خروجی می‌گردد. توجه شود که بعضی پیکان‌ها در دیاگرام با نام یک primitive مشخص گردیده که نمایشگر یک درخواست است. پیکان‌های بدون نام، نمایشگر پاسخ به یک درخواست بوده و سایه‌ها نمایش انطباق بین درخواست و پاسخ مرتبط با آن است.

RFC 2573 بصورت کلی، رَویه‌هائی که هنگام تولید یک PDU برای ارسال و یا پردازش یک PDU ورودی، برای هر یک از کاربردها دنبال می‌شود را تعریف می‌کند. در همهٔ موارد، رَویه‌ها بر اساس تعامل با حمل‌کننده و برحسب Dispatcher Primitives تعریف می‌شوند.

جدول ۲-۸ فرهنگ واژه‌های SNMPv3

snmpEngineID
شناسهٔ یکتا و بدون ابهام یک موتور SNMP، و همچنین موجودیت SNMP که مرتبط با آن موتور است. این شناسه برحسب قرارداد بصورت Octet String تعریف می‌شود.

contextEngineID
بطور یکتا یک موجودیت SNMP که ممکن است نمونه‌ای از یک مقوله یا یک contextName خاص را شکل دهد تعریف می‌کند.

contextName
یک مقولهٔ بخصوص در یک موتور SNMP را معرفی می‌کند. این مقدار بعنوان یک پارامتر به Dispatcher و زیرسیستم کنترل دست‌یابی رد می‌شود.

scopedPDU
یک بلوک دیتا که شامل یک contextEngineID، یک contextName و یک SNMP PDU است. بصورت یک پارامتر به زیرسیستم امنیت عبور کرده و یا از آن گرفته می‌شود.

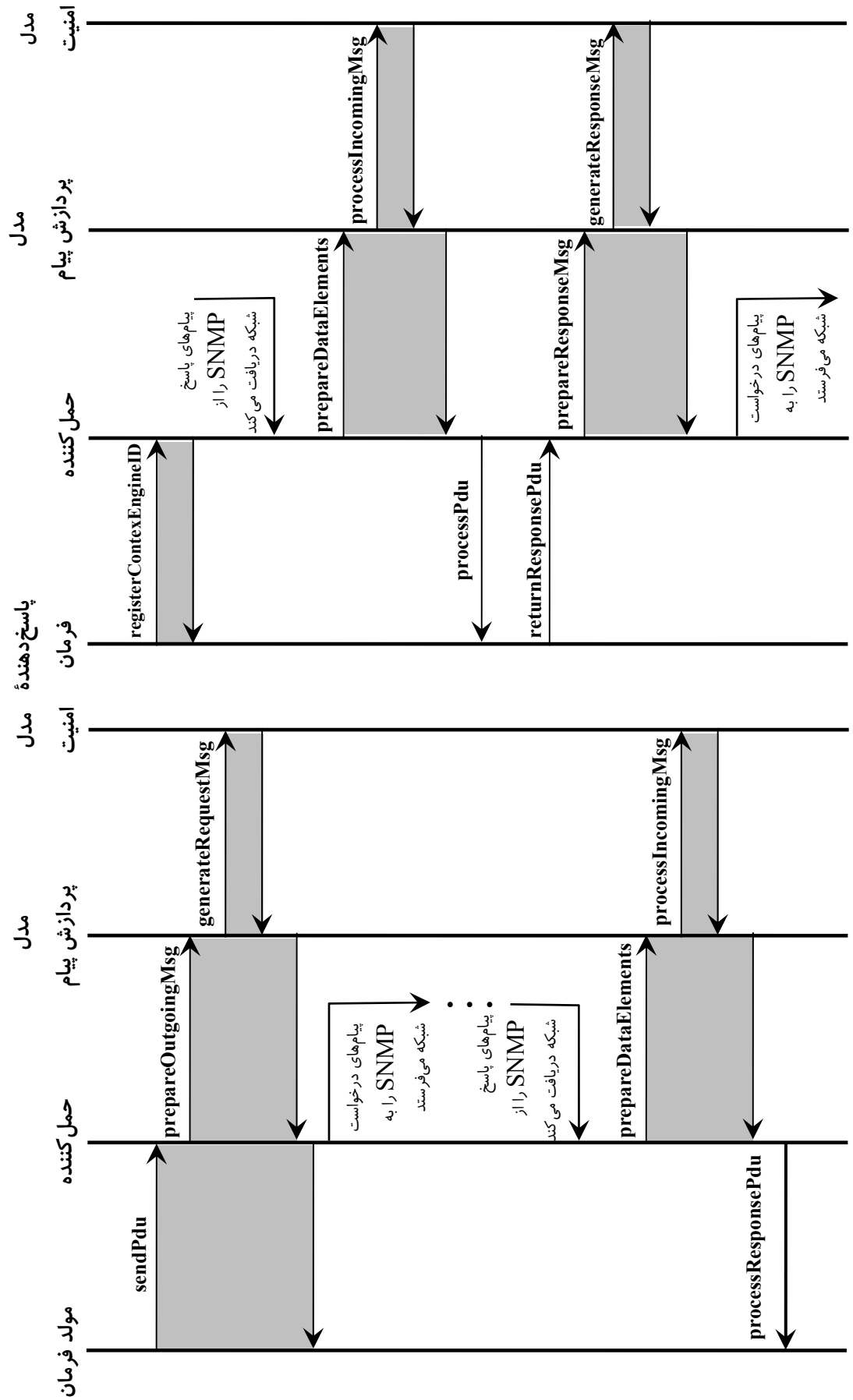
snmpMessageProcessingModel
شناسهٔ یکتای یک مدل پردازش پیام در زیرسیستم پردازش پیام است. مقادیر ممکن شامل SNMPv1، SNMPv2c و SNMPv3 است. بر حسب قرارداد بصورت Integer تعریف می‌شود.

snmpSecurityModel
شناسهٔ یکتای یک مدل امنیتی در زیرسیستم امنیت است. مقادیر ممکن شامل SNMPv1، SNMPv2c و USM است. بر حسب قرارداد بصورت Integer تعریف می‌شود.

snmpSecurityLevel
یک سطح امنیتی که در آن سطح، پیام‌های SNMP می‌توانند ارسال شده و یا عملیات پردازش صورت پذیرد و بدین صورت بیان می‌گردد که آیا اعتبارسنجی و/ یا محرمانگی بکار گرفته می‌شود یا نه. مقادیر ممکن noAuthnoPriv، authNoPriv و authPriv است و برحسب قرارداد بصورت Integer تعریف می‌شود.

principal
موجودیتی که از جانب او سرویس‌ها فراهم شده و یا پردازش‌ها انجام می‌شوند. یک principal می‌تواند فردی در یک نقش مشخص، مجموعه‌ای از افراد که هر کدام دارای نقش معین هستند، یک کاربرد و یا مجموعه‌ای از کاربردها و یا ترکیبی از همه این انواع باشد.

securityName
یک رشته قابل خواندن بتوسط انسان که نمایش‌دهندهٔ یک principal است. بصورت یک پارامتر در تمام فرامین اولیه SNMP ردوبدل می‌شود (Dispatcher ، Message Processing ، Security ، Access Control).



(الف) مولد فرمان یا مولد اخطار (ب) پاسخ‌دهندهٔ فرمان

شکل ۸-۸ جریان عملیات SNMP

یک **کاربرد مولد فرمان (command generator application)** از فرامین ابتدائی حمل‌کنندهٔ sendPdu و processResponsePdu استفاده می‌کند. sendPdu، حمل‌کننده را با اطلاعاتی در مورد مقصد مورد نظر، پارامترهای امنیتی و PDU واقعی که باید ارسال شود تجهیز می‌کند. حمل‌کننده آنگاه مدل پردازش پیام (Message Processing Model) را بکار گرفته که آنهم بنوبهٔ خود مدل امنیتی (Security Model) را بخدمت طلبیده تا پیام را آماده نمایند. حمل‌کننده، پیام آماده شده را برای انتقال به لایهٔ حمل‌ونقل (مثلاً UDP) می‌فرستد. اگر آماده‌سازی پیام دچار مشکل شود، اندازهٔ برگشتی sendPdu primitive که بتوسط حمل‌کننده تنظیم می‌شود نمایشگر یک خطا خواهد بود. اگر آماده‌سازی پیام موفقیت‌آمیز باشد، حمل‌کننده یک شناسه sendPduHandle به این PDU اختصاص داده و اندازهٔ آن را به مولد فرمان برمی‌گرداند. مولد فرمان این sendPduHandle را ذخیره نموده تا بتواند PDU پاسخ متعاقب را، با این درخواست اولیه تطبیق دهد.

حمل‌کننده هر PDU، پاسخ ورودی را با استفاده از فرمان ابتدائی processResponsePdu به کاربرد مولد فرمان تحویل می‌دهد.

یک **کاربرد پاسخ‌دهندهٔ فرمان (command responder application)** از چهار فرمان ابتدائی حمل‌کننده (registerContextEngineID، unregisterContextEngineID، processPdu و returnResponsePdu) و یک فرمان ابتدائی Access Control Subsystem استفاده می‌کند.

registerContextEngineID primitive یک کاربرد پاسخ‌دهندهٔ فرمان را قادر می‌سازد تا خود را با یک موتور SNMP برای پردازش انواع PDU معین برای یک موتور context مرتبط نماید. وقتی یک پاسخ‌دهندهٔ فرمان، ثبت‌نام گردید، تمام پیام‌های آسنکرون دریافت‌شده که شامل ترکیب‌های ثبت‌نام‌شدهٔ contextEngineID و pduType مورد پشتیبانی هستند به پاسخ‌دهندهٔ فرمانی که برای پشتیبانی از این ترکیب ثبت‌نام شده است ارجاع می‌شوند. یک پاسخ‌دهندهٔ فرمان می‌تواند خود را از یک موتور SNMP که از unregisterContextEnginID primitive استفاده می‌کند جدا سازد.

حمل‌کننده هر PDU درخواست ورودی را با استفاده از processPdu primitive به کاربرد پاسخ‌دهندهٔ فرمان صحیح تحویل می‌دهد. به دنبال آن پاسخ‌دهندهٔ فرمان قدم‌های زیر را برمی‌دارد:

- پاسخ‌دهندهٔ فرمان، محتوای PDU درخواست را وارسی می‌کند. نوع عملیات بایستی با یکی از انواعی که قبلاً بتوسط این کاربرد ثبت نام شده است، تطبیق داشته باشد.
- پاسخ‌دهندهٔ فرمان تعیین می‌کند که آیا دست‌یابی برای عملیات مدیریتی تقاضا شده در این PDU مجاز است. برای این منظور AccessAllowed primitive احضار می‌گردد.
- پارامتر securityModel نشان می‌دهد که زیرسیستم کنترل دست‌یابی برای پاسخ به این درخواست از کدام مدل امنیتی باید استفاده کند. زیرسیستم کنترل دست‌یابی تعیین می‌کند که آیا این رئیس متقاضی (securityName) در این سطح امنیتی(securityLevel) حق درخواست این عمل مدیریتی (viewType) در این موضوع مدیریتی (variableName) در این مقوله (contextName) را داراست.
- اگر اجازهٔ دست‌یابی داده شد، پاسخ‌دهندهٔ فرمان عمل مدیریتی درخواست شده را انجام داده و یک PDU پاسخ را تهیه می‌کند. اگر دست‌یابی مجاز شناخته نشود، پاسخ‌دهنده فرمان، PDU پاسخ مناسب برای شکست عملیات را تهیه می‌کند.
- پاسخ‌دهندهٔ فرمان، حمل‌کننده را با یک returnResponsePdu فراخوانده تا PDU پاسخ را ارسال کند.

یک **کاربرد مولد اخطار (notification generator application)** همان رَویه‌های عمومی یک کاربرد مولد فرمان را دنبال می‌کند. اگر قرار است که یک Inform Request PDU ارسال شود، هم از sendPdu primitive و هم از processResponse primitive، بهمان روش کاربردهای مولد فرمان، استفاده می‌شود. اگر قرار است یک trap PDU ارسال شود تنها از sendPdu primitive استفاده می‌شود.

یک **کاربرد گیرندهٔ اخطار (notification receiver application)** از یک زیرمجموعه از رَویه‌های عمومی همانند کاربرد پاسخ‌دهندهٔ فرمان استفاده می‌کند. گیرندهٔ اخطار ابتدا بایستی ثبت‌نام شده تا PDU های Inform و/یا trap را دریافت کند. هر دو نوع PDU بتوسط یک processPdu primitive دریافت می‌شوند. برای پاسخگوئی به یک Inform PDU از یک returnResponsePdu primitive استفاده می‌شود.

یک **کاربرد جلوبرندهٔ پروکسی (proxy forwarder application)** از Dispatch primitives برای به جلو راندن پیام‌های SNMP استفاده می‌کند. جلوبرندهٔ پروکسی چهار نوع پیام اصلی را پشتیبانی می‌کند:

- پیام‌هائی که شامل انواع PDU از یک کاربرد مولد پیام هستند. جلوبرندهٔ پروکسی موتور SNMP مقصد، و یا نزدیک‌ترین موتور SNMP پائین دست را، تعیین کرده و PDU درخواست مناسب را ارسال می‌دارد.
- پیام‌هائی که شامل انواع PDU از یک کاربرد مولد اخطار هستند. جلوبرندهٔ پروکسی تعیین می‌کند که کدام موتور SNMP بایستی اخطار را دریافت کرده و PDU یا PDUهای مناسب را ارسال می‌کند.
- پیام‌هائی که شامل یک نوع Response PDU هستند. جلوبرندهٔ پروکسی تعیین می‌کند که کدام درخواست یا اخطار ارسال شدهٔ قبل با این پاسخ تطبیق داشته و PDU مناسب را ارسال می‌کند.
- پیام‌هائی که شامل یک گزارش‌اند. Report PDUها ارتباطات موتور- به- موتور SNMPv3 هستند. جلوبرندهٔ پروکسی تعیین می‌کند که کدام درخواست و یا اخطار بجلو رانده شدهٔ قبلی با این گزارش تطبیق داشته و گزارش را به آغازگر درخواست یا اخطار بازپس می‌فرستد.

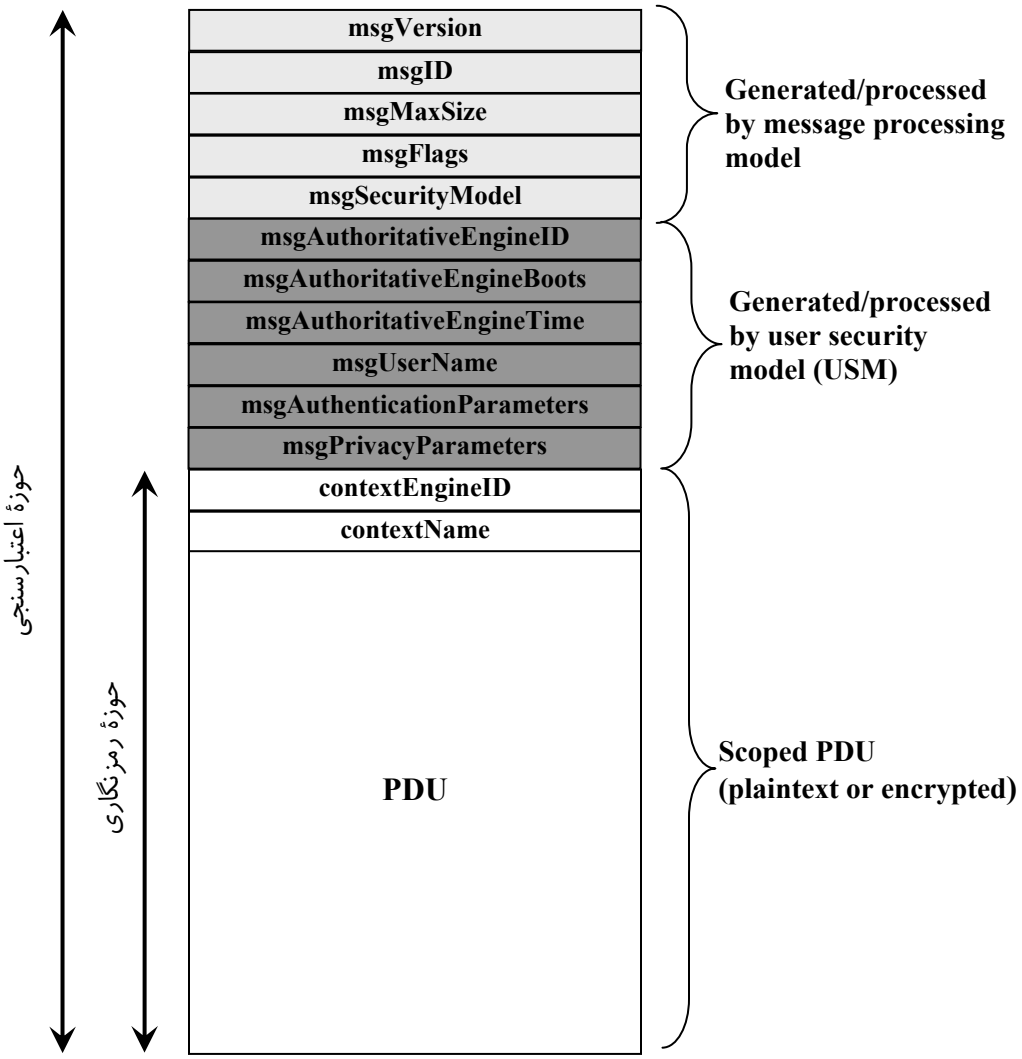
پردازش پیام و مدل امنیتی کاربر

پردازش پیام شامل یک مدل پردازش پیام همه- منظوره و یک مدل امنیتی خاص است. این ارتباط در شکل ۸-۸ نشان داده شده است.

مدل پردازش پیام

RFC 2572 یک مدل پردازش پیام همه- منظوره را تعریف کرده است. این مدل مسئول پذیرش PDUها از حمل‌کننده بوده، آنها را به فرم پیام کپسولی نموده و با استفاده از USM پارامترهای مرتبط با امنیت را در سرآیند پیام‌ها وارد می‌کند. مدل پردازش پیام همچنین پیام‌های ورودی را پذیرفته، USM را برای پردازش پارامترهای مرتبط با امنیت در سرآیند پیام بکار گرفته و PDUهای کپسولی شده را به حمل‌کننده تحویل می‌دهد.

شکل ۹-۸ ساختار پیام را نشان می‌دهد. پنج میدان اولیه بتوسط مدل پردازش‌گر پیام برای پیام‌های خارج‌شونده تولید، و بتوسط مدل پردازش‌گر پیام برای پیام‌های واردشونده پردازش می‌گردند. شش میدان بعدی نشان‌دهندهٔ پارامترهای امنیتی استفاده شده بتوسط USM هستند. بالاخره، PDU به همراه contextEngineID و contextName یک *scoped* PDU را تشکیل می‌دهند که برای پردازش PDU بکار می‌رود.



شکل ۹-۸ فرمت پیام SNMPv3 با USM

پنج میدان اول بشرح زیراند:

- **msgVersion**: برابر 3 (snmpv3) است.
- **msgID**: یک شناسهٔ یکتا که بین دو موجودیت SNMP برای هم‌آهنگ کردن پیام‌های درخواست و پاسخ بکار رفته و پردازش‌گر پیام نیز از آن برای هم‌آهنگ کردن پردازش پیام بتوسط مدل‌های مختلف زیرسیستم‌ها در معماری استفاده می‌کند. محدودهٔ این ID ، صفر تا $۲^{۳۱}-۱$ است.
- **msgMaxSize**: بیانگر اندازهٔ ماکزیمم یک پیام بر حسب اُکتت است که از سوی فرستنده پشتیبانی می‌شود و محدودهٔ آن ۴۸۴ تا $۲^{۳۱}-۱$ است. این اندازهٔ ماکزیمم سِگمنتی است که فرستنده می‌تواند از یک موتور SNMP دیگر بپذیرد (چه یک پاسخ و چه انواع دیگر پیام‌ها).

- **msgFlags**: یک اُکتت که کم اهمیت‌ترین سه بیت آن شامل سه پرچم است: reportableFlag، privFlag و authFlag. اگر reportableFlag = 1 باشدآنگاه یک Report PDU بایستی، در تحت شرایطی که می‌تواند باعث تولید یک Report PDU گردد، به فرستنده بازگردانده شود. وقتی این پرچم صفر است، امکان ارسال یک Report PDU وجود ندارد. پرچم reportableFlag در تمام پیام‌هائی که شامل یک درخواست (GET و SET) و یا یک Inform است برابر 1 و برای پیام‌هائی که شامل یک Response، یک Trap و یا یک Report PDU است برابر 0 قرار داده می‌شود. ReportableFlag به این امر کمک می‌کند که چه زمانی یک Report ارسال شود. از این امر تنها در مواردی استفاده می‌شود که در آن، بخش PDU پیام نتواند کُدگشائی شود (مثلاً وقتی که بعلت کلید ناصحیح، عمل رمزگشائی با شکست مواجه شود). privFlag و authFlag بتوسط فرستنده افراشته شده و برای نشان دادن سطح امنیتی تخصیص داده شده به پیام بکار می‌رود. privFlag = 1 به مفهوم اعمال رمزنگاری و privFlag = 0 به مفهوم اعمال اعتبارسنجی است. تمام ترکیب‌های ممکن بجز authFlag = 0 AND privFlag = 1 قابل اعمال است، یعنی فقط رمزنگاری بدون اعتبارسنجی ممکن نیست.
- **msgSecurityModel**: یک شناسه در محدودهٔ صفر تا $۲^{۳۱}-۱$ است که نشان می‌دهد کدام مدل امنیتی از طرف فرستنده برای آماده‌سازی پیام بکار گرفته شده است و بنابراین تعیین می‌کند که گیرنده بایستی از کدام مدل امنیتی برای پردازش پیام استفاده کند. اندازه‌های رزرو شده شامل 1 برای SNMPv1، 2 برای SNMPv2c (SNMPv2 با تسهیلات جامعه‌ای SNMPv1) و 3 برای SNMPv3 است.

مدل امنیتی کاربر

RFC 2574 مدل امنیتی کاربر User Security Model (USM) را تعریف می‌کند. USM سرویس‌های اعتبارسنجی و محرمانگی را برای SNMP فراهم می‌آورد. USM علی‌الخصوص برای حفظ امنیت در برابر تهدیدهای زیر طراحی شده است:

- **دستکاری اطلاعات**: یک موجودیت ممکن است یک پیام در حال ترانزیت، که بتوسط یک موجودیت معتبر تولید شده است، را طوری تغییر دهد که باعث انجام عملیات مدیریتی غیرمجاز گردد. نتیجهٔ این تهدید این است که یک موجودیت غیرمجاز بتواند هر پارامتر مدیریتی که از جمله شامل پیکربندی، عملیات و حسابرسی است را عوض کند.
- **نقاب‌گذاری**: عملیات مدیریتی غیرمجاز برای یک موجودیت ممکن است بتوسط آن موجودیت،که نقاب یک موجودیت مجاز را به چهره گذاشته است، انجام شود.
- **دستکاری جریان پیام**: SNMP برای کاربرد روی یک پروتکل حمل‌ونقل بدون اتصال طراحی شده است. این تهدید وجود دارد که پیام‌های SNMP جابجاشده، به تأخیرافتاده و یا بازخوانی شده و به عملیات غیرمجاز مدیریتی منجر گردند. برای مثال پیامی برای reboot کردن یک دستگاه ممکن است کپی شده و در آینده مجدداً ارسال گردد.
- **افشا**: یک موجودیت ممکن است مبادلات بین یک مدیر و یک عامل را مشاهده کرده و از این طریق اندازه‌های موضوعات مدیریت شده و همچنین پیشامدهای قابل اخطار را کشف کند. برای مثال، مشاهدهٔ یک مجموعه از فرامین که کلمات عبور را تغییر می‌دهد، یک حمله‌کننده را قادر خواهد ساخت تا کلمات عبور جدید را بدست آورد.

USM برای مقابله با دو تهدید زیر طراحی نشده است:

- **انکار سرویس**: یک حمله‌کننده ممکن است مانع مبادلات بین یک مدیر و یک عامل شود.
- **تحلیل ترافیک**: یک حمله‌کننده ممکن است الگوی عمومی ترافیک بین مدیران و عوامل را مشاهده نماید.

عدم وجود یک مکانیسم مقابله با تهدید انکار سرویس را ممکن است به دو دلیل زیر منطقی دانست: اول اینکه حملات انکار سرویس در بسیاری موارد از خرابی‌های شبکه، که خودکاربردهای مدیریت شبکه باید آن را مدیریت نمایند، قابل تفکیک نیستند. در ثانی یک حملهٔ انکار سرویس احتمالاً باعث از هم گسیختن تمام مبادلات شده و در نتیجه در مقولهٔ تسهیلات امنیتی کل شبکه، نه تنها آنچه در پروتکل مدیریت شبکه وجود دارد، قرار می‌گیرد. در مورد تحلیل ترافیک، بسیاری از پترن‌های ترافیکی مدیریت شبکه قابل پیش‌بینی بوده (مثلاً موجودیت‌های مختلف ممکن است از طریق فرامین SNMP، از سوی یک یا چند ایستگاه مدیریتی، با ضرب‌آهنگ منظم مدیریت شوند) و بنابراین محافظت از آنها در برابر شنود بیگانه امتیاز قابل توجه‌ای دربر ندارد.

توابع رمزنگاری

دو تابع رمزی برای USM تعریف شده است: اعتبارسنجی و رمزنگاری. برای حمایت از این دو تابع، یک موتور SNMP نیاز به دو مقدار دارد: یک کلید خصوصی‌سازی (privKey) و یک کلید اعتبارسنجی (authKey). اندازه‌های متفاوت این دو کلید برای کاربران زیر نگهداری می‌شوند:

- **کاربران محلی**: هر رئیسی (principal) در این موتور SNMP است که برای او عملیات مدیریت مجاز است.
- **کاربران دور**: هر رئیسی در یک موتور SNMP دور است که برای او ارتباطات مورد نیاز است.

این مقادیر از جمله صفات منتسب به کاربر برای هر کاربر مرتبط است. اندازه‌های privKey و authKey از طریق SNMP قابل دسترس نیستند.

USM استفاده از یک و یا هر دو پروتکل اعتبارسنجی HMAC-MD5-96 وHMAC-SHA-96 را مجاز می‌شمارد. HMAC که در فصل ۳ تشریح گردید، از یک تابع درهم‌ساز امن بعلاوهٔ یک کلید سرّی، برای تولید یک کُد اعتبارسنجی پیام، استفاده می‌کند. HMAC-MD5-96 از MD5 بعنوان تابع hash درونی استفاده می‌کند. یک authKey با طول ۱۶ اُکتت (۱۲۸ بیت) برای ورودی الگوریتم HMAC بکار می‌رود. الگوریتم یک خروجی ۱۲۸- بیتی را ایجاد کرده که تنها ۱۲ اُکتت (۹۶ بیت) آن مورد استفاده قرار می‌گیرد. برای HMAC-SHA-96 تابع hash درونی SHA-1 است. در اینجا authKey دارای طول ۲۰ اُکتت (۱۶۰ بیت) است. الگوریتم یک خروجی ۲۰- اُکتتی ایجاد می‌کند که در اینجا نیز فقط از ۱۲ اُکتت آن استفاده می‌شود.

USM از مُود زنجیره‌ای رمز قالبی(CBC) استاندارد رمزنگاری دیتا (DES) استفاده می‌کند. یک privKey با ۱۶ اُکتت طول برای ورودی پروتکل رمزنگاری بکار گرفته می‌شود. اولین ۸ اُکتت (۶۴ بیت) این privKey بعنوان کلید DES مورد استفاده قرار می‌گیرد. چون DES تنها به یک کلید ۵۶- بیتی نیاز دارد، کم اهمیت‌ترین بیت‌های هر اُکتت نادیده گرفته می‌شود. برای مُود CBC یک بردار اولیهٔ(IV) ۶۴- بیتی مورد نیاز است. آخرین ۸ اُکتت privKey شامل اندازه‌ای است که برای تولید IV از آن استفاده می‌گردد.

موتورهای مسئول و غیرمسئول

در هر انتقال پیام، یکی از دو واحد فرستنده یا گیرنده، بر طبق قواعد زیر بعنوان موتور SNMP مسئول انتخاب می‌شود:

- وقتی یک پیام SNMP شامل محموله‌ای است که انتظار یک پاسخ را دارد (مثل Get,GetNext,GetBulk,Set یا Inform PDU)، آنگاه گیرندهٔ چنین پیامی مسئول تلقی می‌گردد.
- وقتی یک پیام SNMP شامل محموله‌ای است که انتظار یک پاسخ را ندارد (مثل یک snmpv2-Trap، Response یا Report PDU)، آنگاه فرستندهٔ چنین پیامی، مسئول تلقی می‌گردد.

بنابراین، برای پیام‌هائی که از سوی یک مولد فرمان ارسال می‌شود و برای پیام‌های Inform که از جانب یک مولد اخطار ارسال می‌گردد، گیرنده مسئول است. برای پیام‌های یک پاسخ‌دهندهٔ فرمان و پیام‌های trap که از سوی یک مولد اخطار ارسال می‌شود، فرستنده مسئول است. این نوع تخصیص مسئولیت دو هدف را دنبال می‌کند:

- بهنگام بودن یک پیام با ساعت موتور مسئول ارزیابی می‌شود. وقتی یک موتور مسئول، یک پیام را ارسال می‌کند (Trap,Response,Report)، اندازهٔ جاری ساعت خود را در آن قرار می‌دهد تا گیرندهٔ غیرمسئول بتواند خود را با آن ساعت هماهنگ سازد. وقتی یک موتور غیرمسئول یک پیام را ارسال می‌کند (Get,GetNext,GetBulk,Set,Inform)، اندازهٔ تخمینی ساعت جاری مقصد را در پیام قرار می‌دهد تا مقصد بتواند بهنگام بودن پیام را ارزیابی نماید.
- عمل محلی‌کردن کلیدها که بعداً تشریح خواهد شد، یک رئیس منفرد را قادر خواهد ساخت تا کلیدهای ذخیره شده در موتورهای متعدد را صاحب شود. این کلیدها در موتور مسئول بنحوی محلی خواهند شد که رئیس منفرد، تنها مسئول یک کلید بوده و از ریسک امنیتی ذخیره‌سازی کپی‌های متعدد کلید در یک شبکهٔ توزیع‌شده جلوگیری شود.

منطقی است که گیرندهٔ PDUهای Command Generator و Inform را، بعنوان موتور مسئول، و در نتیجه کنترل‌کنندهٔ بهنگام بودن پیام دانست. اگر یک response یا trap بتأخیر افتاده و یا بازخوانی شود، خطرآن خیلی جدی نیست، در صورتی که Command Generator PDU، و تا حدودی Inform PDU، منجر به عملیات مدیریتی همچون خواندن و یا تغییر موضوعات MIB می‌گردند. بنابراین تضمین این امر که چنین PDUهائی به تأخیر نیفتاده و یا بازخوانی نشده است مهم بوده زیرا می‌توانند اثرات نامطلوبی داشته باشند.

پارامترهای پیام USM

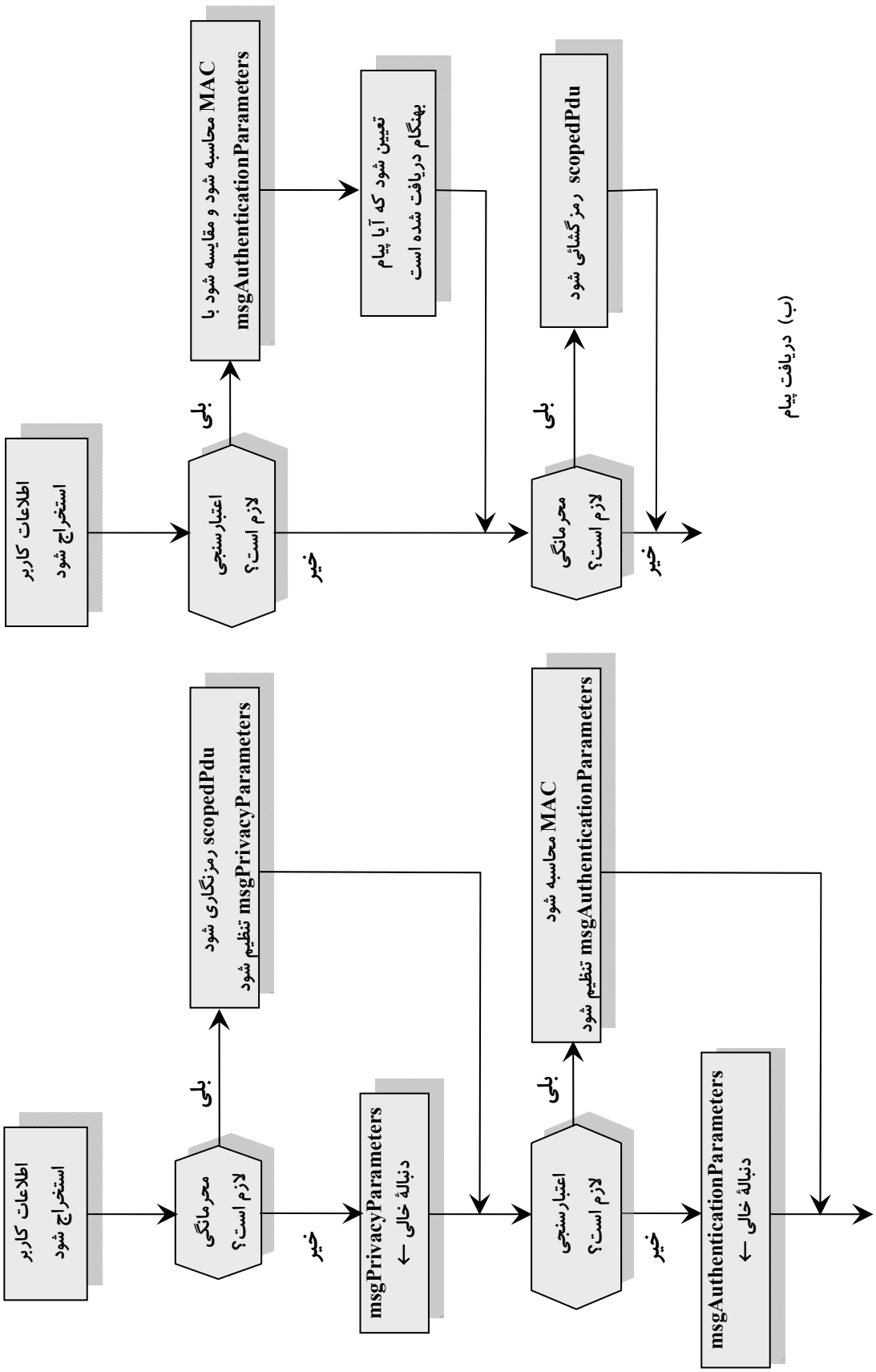
وقتی یک پیام خارج‌شونده، از سوی پردازش‌گر پیام به USM داده می‌شود، USM پارامترهای مرتبط با امنیت را در سرآیند پیام وارد می‌کند. وقتی یک پیام واردشونده از سوی پردازش‌گر پیام به USM داده می‌شود، USM پارامترهای امنیتی موجود در سرآیند پیام را بررسی و پردازش می‌نماید. پارامترهای مرتبط با امنیت به قرار زیراند:

- **msgAuthoritativeEngineID**: شناسهٔ snmpEngineID موتور SNMP مسئول در این مبادله است. بنابراین این مقدار در مورد یک Trap، یک Response و یک Report به مبدأ، و در مورد Get، GetNext، GetBulk، Set و Inform به مقصد اشاره می‌نماید.
- **msgAuthoritativeEngineBoots**: اندازهٔ snmpEngineBoots موتور SNMP مسئول در مبادلهٔ این پیام را نشان می‌دهد. موضوع snmpEngineBoots یک عدد صحیح در بازهٔ صفر تا $۲^{۳۱}-۱$ است که تعداد دفعاتی که موتور SNMP از پیکربندی اولیهٔ خود مجدداً آغازیدن گرفته است را نشان می‌دهد.
- **msgAutoritativeEngineTime**: اندازهٔ snmpEngineTime موتور SNMP مسئول در مبادلهٔ این پیام را نشان می‌دهد. موضوع snmpEngineTime یک عدد صحیح در بازهٔ صفر تا $۲^{۳۱}-۱$ است که تعداد ثانیه‌هائی را نشان می‌دهد که از آخرین باری که موتور SNMP مسئول، موضوع snmpEngineBoots را یک واحد اضافه کرده است، گذشته است. هر موتور SNMP مسئول، مسئولیت بالا بردن اندازهٔ snmpEngineTime خود به میزان یک واحد در هر ثانیه را داراست. یک موتور غیرمسئول، مسئولیت اضافه کردن snmpEngineTime خود، به اندازهٔ یک واحد برای هربار ارتباط با یک موتور مسئول، را بعهده دارد.
- **msgUserName**: کاربری (رئیس) که پیام از سوی او مبادله می‌گردد.
- **msgAuthenticationParameters**: اگر اعتبارسنجی برای این مبادله بکار نرود، خالی است. در غیر این صورت یک پارامتر اعتبارسنجی است. برای تعاریف فعلی USM، پارامتر اعتبارسنجی یک کُد اعتبارسنجی پیام HMAC است.
- **msgPrivacyParameters**: اگر محرمانگی برای این پیام بکار نرود، خالی است. در غیر این صورت یک پارامتر محرمانگی است. برای تعاریف فعلی USM، پارامتر محرمانگی مقداری است که بردار اولیه(IV) در الگوریتم CBC DES را نشان می‌دهد.

شکل ۱۰-۸ عملیات USM را خلاصه کرده است. برای انتقال پیام، اگر لازم باشد، ابتدا رمزنگاری انجام می‌شود. Scoped PDU رمزنگاری شده و در محمولهٔ پیام قرار می‌گیرد و اندازهٔ msgPrivacyParameters با مقداری پر می‌شود که برای تولید IV مورد نیاز است. آنگاه، اگر لازم باشد، اعتبارسنجی انجام می‌شود. تمام پیام که شامل scoped PDU است در ورودی HMAC قرار گرفته و کُد اعتبارسنجی بدست آمده در msgAuthenticationParameters قرار داده می‌شود. برای پیام‌های ورودی، اگر لازم باشد اعتبارسنجی انجام می‌شود. USM ابتدا MAC ورودی را با MAC محاسبه شدهٔ خود مقایسه کرده و اگر این دو با هم برابر باشند، پیام معتبر فرض می‌شود (از یک منبع مجاز صادر شده و در هنگام انتقال تغییر نکرده است). سپس USM کنترل می‌کند که آیا پیام، برابر آنچه بعداً تشریح خواهد شد، در درون یک بازهٔ زمانی مجاز دریافت شده باشد. اگر پیام بهنگام نباشد، غیر معتبر تلقی شده و معدوم خواهد شد. بالاخره اگر scoped PDU رمزنگاری شده باشد، USM رمزگشائی انجام داده و متن ساده را استخراج می‌کند.

مکانیسم USM برای کنترل بهنگام بودن

USM شامل مجموعه‌ای از مکانیسم‌های کنترل بهنگام بودن برای مقابله با تأخیر و یا بازخوانی پیام است. هر موتور SNMP که بتواند با ظرفیت یک موتور مسئول کارکند، بایستی دربرگیرندهٔ دو موضوع snmpEngineBoots و snmpEngineTime باشد که موضوع اخیر به زمان محلی در موتور اشاره دارد. وقتی یک موتور SNMP در ابتدا نصب می‌گردد اندازهٔ این دو موضوع برابر 0 تنظیم می‌گردد. پس از آن، snmpEngineTime در هر ثانیه یک واحد زیاد می‌شود.



شکل ۱۰-۸ پردازش پیام USM

اگر snmpEngineTime به مقدار ماکزیمم خود $۲^{۳۱}-۱$ برسد، snmpEngineBoots یک واحد اضافه می‌شود(مثل اینکه سیستم reboot شده باشد) و snmpEngineTime به صفر برگشته و مجدداً زیاد شدن را ادامه می‌دهد. با استفاده از یک مکانیسم سنکرونیزاسیون، یک موتور غیرمسئول می‌تواند تخمینی از اندازه‌های زمانی هر موتور مسئولی که با او ارتباط دارد را نگهداری کند. این اندازه‌های تخمینی در هر پیام خارج‌شونده قرار داده شده و به موتور مسئول دریافت‌کنندهٔ پیام اجازه می‌دهد تا تعیین کند که آیا پیام واردشونده بهنگام است یا خیر.

مکانیسم سنکرونیزاسیون بصورت زیر کار می‌کند. یک موتور غیرمسئول برای هر موتور SNMP مسئول که با او سروکار دارد، کپی سه مقدار را نگهداری می‌کند:

- **snmpEngineBoots**: آخرین اندازهٔ snmpEngineBoots برای هر موتور مسئول.
- **snmpEngineTime**: تخمین موتور غیرمسئول از snmpEngineTime موتور مسئول دور. این اندازه با موتور مسئول دور بصورتی که بعداً تشریح خواهد شد هم‌آهنگ می‌گردد. بین هم‌آهنگ‌سازی‌ها، این مقدار به میزان یک واحد در ثانیه افزایش یافته تا یک هم‌آهنگی نسبی با موتور مسئول دور بوجود آید.
- **latestReceivedEngineTime**: بزرگترین اندازهٔ msgAuthoritativeEngineTime که بتوسط این موتور از موتور مسئول دور دریافت گردیده است. این مقدار هربار که اندازهٔ بزرگتری از msgAuthoritativeEngineTime دریافت می‌گردد، بروزرسانی می‌شود. هدف این متغیر، حفاظت در برابر حملهٔ بازخوانی پیامی است که تخمین موتور SNMP غیرمسئول از snmpEngineTime را از جلورفتن بازدارد.

یک مجموعه از این سه متغیر، برای هر موتور مسئول دور که با این موتور ارتباط دارد، نگهداری می‌شود. از نظر منطقی، اندازه‌ها در نوعی حافظهٔ موقت نگهداری می‌شوندکه فهرستی از snmpEngineID هر موتور مسئول و اندازه‌های اخیر در آن ضبط شده است.

برای اینکه موتورهای غیرمسئول بتوانند هم‌آهنگی زمانی را حفظ کنند، هر موتور مسئول اندازهٔ boot و time را در کنار اندازهٔ snmpEngineID در هر پیام خروجی Response، Report و Trap خود در میدان‌های msgAuthoritativeEngineBoots، msgAuthoritativeEngineTime و msgAuthoritativeEngineID قرار می‌دهد. اگر پیام معتبر بوده و در پنجرهٔ زمانی معقول دریافت شود، آنگاه موتور غیرمسئول گیرنده، متغیرهای محلی خود (snmpEngineBoots، snmpEngineTime و latestReceivedEngineTime) برای آن موتور دور را بر طبق ضوابط زیر بروز می‌رساند:

۱- یک بروزرسانی وقتی انجام می‌شود که حداقل یکی از دو شرط زیر صحیح باشد:

- (msgAuthoritativeEngineBoots > snmpEngineBoots) **یا**
- [(msgAuthoritativeEngineBoots = snmpEngineBoots) **و**

[(msgAuthoritativeEngineTime > latestReceivedEngineTime)

اولین شرط می‌گویدکه بروزرسانی در صورتی باید واقع شود که اندازهٔ boot موتور مسئول از آخرین بروزرسانی افزایش یافته باشد. شرط دوم می‌گویدکه اگر اندازهٔ boot افزایش نیافته باشد، بروزرسانی فقط در صورتی باید انجام شود که زمان موتور ورودی بیشتر از زمان قبلی واردشدهٔ همین موتور باشد. زمان موتور ورودی در صورتی کمتر از آخرین زمان دریافت‌شدهٔ همین موتور خواهد بود که دو پیام ورودی خارج از نظم دریافت شده باشند. این امر هم بصورت عادی ممکن است واقع شود و هم ممکن است به دلیل یک حملهٔ بازخوانی صورت پذیرد و درهرحال موتور گیرنده بروزرسانی نخواهد شد.

۲- اگر بروزرسانی مجاز تشخیص داده شود، آنگاه تغییرات زیر انجام خواهد شد:

- اندازهٔ snmpEngineBoots به اندازهٔ msgAuthoritativeEngineBoots تغییر می‌یابد.
- اندازهٔ snmpEngineTime به اندازهٔ msgAuthoritativeEngineTime تغییر می‌یابد.
- اندازهٔ latestReceivedEngineTime به اندازهٔ msgAuthoritativeEngineTime تغییر می‌یابد.

اگر منطق را بچرخانیم می‌بینیم که اگر msgAuthoritativeEngineBoots < snmpEngineBoots باشد، آنگاه هیچ بروزرسانی انجام نخواهد شد. چنین پیامی نامعتبر تلقی شده و بایستی نادیده گرفته شود. اگر snmpEngineBoots = msgAuthoritativeEngineBoots بوده ولی msgAuthoritativeEngineTime < latestReceivedEngineTime باشد، بازهم بروزرسانی انجام نخواهد شد. در این مورد ممکن است پیام معتبر بوده ولی نظم آن بهم خورده باشد که در این صورت بروزرسانی snmpEngineTime منطقی نیست.

توجه شود که سنکرونیزاسیون تنها وقتی انجام خواهد شد که سرویس اعتبارسنجی برای این پیام بکار گرفته شده باشد و پیام بتوسط HMAC معتبر تلقی گردد. این محدودیت ضروری بوده زیرا فضای اعتبارسنجی شامل msgAuthoritativeEngineID، msgAuthoritativeEngineBoots و msgAuthoritativeEngineTime بوده و بنابراین از اعتبار این سه مقدار اطمینان حاصل خواهد شد.

SNMPv3 این قید را بوجود می‌آورد که برای اجتناب از حملات تأخیر و بازخوانی، پیام بایستی در یک پنجرهٔ زمانی معقول دریافت شده باشد. پنجرهٔ زمانی بایستی تا حد ممکن کوچک انتخاب گردد و درآن دقت ساعت‌های فرستنده و گیرنده، تأخیرهای رفت و برگشت پیام و دورهٔ تناوب هماهنگی ساعت‌ها ملحوظ شده باشد. اگر پنجرهٔ زمانی خیلی کوچک انتخاب شود، پیام‌های معتبر، بصورت پیام‌های نامعتبر جلوه خواهند کرد. از سوی دیگر بزرگ بودن پنجرهٔ زمانی، آسیب‌پذیری به تأخیرهای بداندیشانهٔ پیام را افزایش خواهد داد.

مورد مهم‌تر یک گیرندهٔ مسئول را بیشتر بررسی می‌کنیم. آزمایش بهنگام بودن در مورد یک گیرندهٔ غیرمسئول کمی متفاوت خواهد بود. با هر پیام ورودی که معتبر تشخیص داده شود و msgAuthoritativeEngineID آن با snmpEngineID این موتور برابر باشد، موتور اندازهٔ msgAuthoritativeEngineBoots و msgAuthoritativeEngineTime پیام ورودی را با اندازه‌های snmpEngineBoots و snmpEngineTime نگهداری شده در این موتور مطابقت خواهد داد. پیام ورودی در صورتی خارج از پنجرهٔ زمانی تشخیص داده می‌شود که یکی از شرایط زیر صحیح باشد:

- snmpEngineBoots = 2^{31}-1 **یا**
- msgAuthoritativeEngineBoots≠snmpEngineBoots **یا**
- اندازهٔ msgAuthoritativeEngineTime از اندازهٔ snmpEngineTime بمیزان ۱۵۰± ثانیه تفاوت داشته باشد.

شرط اول می‌گوید که اگر snmpEngineBoots دارای اندازهٔ ماکزیمم خود باشد، هیچ پیام ورودی معتبر تلقی نخواهد شد. شرط دوم می‌گوید که یک پیام بایستی دارای زمان boot مساوی با زمان boot موتور محلی باشد. برای مثال اگر موتور محلی reboot شده و موتور دور از زمان reboot با موتور محلی هم‌آهنگ نشده باشد، پیام‌های واردشده از آن موتور دور معتبر تلقی نخواهد شد. شرط آخر می‌گوید که زمان پیام‌های واردشده بایستی از زمان محلی منهای ۱۵۰ ثانیه بیشتر، و از زمان محلی باضافهٔ ۱۵۰ ثانیه کمتر باشد.

اگر یک پیام خارج از پنجرهٔ زمانی قرار داشته باشد، آنگاه آن پیام نامعتبر تلقی شده و یک نمایش خطا (notInTimeWindow) به مدول فرستندهٔ پیام برگردانده خواهد شد.

در اینجا نیز همانندکنترل سنکرونیزاسیون، کنترل بهنگام بودن تنها وقتی انجام می‌شود که سرویس اعتبارسنجی فعال بوده و پیام معتبر تشخیص داده شود تا از صحت سرآیندهای پیام اطمینان حاصل شده باشد.

محلی کردن کلید

یکی از لازمه‌های استفاده از سرویس‌های اعتبارسنجی و محرمانگی SNMPv3 این است که برای هر ارتباط بین یک رئیس مستقر در یک موتور غیرمسئول و یک موتور مسئول دور، یک کلید سرّی اعتبارسنجی و یک کلید سرّی رمزنگاری بین این دو به اشتراک گذاشته شود. این کلیدها، یک کاربر در یک موتور غیرمسئول (معمولاً یک سیستم مدیریت) را قادر می‌سازد تا اعتبارسنجی و محرمانگی را با سیستم‌های مسئول دور (معمولاً سیستم‌های عامل) که کاربر آنها را مدیریت می‌کند، عملیاتی نماید. RFC 2574 روش خلق، بروزرسانی و مدیریت این کلیدها را بیان می‌کند.

برای اینکه وظیفهٔ رؤسا از نظر مدیریت کلید سهل‌تر شود، هر رئیس لازم است که تنها یک کلید اعتبارسنجی و تنها یک کلید رمزنگاری را نگهداری نماید. این کلیدها در یک MIB ذخیره نشده‌اند و از طریق SNMP قابل دست‌یابی نیستند. در این قسمت ابتدا به نحوهٔ تولید این کلیدها از یک کلمهٔ عبور نگاه می‌کنیم. سپس به مفهوم محلی کردن کلید می‌پردازیم که یک رئیس را قادر می‌سازد تا در حالی که بطور محلی تنها صاحب یک کلید اعتبارسنجی و یک کلید رمزنگاری است، اما با هر موتور دور، یک کلید اعتبارسنجی یکتا و یک کلید رمزنگاری یکتا را به اشتراک بگذارد. این دو تکنیک ابتدا در [BLUM97a] پیشنهاد شدند.

یک کاربر نیاز به یک کلید محرمانگی ۱۶- اُکتتی و یک کلید اعتبارسنجی ۱۶ یا ۲۰- اُکتتی دارد. برای کلیدهائی که صاحبان آنها انسان هستند، مطلوب این است که کاربر بتواند از کلمهٔ عبوری که قابل خواندن بتوسط انسان باشد، و نه از کلیدی که از یک رشته بیت درست شده است، استفاده کند. بهمین جهت RFC 2574 یک الگوریتم برای نگاشت یک کلمهٔ عبور به یک کلید ۱۶ یا ۲۰- اُکتتی را تعریف کرده است. USM هیچ محدودیتی برای خود کلمهٔ عبور قائل نشده است اما سیاست‌های مدیریتی محلی بایستی کاربران را به استفاده از کلمات عبوری که بسهولت قابل حدس زدن نباشند مقید سازد. تولید کلید از کلمهٔ عبور بصورت زیر انجام می‌شود:

- کلمهٔ عبور کاربر را گرفته و با تکرار اندازهٔ کلمهٔ عبور به هر مقدار که لازم است و قطع کردن آخرین اندازهٔ تکرارشده در صورت لزوم، یک رشته بیت با طول 2^{20} اُکتت (۱,۰۴۸,۵۷۶ اُکتت) بنام digest0 ایجاد کنید. مثلاً یک کلمهٔ عبور ۸- کاراکتری (2^3 اُکتت)، بایستی 2^{17} بار با خودش جمع رشته‌ای شود تا digest0 را بوجود آورد.
- اگر یک کلید ۱۶- اُکتتی مورد نیاز است، تابع درهم‌ساز MD5 مرتبط با digest0، را محاسبه کرده تا digest1 حاصل شود. اگر یک کلید ۲۰- اُکتتی مورد نیاز است، تابع درهم‌ساز SHA-1 مرتبط با digest0، را محاسبه کرده تا digest1 بدست آید. digest1 همان کلید کاربر است.

یک حسن این روش این است که حملهٔ همه جانبهٔ لغت‌نامه‌ای که در آن یک دشمن تلاش می‌کند تا کلمات عبور مختلف را امتحان کرده، کلید مرتبط با هر کدام را ساخته و آنگاه این کلید را روی دیتای اعتبارسنجی یا رمزنگاری موجود امتحان کند را بشدت کند می‌سازد. برای مثال اگر یک دشمن یک پیام معتبر را استراق سمع کند، او می‌تواندبا کلیدهای

مختلف اندازهٔ HMAC را محاسبه نماید. اگر بین این مقدار و دیتای موجود تطبیقی یافت شود، دشمن می‌تواند تصور کند که کلمهٔ عبور کشف شده است. عملیاتی که در بالا تشریح شد، زمان لازم برای چنین حمله‌ای را بطور چشمگیری افزایش می‌دهد.

مزیت دیگر این تکنیک این است که کلیدهای کاربر را از نوع سیستم مدیریت شبکه (NMS) مجزا می‌کند. هیچ NMS ای مجبور نیست که اندازهٔ کلیدهای کاربر را نگهداری کند. در عوض هروقت لازم است، کلید کاربر از روی کلمهٔ عبور او ساخته می‌شود. [BLUM97b] ملاحظات زیرکه انگیزهٔ استفاده از کلمهٔ عبور مستقل از NSM است را لیست کرده است:

- اگر قرار باشد بجای تولید کلید از روی کلمهٔ عبور، خود کلید در جائی ذخیره گردد، یک روش این است که کلیهٔ کلیدهای سرّی در یک مخزن نگهداری شود. چنین روشی روی قابلیت اعتماد کل سیستم تأثیر سوء داشته زیرا ممکن است که خود مخزن در زمان مورد نیاز در دسترس نبوده و عیب‌یابی سیستم را غیرممکن سازد.
- از سوی دیگر اگر برای رفع مشکل بالا، مخازن متعددی ایجاد شوند، این امر با فراهم آوردن اهداف متعدد حمله برای دشمنان، امنیت را شکننده‌تر خواهد کرد.
- اگر یک مخزن متمرکز و یا چندین مخزن پراکنده مورد استفاده قرار گیرد، آنها را بایستی در محل‌های امن نگهداری کرد. این امر می‌تواند فرصت ایجاد یک "forward camp" در خلال عملیات اطفاء حریق (یعنی عیب‌یابی سیستم در هنگامی که بخش‌های غیرقابل پیش‌بینی از شبکه غیرعملیاتی شده و/ یا برای مدت زمان غیرقابل پیش‌بینی در دسترس نیستند) را کاهش دهد.

یک کلمهٔ عبور منفرد می‌تواند برای تولید یک کلید منفرد، هم برای اعتبارسنجی و هم رمزنگاری، بکار رود. روش امن‌تر این است که از دو کلمهٔ عبور استفاده شود که یکی برای تولید کلید اعتبارسنجی و دیگری برای تولید کلید رمزنگاری بکار رود.

یک کلید محلی شده، در RFC 2574، بصورت یک کلید سرّی مشترک بین یک کاربر و یک موتور SNMP مسئول تعریف شده است. هدف این است که کاربر لازم باشد تنها یک کلید (یا دو کلید، اگر اعتبارسنجی ومحرمانگی مورد نیاز است) را نگهداری کرده و بنابراین تنها لازم باشد فقط یک (یا دو) کلمهٔ عبور را بخاطر بسپارد. اما در واقع اَسرار به‌اشتراک گذاشته شده بین یک کاربر بخصوص با هر موتور SNMP مسئول متفاوت است. عملی که بتوسط آن یک کلید منفرد کاربر به چندین کلید متفاوت یکتا که هرکدام مربوط به یک موتور SNMP دور هستند تبدیل می‌شود را محلی کردن کلید (key localization) گویند.[BLUM97a] انگیزه‌های این استراتژی را ذکر کرده که در اینجا آنها را بصورت خلاصه بیان می‌کنیم.

برای مدیریت کلید اهداف زیر را می‌توان تعریف نمود:

- هر سیستم عامل SNMP در یک شبکهٔ گسترده، دارای یک کلید یکتا برای هر کاربر معتبری است که می‌خواهد آن را مدیریت نماید. اگر کاربران متعددی بعنوان مدیران معتبر شناخته می‌شوند، عامل دارای یک کلید اعتبارسنجی یکتا و یک کلید رمزنگاری یکتا برای هر کاربر است. در نتیجه اگر کلید یک کاربر لو برود، کلیدهای کاربران دیگر لورفته نخواهند بود.
- کلیدهای یک کاربر برای عامل‌های مختلف متفاوت‌اند. بنابراین اگر یک عامل لو برود، تنها کلیدهای کاربر برای آن عامل لورفته و کلیدهای متعلق به عاملان دیگر لورفته نخواهند بود.
- مدیریت شبکه از هر نقطهٔ شبکه، صرف نظر از پیکربندی سیستم مدیریت شبکه (NMS)، امکان‌پذیر است. این امر به کاربر اجازه می‌دهد تا عملیات مدیریتی را از هر ایستگاه مدیریت انجام دهد. این قابلیت بتوسط الگوریتم تبدیل کلمهٔ عبور- به- کلید که قبلاً تشریح گردید امکان‌پذیر است.

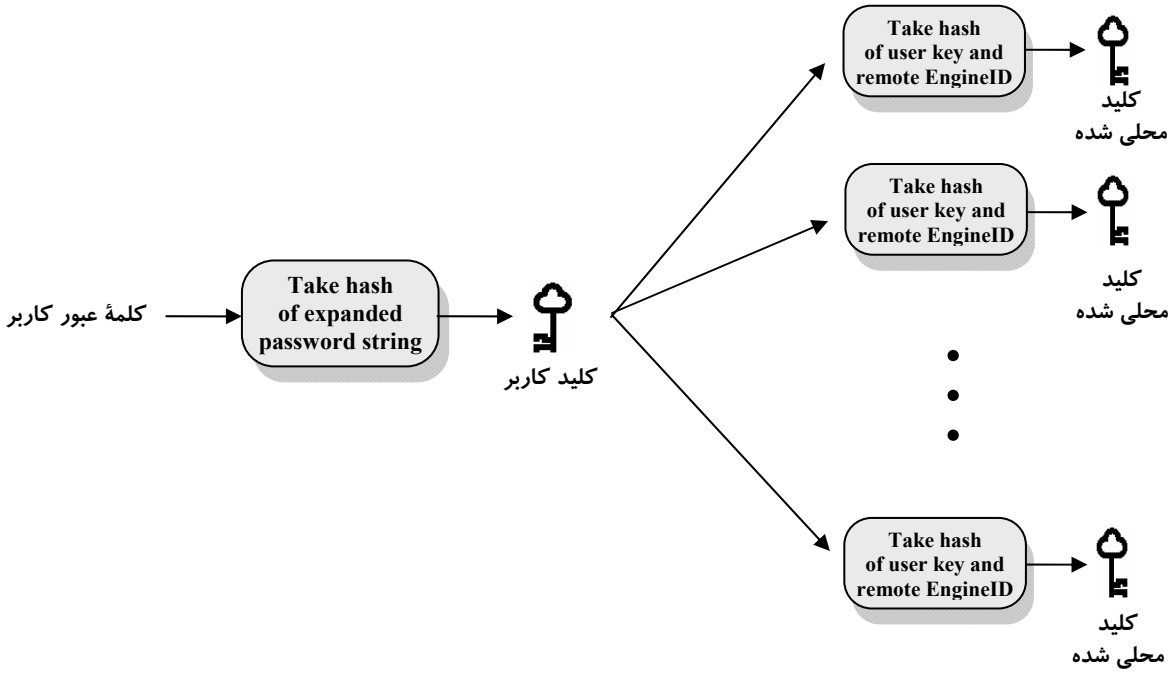
همچنین می‌توان به موارد زیر که بایستی از آنها اجتناب شود اشاره کرد:

- یک کاربر لازم است تعداد زیادی کلید را بخاطر سپارد (یا مدیریت کند) که این تعداد با اضافه شدن عوامل جدید مدیریت شده، بسرعت رشد می‌کند.
- یک دشمن که کلید مرتبط با یک عامل را کشف کند، قادر خواهد بود تا خود را جای هر عامل دیگر برای هر کاربر، و جای هر کاربر برای هر عامل دیگر جا بزند.

برای رسیدن به اهداف و ملاحظات ذکرشده، یک کلید منفرد کاربر بتوسط یک تابع یک- طرفهٔ برگشت‌ناپذیر (یعنی یک تابع hash) به فرم کلیدهای محلی شدهٔ متفاوت برای موتورهای مسئول متفاوت (عوامل متفاوت) درمی‌آید. روش کار چنین است:

- رشته بیت digest2 را از جمع رشته‌ای digest1 (قبلاً تشریح گردید)، اندازهٔ snmpEngineID موتور مسئول، و digest1 تولید کنید.
- اگر یک کلید ۱۶- اُکتتی مورد نیاز است، تابع درهم‌ساز MD5 رشتهٔ digest2 را بسازید. اگر یک کلید ۲۰- اُکتتی مورد نیاز است تابع درهم‌ساز SHA-1 آن را حساب کنید. خروجی بدست‌آمده، کلید محلی شدهٔ کاربر خواهد بود.

کلید محلی شده آنگاه می‌تواند در سیستم عامل به فرم امنی پیکربندی شود. نظر به یک- طرفه بودن MD5 و SHA-1 برای یک دشمن امکان نخواهد داشت که حتی در صورت کشف یک کلید محلی شده، کلید منفرد کاربر را پیدا کند. شکل ۱۱-۸ محلی کردن کلید را نشان می‌دهد.



شکل ۱۱-۸ محلی کردن کلید

کنترل دست‌یابی مبتنی بر منظر (View-Based)

کنترل دست‌یابی یک عمل امنیتی است که در سطح PDU انجام می‌شود. یک دستورالعمل کنترل دست‌یابی، مکانیسم‌هائی را برای تعیین اینکه آیا دست‌یابی به یک موضوع مدیریت شده در یک MIB محلی بتوسط یک رئیس دور مجاز است یا خیر تعریف می‌کند. مکانیسم‌های متعدد و متصوری را می‌توان برای کنترل دست‌یابی تعریف کرد. اسناد SNMPv3 مدل کنترل دست‌یابی view-based access control model (VACM) را تعریف می‌کند. VACM خود از یک MIB که خط‌مشی کنترل دست‌یابی برای این عامل را تعریف کرده است، استفاده کرده و استفاده از پیکربندی دور را ممکن می‌سازد.

VACM دارای دو مشخصهٔ مهم است:

- VACM تعیین می‌کند که آیا دست‌یابی به یک موضوع مدیریت شده در یک MIB محلی بتوسط یک رئیس دور مجاز است.
- VACM از یک MIB استفاده می‌کند که
  - خط‌مشی کنترل دست‌یابی برای این عامل را تعریف می‌کند.
  - استفاده از پیکربندی دور را ممکن می‌سازد.

عناصر یک مدل VACM

پنج عنصر که VACM را تشکیل می‌دهنددر RFC 2575 تعریف شده‌اند: گروه‌ها، سطح امنیتی، مقوله‌ها، منظرMIB و خط مشی دست‌یابی.

یک **گروه** (group) مجموعه‌ای از هیچ یا چند جفت<securityModel,securityName> را تعریف می‌کند که موضوعات مدیریتی SNMP می‌تواند از سوی آنها مورد دست‌یابی قرار گیرد. یک securityName به یک رئیس اشاره دارد و حقوق دست‌یابی تمام رؤسا در یک گروه یکسان است. هر گروه یک نام groupName یکتا دارد. مقولهٔ گروه یک ابزار سودمند برای طبقه‌بندی مدیران برحسب حقوق دست‌یابی آنهاست. بعنوان مثال، تمام مدیران ردهٔ بالا ممکن است دارای مجموعه‌ای از حقوق دست‌یابی بوده که با مجموعهٔ حقوق دست‌یابی مدیران میانی متفاوت باشد.

هر ترکیبی از securityModel و securityName حداکثر می‌تواند متعلق به یک گروه باشد. یعنی برای این عامل، رئیسی که ارتباطات او بتوسط یک securityModel خاص حفاظت می‌شود تنها می‌تواند در یک گروه جای گیرد.

حقوق دست‌یابی یک گروه می‌تواند برحسب **سطح امنیتی** (security level) پیام درخواست، متفاوت باشد. برای مثال، یک عامل ممکن است دست‌یابی read-only از یک پیام درخواست اعتبارسنجی نشده را بپذیرد ولی برای دست‌یابی به write، اعتبارسنجی را طلب نماید. علاوه برآن برای موضوعات حساس معینی، عامل ممکن است استفاده از سرویس محرمانگی برای درخواست و پاسخ آن را طلب نماید.

یک **مقوله MIB** (MIB context) یک زیرمجموعه با یک نام مشخص در یک MIB محلی است. ایجاد مقوله‌ها یک روش مفید برای گردآوردن موضوعاتی با سیاست‌های دست‌یابی یکسان تحت عنوان یک نام است.

مقوله، مفهومی مرتبط با کنترل دست‌یابی است. وقتی یک ایستگاه مدیریت با یک عامل ارتباط برقرارکرده تا به اطلاعات مدیریتی موجود در آن عامل دست یابد، آنگاه تعامل بین یک مسئول مدیریت با موتور SNMP عامل ایجاد می‌گردد و امتیازات کنترل دست‌یابی متعلق به این رئیس در این مقوله در یک منظر MIB گنجانده شده است. مقوله‌ها دارای مشخصه‌های کلیدی زیراند:

- یک موجودیت SNMP که بطور یکتا بتوسط یک contextEngineID مشخص می‌شود و ممکن است بیش از یک مقوله را نگهداری کند.
- یک موضوع و یا موردی از یک موضوع ممکن است در بیش از یک مقوله ظاهر شود.
- وقتی مقوله‌های متعددی وجود دارد، برای شناسائی موردی از یک موضوع، contextName و contextEngineID آن علاوه بر نوع موضوع و مورد آن بایستی شناسائی گردند.

اغلب علاقه‌مندیم تا دست‌یابی یک گروه خاص به زیرمجموعه‌ای از موضوعات مدیریت شده در یک عامل را محدود سازیم. برای رسیدن به این هدف، دست‌یابی به یک مقوله بتوسط **منظر MIB (view)** صورت می‌پذیرد که مجموعهٔ خاصی از موضوعات مدیریت شده (و بطور اختیاری موارد یک موضوع) را تعریف می‌کند. VACM از یک تکنیک قدرتمند و قابل انعطاف برای تعریف منظر MIB استفاده کرده که بر مفاهیم زیرشاخه‌های منظر و خانواده‌های منظر استوار است. منظر MIB بصورت یک مجموعه از زیرشاخه‌ها تعریف می‌شود که هر زیرشاخه یا در منظر وجود داشته و یا در آن موجود نیست.

موضوعات مدیریت شده در یک پایگاه داده محلی بصورت سلسله مراتبی و یا درختی و بر اساس شناسه‌های موضوعات سازمان می‌یابند. این پایگاه دادهٔ محلی شامل یک زیرمجموعه از انواع موضوعات تعریف شده بر اساس Internet-standard Structure of Management Information (SMI) بوده و شامل موارد موضوعاتی است که شناسه‌های آن منطبق با قراردادهای SMI است.

SNMPv3 شامل مفهومی به نام زیرشاخه است. یک زیرشاخه، بطور ساده یک گره در یک سلسله مراتب نام‌های MIB باضافهٔ تمام عناصر مرتبط با آن است. بطور رسمی‌تر، یک زیرشاخه ممکن است بصورت مجموعهٔ تمام موضوعات و موارد مختلف آنها بوده که پیشوند مشترک ASN.1 OBJECT IDENTIFIER در نام‌های آنان وجود دارد. طویل‌ترین پیشوند مشترک تمام موارد در یک زیرشاخه، شناسهٔ موضوع گرهٔ مادر در آن زیرشاخه است.

به هر مورد موجود در vacmAccessTable سه منظر MIB مرتبط است که یکی مربوط به دست‌یابی خواندن، یکی مربوط به دست‌یابی نوشتن و سومی مربوط به دست‌یابی اخطار است. هر منظر MIB شامل یک مجموعه از زیرشاخه‌های منظر است. هر زیرشاخهٔ منظر در منظر MIB یا حضور دارد و یا حضور ندارد. یعنی منظر MIB یا شامل تمام موارد موضوعی در زیرشاخه هست یا نیست. علاوه بر آن یک view mask برای این امر تعریف شده تا میزان اطلاعات پیکربندی لازم برای کنترل دست‌یابی به موارد کم‌ارزش (مثلاً کنترل دست‌یابی در سطح موردی یک موضوع) را کاهش دهد.

VACM یک موتور SNMP را قادر می‌سازد تا طوری پیکربندی شود که یک مجموعهٔ خاص از حقوق دست‌یابی که یک **خط‌مشی دست‌یابی** را تشکیل می‌دهد گردهم آورد. تعیین دست‌یابی به فاکتورهای زیر وابسته است:

- یک **رئیس** که درخواست دست‌یابی می‌کند. VACM یک عامل را قادر می‌سازد تا امتیازات دست‌یابی متفاوت برای کاربران مختلف قائل شود. برای مثال، یک سیستم مدیریت مسئول پیکربندی کل شبکه ممکن است امتیاز دست‌یابی وسیع تغییر اقلام در یک MIB محلی را داشته باشد درحالی‌که یک مدیر میانی با مسئولیت پایش، تنها حق read-only داشته و شاید تنها حق داشته باشد که به بخشی از MIB محلی دست یابد. همانطور که قبلاً بحث شد، رؤسا در گروه‌های مختلف دسته‌بندی شده و خط‌مشی دست‌یابی در رابطه با یک گروه تعریف می‌شود.
- **سطح امنیتی** که پیام درخواست SNMP در آن سطح قرار دارد. معمولاً یک عامل نیازمند به استفاده از اعتبارسنجی برای پیام‌هائی که شامل درخواست set (عمل نوشتن) هستند می‌باشد.

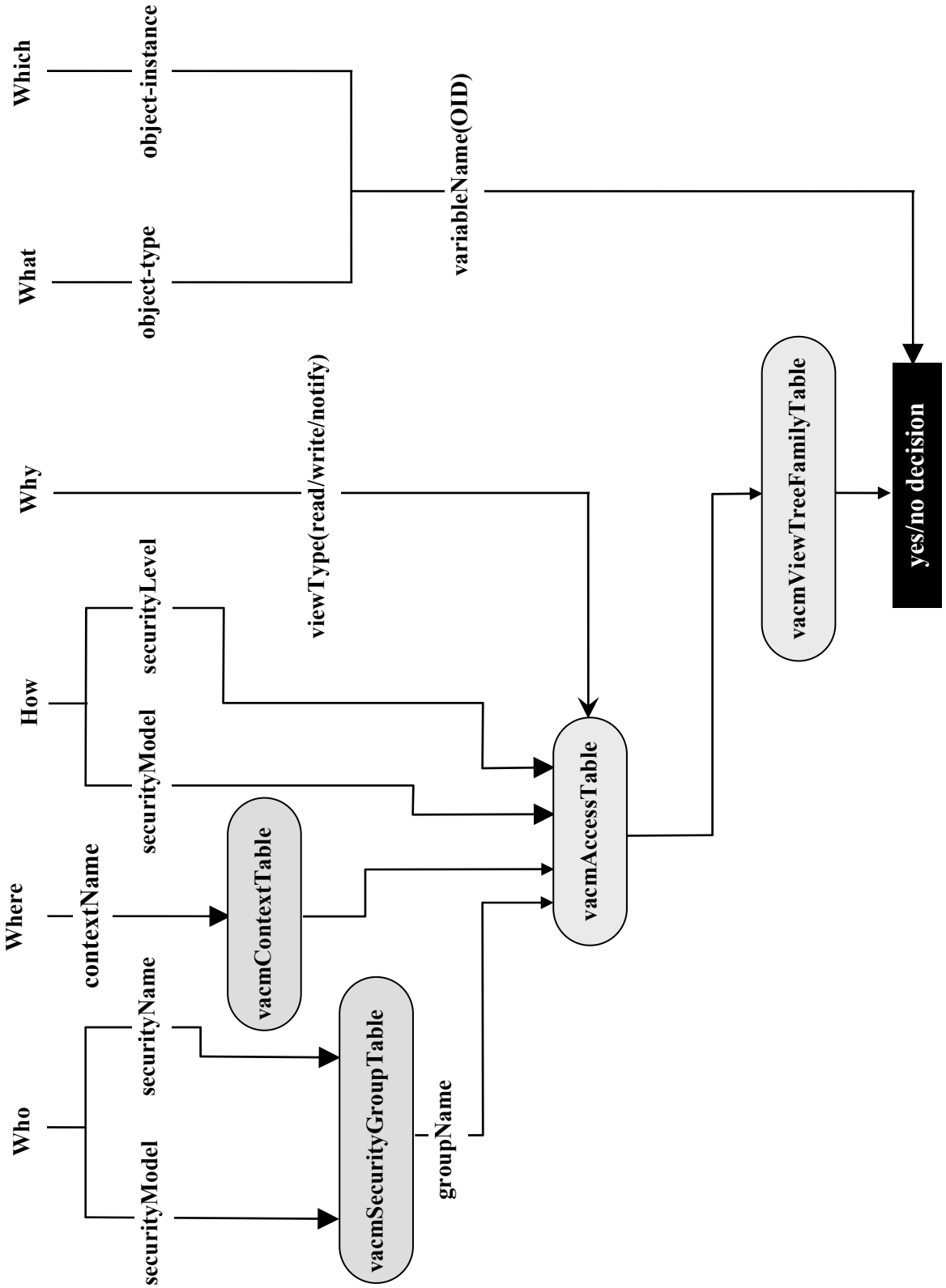
- **مدل امنیتی** استفاده شده برای پردازش پیام درخواست است. اگر مدل‌های امنیتی متعددی در یک عامل پیاده‌سازی شده باشد، عامل ممکن است طوری پیکربندی شود که به پیام‌هائی که با مدل‌های مختلف پردازش شده‌اند، سطوح دست‌یابی متفاوت تخصیص دهد. مثلاً اگر درخواست از طریق USM رسیده باشد اقلام معینی قابل دست‌یابی بوده در حالی که اگر مدل امنیتی SNMPv1 باشد این اقلام قابل دست‌یابی نباشند.
- **مقولهٔ MIB** برای این درخواست.
- **نوع موضوع** که دست‌یابی به آن درخواست شده است. برخی موضوعات، اطلاعات حیاتی‌تر و یا حساس‌تری را نسبت به بقیه نگاه می‌دارند و بنابراین خط‌مشی دست‌یابی بایستی وابسته به نوع موضوع مورد تقاضا باشد.
- **نوع دست‌یابی** تقاضا شده (خواندن، نوشتن، هشدار). خواندن، نوشتن و هشدار عملیات مدیریتی متمایز بوده و خط‌مشی‌های کنترل دست‌یابی مختلف ممکن است به هریک از آنها اعمال شود.

رَوَند کنترل دست‌یابی

یک کاربرد SNMP، VACM را از طریق isAccessAllowed primitive با پارامترهای ورودی securityModel، securityName، securityLevel، viewType، contextName و variableName بکار می‌گیرد. تمام این پارامترها برای تصمیم‌گیری نسبت به کنترل دست‌یابی لازم‌اند. بعبارت دیگر زیرسیستم کنترل دست‌یابی بنحوی تعریف شده است تا یک ابزار بسیار انعطاف‌پذیر برای پیکربندی کنترل دست‌یابی در عامل از طریق شکستن تصمیمات کنترل دست‌یابی به شش متغیر مجزا بوجود آورد.

شکل ۱۲-۸ که از شکلی در RFC 2575 اقتباس شده است، یک روش مفید برای مشاهدهٔ متغیرهای ورودی بوده و نشان می‌دهد که چگونه جداول مختلف در یک VACM MIB در اخذ تصمیمات کنترل دست‌یابی دخالت می‌کنند.

- **who**: ترکیب securityModel و securityName، معرفی‌کنندهٔ چه *کسی* در این عملیات است. این مقوله در واقع یک رئیس را تعریف کرده که ارتباطات او بتوسط یک securityModel حفاظت می‌شود. ترکیب بالا در این موتور SNMP حداکثر به یک گروه تعلق دارد. vacmSecurityToGroupTable با داشتن securityModel و securityName، groupName را استخراج می‌کند.
- **where**: contextName تعیین می‌کند که موضوع مدیریتی مطلوب در *کجا* بایستی یافت شود. vacmContextTable شامل یک لیست از contextNameهای شناخته شده است.
- **how**: ترکیب securityModel و securityLevel تعریف می‌کنند که *چگونه* درخواست ورودی و Inform PDU حفاظت شده‌اند. ترکیب who، where و how هیچ و یا یک قلم از vacmAccessTable را مشخص می‌کند.
- **why**: viewType مشخص می‌سازد که *چرا* دست‌یابی درخواست شده است: برای خواندن، نوشتن و یا هشدار. مورد انتخاب شده در vacmAccessTable شامل یک MIB viewName برای هریک از سه نوع عمل بالاست و viewType برای انتخاب یک viewName مشخص است. این viewName، منظرMIB مناسب از vacmViewTreeFamilyTable را انتخاب می‌کند.
- **what**: variableName یک شناسهٔ موضوعات است که پیشوند آن یک نوع موضوع مشخص و پسوند آن یک مورد از موضوع مشخص را تعریف می‌کند. نوع موضوع نشان می‌دهد که چه نوع اطلاعات مدیریتی درخواست شده است.



شکل ۱۲-۸ منطق VACM

- **which**: مورد موضوع نشان می‌دهد که *کدام* قلم خاص اطلاعات درخواست شده است.

بالاخره variableName با منظر MIB استخراج شده مقایسه می‌گردد. اگرvariableName با یک عنصر موجود در منظر MIB تطبیق داشته باشد، آنگاه دست‌یابی اعطا می‌گردد.

انگیزش

مفاهیمی که VACM را می‌سازند بنظر می‌رسد که منجر به تعریف پیچیده‌ای از کنترل دست‌یابی می‌گردند. انگیزهٔ معرفی چنین مفاهیمی، روشن ساختن ارتباطات موجود در کسب اطلاعات مدیریت و به حداقل رساندن نیازهای ذخیره‌سازی و پردازش در یک عامل است. برای فهم چنین انگیزه‌هائی به مورد زیر توجه کنید. در SNMPv1 مفهوم جامعه برای معرفی اطلاعات مرتبط با امنیت بکار می‌رود:

- هویت موجودیت درخواست‌کننده (ایستگاه مدیریت)
- هویت موجودیت انجام‌دهندهٔ درخواست (عاملی که برای خود و یا برای یک موجودیت پروکسی شده عمل می‌کند)
- هویت محلی که اطلاعات مدیریتی بایستی از آنجا دریافت شود (عامل یا واحد پروکسی شده)
- اطلاعات اعتبارسنجی
- اطلاعات کنترل دست‌یابی (مجاز بودن برای انجام عملیات درخواست شده)
- اطلاعات منظر MIB

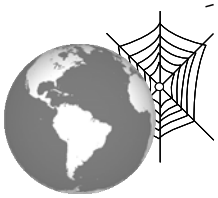
با جمع کردن تمام این مفاهیم در یک متغیر منفرد، انعطاف‌پذیری و کارآئی از دست رفته‌اند. VACM همین مجموعه اطلاعات مرتبط با امنیت را، با استفاده از متغیرهای مشخص برای هر مورد فراهم آورده است. این یک مزیت قابل توجه نسبت به SNMPv1 است. این روش مفاهیم مختلف را از یکدیگر مجزا نموده تا اندازهٔ متغیرها بتوانند بطور جداگانه به هر مورد تخصیص یابند.

۸-۴ منابع مطالعاتی

[STAL99] یک بررسی کامل و تحلیلی از SNMP، SNMPv2 و SNMPv3 ارائه داده است. این کتاب همچنین مروری بر تکنولوژی مدیریت شبکه دارد.

STAL99 Stallings, W. *SNMP, SNMPv2, SNMPv3, and RMON 1 and 2*. Reading, MA: Addison-Wesley, 1999.

وِب سایت‌های مفید



- **SNMPv3 Web site**: این سایت بتوسط Technical University of Braunschweig نگهداری می‌شود. این سایت لینک‌هائی به RFCها و پیش‌نویس‌های اینترنت، کپی پیشنهادهای داده شده برای تغییرات به گروه‌های کاری، و همچنین لینک‌هائی به فروشندگان پیاده‌سازی‌های SNMPv3 فراهم آورده است.
- **The Simple Web site**: این سایت بتوسط University of Twente نگهداری می‌شود. منبع خوبی برای اطلاعات راجع به SNMP است که اشاره به پیاده‌سازی‌های عمومی این پروتکل داشته و همچنین شامل لیستی از کتاب‌ها و مقالات مرتبط است.

۸-۵ واژه‌های کلیدی، سؤالات مرورکنندهٔ بحث و مسائل

واژه‌های کلیدی

مدل پردازش پیام	message processing model	خط‌مشی دست‌یابی	access policy
مدیریت شبکه	network management	عامل	agent
پروکسی	proxy	جامعه	community
Simple Network Management Protocol (SNMP) پروتکل سادهٔ مدیریت شبکه		نام جامعه	community name
مدل امنیتی کاربر	User security model (USM)	محلی کردن کلید	key localization
view-based access control model (VACM) مدل کنترل دست‌یابی مبتنی بر منظر		management information base (MIB) پایگاه اطلاعات مدیریت	
		ایستگاه مدیریت	management station

سؤالات مرورکننده بحث

۸-۱ یک معماری مدیریت شبکه با چه مفهومی یکپارچه تلقی می‌شود؟

۸-۲ عناصر کلیدی مدل SNMP کدامند؟

۸-۳ یک MIB چیست؟

۸-۴ چه قابلیت‌ها یا فرامین اصلی در SNMPv1 فراهم آمده‌اند؟

۸-۵ وظیفهٔ یک پروکسی SNMP چیست؟

۸-۶ مفهوم جامعهٔ SNMPv1 را بطور مختصر بیان کنید.

۸-۷ رابطهٔ بین SNMPv1، SNMPv2 و SNMPv3 چیست؟

۸-۸ USM برای مقابله با چه تهدیدهائی طراحی شده است؟

۸-۹ تفاوت بین یک موتور مسئول با یک موتور غیرمسئول کدام است؟

۸-۱۰ محلی کردن کلید چیست؟

۸-۱۱ عناصری که VACM را می‌سازند لیست کرده و بطور مختصر تعریف نمائید.

مسائل

۸-۱ SNMPv1 یک نوع دیتا که Gauge نامیده می‌شود را تعریف کرده و آن را بصورت زیر تشریح کرده است:

این نوع دیتا با کاربرد وسیع، نمایشگر یک عدد صحیح غیرمنفی است که می‌تواند زیاد یا کم شده ولی به یک مقدار ماکزیمم latch می‌گردد. استاندارد مقدار ماکزیمم $۲^{۳۱}-۱$ (۴,۲۹۴,۹۶۷,۲۹۵ دهدهی) برای Gauge را تعیین کرده است.

متأسفانه کلمهٔ *latch* تعریف نشده است و این امر دو تعبیر مختلف را بوجود آورده است. استاندارد SNMPv2 این ابهام را با تعریف زیر رفع کرده است:

اندازهٔ یک Gauge وقتی دارای ماکزیمم مقدار خود است که اطلاعاتی که مدل می‌شود بزرگتر یا مساوی آن مقدار ماکزیمم باشد. اگر اطلاعات مدل شده پس از آن از مقدار ماکزیمم کمتر شود، Gauge هم کاهش می‌یابد.

الف- تعبیر دیگر چیست؟

ب- در نقاط قوت و ضعف این دو تعبیر بحث کنید.

۸-۲ در SNMPv1، برای هر موضوع در یک MIB یک MIB access Category تعریف شده است که هریک از اندازه‌های read-only، write-only و not-accessible می‌تواند به آن اختصاص یابد. یک read با یک عمل get یا trap انجام شده و یک write با عمل set صورت می‌پذیرد. برای write-only، موضوع ممکن است برای عملیات get و trap در دسترس باشد ولی این امر وابسته به پیاده‌سازی است. MIB Access Category ماکزیمم دست‌یابی مجاز به یک موضوع را تعیین می‌کند ولی در تسهیلات جامعه‌ای SNMPv1، Access Mode ممکن است برای یک پروفایل جامعهٔ بخصوص این دست‌یابی را بازهم محدودتر کند. در جدول زیر نوع دست‌یابی مجاز برای هر مورد را تعیین کنید.

MIB Access Category	SNMP Access Mode	
	READ-ONLY	READ-WRITE
read-only		
Read-write		
Write-only		
Not-accessible		

۸-۳ **الف**- RFC 2574 چنین بیان می‌کند که برای یک موتور غیرمسئول، اندازه‌های msgAuthoritativeEngineBoots و msgAuthoritativeEngineTime در یک سرآیند پیام خروجی تنها در صورتی تنظیم می‌شوند که پیام بخواهد بتوسط گیرندهٔ مسئول اعتبارسنجی گردد. چرا این محدودیت دارای معنی است؟

ب-از طرف دیگر، برای یک پیام Response از سوی یک موتور مسئول، اندازه‌های msgAuthoritativeEngineBoots و msgAuthoritativeEngineTime در سرآیند پیام‌های خارج‌شونده همیشه تنظیم میشوند. چرا بایستی چنین باشد؟

۸-۴ RFC 2574 چنین تعیین میکندکه سنکرونیزاسیون ساعت (بروزرسانی ساعت محلی بر حسب مقادیر ورودی) قبل از تأئید پنجرهٔ زمانی (کنترل اینکه پیام ورودی به هنگام است) صورت پذیرد. این امر بدین معنی است که یک موتور غیر مسئول می‌تواند تخمین خود از ساعت موتور مسئول را حتی در صورتی که پیام در خارج از پنجرهٔ زمانی مجاز قرار داشته باشد، بروزرسانی نماید. از زمان انتشار RFC، نظرات مخالف مستمری در این مورد در لیست پستی SNMPv3 وجود داشته است ولی تا زمان کتابت این کتاب بنظر نمی‌رسد که تغییری در بیان استاندارد رخ دهد. آموزنده است که به این مورد نگاهی بیندازیم. با تعاریف زیر:

MAEB = msgAuthoritativeEngineBoots
MAET = msgAuthoritativeEngineTime
SEB = تخمین محلی از snmpEngineBoots موتور مسئول دور
SET = تخمین محلی از snmpEngineTime موتور مسئول دور
LRET = snmpEngineTime اخیرترین

آنگاه فرض کنید که یک موتور غیرمسئول پیامی را دریافت می‌کند که برحسب آن

$$(MAEB = SEB) \text{ AND } [LRET < MAET < (SET - 150)]$$

آنگاه شرایط برای بروزرسانی ساعت مهیا بوده و بنابراین این کار انجام می‌شود:

SET := MAET; LRET := MAET

حال وقتی به کنترل پنجرهٔ زمانی می‌رسیم، داریم

$$(MAEB = SEB) \text{ AND } (MAET = SET)$$

و بنابراین پیام را بهنگام اعلام می‌کنیم. حال فرض کنید که ابتدا پنجرهٔ زمانی کنترل شود، آیا پیام را بهنگام و یا نابهنگام اعلام می‌کردیم؟

۸ -۵ در نسخهٔ اولیهٔ انتشاریافتهٔ مشخصه‌های USM (RFC 2274)، در بحث پیرامون روش‌های سنکرونیزاسیون ساعت و تأئید پنجرهٔ زمانی، چنین آمده است: «توجه شود که این رَویه، اجازهٔ سنکرونیزاسیون اتوماتیک ساعت در صورتی که موتور SNMP غیرمسئول خارج از هماهنگی بوده و موتور SNMP مسئول بیش از ۱۵۰ ثانیه عقب‌تر از موتور SNMP غیرمسئول باشد، را نمی‌دهد.» این جمله پس از آنکه نویسندهٔ این کتاب به گروه کاری مربوطه خاطرنشان کرد که این امر همیشه صحیح نمی‌باشد از نسخهٔ بازنگری شده (RFC 2574) حذف گردید. از مثال مسألهٔ ۴-۸ استفاده کرده و نشان دهید که واقعاً بیان قبل همیشه صحیح نیست.

۸-۶ SNMPv3 فرض می‌کند که روش امنی برای تحویل کلیدهای محلی شده به سیستم‌های (عامل) اعتبارسنجی شده وجود دارد. این تحویل امن فراتر از حوزهٔ SNMPv3 است. این کار ممکن است بصورت دستی و یا بتوسط پروتکل امن دیگری انجام شود. وقتی یک کلید اولیه (یا یک جفت کلیدبرای اعتبارسنجی و محرمانگی) به یک عامل تحویل شد، SNMPv3 مکانیسمی را برای بروزرسانی کلیدها بصورت امن فراهم می‌آورد. مطلوب این است که برای ارتقاء امنیت، کلیدها هرچندوقت یکبار بروزرسانی شوند. یک کاربر می‌تواند از جانب خودش داوطلب تغییر کلید شده و برای این منظور کلمهٔ عبور جدیدی را ارائه دهد. بصورت مشابه، سیستم مدیریت شبکه (NMS) می‌تواند این کار را آغاز کرده و درخواست کلمهٔ

عبور جدید نماید. در هر صورت، کلید کاربر در NMS بروزرسانی می‌شود. سپس NMS می‌تواند یک کلید محلی برای هریک از عوامل ارتباطی را محاسبه نماید. پس از آن، NMS بایستی با هر عامل بطور امن ارتباط یابد تا عامل را وادار کند که کلید محلی شده خود را بروز برساند. دو حالت اختیاری زیر امکان‌پذیر است:

- کلید جدید را با استفاده از کلید قدیم بعنوان کلید رمز، رمزنگاری نماید.
- نوعی تابع یک- طرفه را بکار گرفته تا یک اندازه از کلید قدیمی درست شود. این اندازه را با کلید جدید XOR کرده و نتیجه را برای عامل بفرستد. عامل آنگاه می‌تواند نتیجهٔ ورودی را با همان تابع که به کلید قدیم اعمال شده است XOR کرده تا کلید جدید را بدست آورد.

SNMPv3 از فرمی به روش دوم استفاده می‌کند. مزیت این روش نسبت به روش اول چیست؟

۸-۷ روش برخورد SNMPv3، شامل استفاده از یک موضوع KeyChange در MIB سیستم هدف است. یک رئیس دور یا NMS این موضوع را تنظیم کرده که پس از آن بتوسط عامل بصورت خودکار برای بروزرسانی کلید مرتبط بکار می‌رود. الگوریتم شامل دو فاز است که یک فاز آن در موتور متقاضی و فاز دیگر آن در موتور عامل دور صورت می‌پذیرد.

عمل وقتی شروع می‌شود که یک متقاضی علاقه‌مند است تا یک کلید موجود keyOld را با یک کلید جدید keyNew تعویض کند. متقاضی قدم‌های زیر را برمی‌دارد:

۱- یک اندازهٔ random از یک تولیدکنندهٔ اعداد شبه تصادفی و یا تولیدکنندهٔ اعداد واقعی تصادفی تولید می‌کند.

۲- مقدار زیر را محاسبه می‌کند

$$digest = Hash(keyOld \;||\; random)$$

که در آن Hash یا MD5 و یا SHA-1 است که بستگی به این دارد که آیا یک کلید ۱۶- اُکتتی و یا ۲۰- اُکتتی مورد نیاز بوده و || علامت جمع رشته‌ای است.

۳- محاسبه می‌کند

$$delta = digest \oplus keyNew$$

$$protocolKeyChange = (random \;||\; delta)$$

که در آن ⊕ عمل XOR است.

۴- اندازهٔ protocolKeyChange آنگاه در یک فرمان Set به یک عامل فرستاده شده تا موضوع KeyChange در MIB عامل را بروز برساند.

عامل برای بروز رساندن کلید، با اندازهٔ ورودی چه باید بکند؟

۸-۸ یک روش ساده‌تر از آنچه در مسألهٔ قبل عنوان شد این است که keyOld را با keyNew بصورت XOR درآورده و نتیجه را ارسال کند. گیرنده آنگاه اندازهٔ XOR مقدار دریافت شده با keyOld را محاسبه کرده تا keyNew را تولید کند. چون یک حمله‌کننده keyOld را نمی‌داند، او نمی‌تواند که keyNew را بدست آورد. مزایای استفاده از عدد تصادفی و تابع امن یک- طرفه در مسألهٔ ۷-۸ در مقایسه با این روش چیست؟

قسمت سوم

امنیت سیستم

قسمت سوم کتاب به مقوله‌های امنیتی در سطح سیستم می‌پردازد که شامل تهدیدها و روش‌های مقابله با آنها از سوی مهاجمین و ویروس‌ها می‌باشد. این قسمت همچنین استفاده از دیوارهای آتش و سیستم‌های معتمد را مورد بحث قرار می‌دهد.

فصل ۹ مهاجمین

فصل ۹ مجموعهٔ متنوعی از تهدیدهائی که به دلیل وجود نقاط آسیب‌پذیر در سیستم‌های کامپیوتری مبتنی بر شبکه، از سوی نفوذگران متوجه سرویس‌ها و دست‌یابی به آنها می‌باشد، را مورد بررسی قرار می‌دهد. این فصل با بحثی در مورد انواع حملاتی که می‌تواند بتوسط کاربران غیر مجاز، یا مهاجمین، روی سیستم انجام شود شروع شده و روش‌های جلوگیری و تشخیص آنها را تحلیل می‌نماید. این فصل همچنین مقولهٔ مرتبط مدیریت کلمهٔ عبور را پوشش می‌دهد.

فصل ۱۰ نرم‌افزارهای بداندیش

فصل ۱۰ تهدیدهای نرم‌افزاری به سیستم‌ها، با تأکید خاصی بر ویروس‌ها و کِرم‌ها، را مورد بررسی قرار می‌دهد. این فصل با بررسی انواع مختلف نرم‌افزارهای موذی شروع شده و نگاه مفصل‌تری به ماهیت ویروس‌ها و کِرم‌ها می‌اندازد. بقیهٔ فصل نگاهی به روش‌های مقابله با این تهدیدها دارد. در انتها حملات توزیع شدهٔ انکار سرویس مورد بحث قرار می‌گیرد.

فصل ۱۱ دیوارهای آتش

یک روش استاندارد برای محافظت منابع کامپیوتری محلی از تهدیدهای خارجی، استفاده از دیوار آتش است. فصل ۱۱ اصول طراحی دیوار آتش را مورد بحث قرار داده و به تکنیک‌های معینی اشاره می‌کند. این فصل همچنین مقولهٔ مرتبط سیستم‌های معتمد را پوشش می‌دهد.

فصل ۹

مهاجمین

کی از مسائل امنیتی مهم در سیستم‌های شبکه‌ای، تعرض خصمانه و یا حداقل ناخواستهٔ کاربران و نرم‌افزارهاست. تعرض کاربر می‌تواند بصورت اتصال غیرمجاز به ماشین، و یا در مورد یک کاربر معتبرکسب امتیازات یا انجام عملیاتی فراتر از آنچه برای او مجاز است، باشد. تعرض نرم‌افزاری می‌تواند بشکل یک ویروس، کِرم و یا اسب تروا ظاهر شود.

تمام این حملات مرتبط با امنیت شبکه‌اند زیرا ورود به سیستم می‌تواند از طریق شبکه صورت پذیرد. ولی در عین حال این حملات منحصراً مبتنی بر شبکه نمی‌باشند. یک کاربر با امکان دسترسی به یک ترمینال محلی ممکن است بدون استفاده از شبکهٔ میانی مبادرت به تعرض نماید. یک ویروس یا یک اسب تروا ممکن است نه از طریق شبکه بلکه از طریق یک دیسکت وارد سیستم شود. تنها کِرم یک پدیدهٔ کاملاً شبکه‌ای است. بنابراین تعرض به سیستم بحثی است که هم مورد توجه امنیت شبکه و هم مورد توجه امنیت کامپیوتر است.

نظر به اینکه این کتاب بر امنیت شبکه متمرکز شده است، ما نسبت به تجزیه و تحلیل مفصّل تک‌ها و پاتک‌های مرتبط با ورود غیرمجاز به سیستم تلاش نخواهیم کرد. در عوض در این بخش مروری کلی بر این مشکلات خواهیم داشت.

این فصل به موضوع تعرض‌کنندگان و یا مهاجمین می‌پردازد. در ابتدا ماهیت حمله را بررسی نموده و سپس نگاهی به استراتژی‌های مربوط به جلوگیری از حمله و در صورت شکست در جلوگیری، استراتژی‌های تشخیص حمله می‌پردازیم. در قسمت بعد مقوله‌های مربوط با مدیریت کلمهٔ عبور را بررسی می‌کنیم.

۹-۱ مهاجمین (INTRUDERS)

یکی از دو تهدید عمدهٔ امنیت سیستم، مهاجمین هستند (دیگری ویروس‌ها می‌باشند)، که معمولاً از آنها با نام هَکر (hacker) و یا کِرَکِر (cracker) یاد می‌شود. در یکی از مطالعات مهم مربوط به مهاجمین، آندرسن [ANDE80] سه دسته از مهاجمین را شناسائی نموده است:

- **نقاب‌دار(Masquerader)** : فردی است که مجاز به استفاده از کامپیوتر نیست ولی از کنترل‌های دست‌یابی به سیستم عبور کرده و اشتراک یک کاربر قانونی را مورد سوء استفاده قرار می‌دهد.
- **سوءاستفاده‌کننده (Misfeasor)**: یک کاربر قانونی است که به دیتا، برنامه‌ها و یا منابعی دست می‌یازد که قانوناً اجازهٔ دست‌یابی به آنها را ندارد. این فرد ممکن است مجاز به دست‌یابی به آنها باشد ولی از آنها بطور غیرقانونی در جهت خواسته‌های خود استفاده می‌نماید.
- **کاربر خُفیه (Clandestine User)**: فردی است که کنترل سوپروایزری سیستم را بدست گرفته و از این کنترل برای فرار از شناخته شدن و یا خنثی کردن عملیات شناسائی استفاده نماید.

مهاجم نقاب‌دار به احتمال زیاد یک فرد غیرخودی است. مهاجم سوءاستفاده‌کننده معمولاً یک فـــرد خودی است و مهاجم خفیه می‌تواند فردی خودی و یا غیرخودی باشد.

حملات مهاجمین می‌تواند از مرز بی‌خطر تا مرز خطرناک متفاوت باشد. در مرز بی‌خطر، افراد بسیاری وجود دارند که بدون قصد سوء به کاوش در اینترنت کنجکاو بوده و می‌خواهند بدانند چه چیز تازه‌ای در اینترنت وجود دارد. در مرز خطرناک افراد معدودی هستند که تلاش می‌کنند تا به داده‌های مهم دست یابند، داده‌ها را بصورت غیرمجاز تغییر داده و یا سیستم را از کار بیندازند.

خطر مهاجمین بنحو قابل توجهی برای عموم روشن شده و شاید واقعهٔ "Wily Hacker" در سالهای ۱۹۸۷-۱۹۸۶ میلادی که بتوسط Cliff Stoll [STOL88,89] ثبت شده است علت عمدهٔ آن باشد. در سال ۱۹۹۰ میلادی، در آمریکا یک نمایش قدرت بر علیه هَکِرهای غیرمجاز صورت پذیرفت که طی آن عده‌ای دستگیر و عده‌ای متهم شدند. یک محاکمهٔ جنجالی برپا گردید، تعدادی محکوم شده و مقادیر زیادی از تجهیزات کامپیوتری و دیتا ضبط گردید [STER92]. بسیاری از مردم تصور کردند که مساله حل شده و مشکل مهار گردیده است.

اما در حقیقت، مساله تحت کنترل قرار نگرفته است. بعنوان مثال گروهی در لابراتوارهای Bell گزارش داده‌اند که حملات دائم و مکرری از طریق اینترنت و بتوسط منابع مختلف در طول زمان روی سایت‌های کامپیوتری آنها صورت پذیرفته است [BELL92, BELL93]. در زمان ارائهٔ این گزارش، گروه فوق شاهد موارد زیر بودند:

- تلاش برای کپی کردن فایل کلمات عبور (بعداً دربارهٔ آن بحث خواهد شد)، بیشتر از هر دو روز یکبار.
- تقاضاهای مشکوک برای فراخوانی از دور (RPC = Remote Procedure Call) به میزان بیش از هر هفته یکبار.
- تلاش برای اتصال به ماشین‌های «طعمه» که وجود خارجی ندارند حداقل هر دو هفته یکبار.

مهاجمین بی‌خطر را ممکن است تحمل نمود ولی باید توجه داشت که چون آنها منابع را بکار می‌گیرند می‌توانند عملیات کاربران قانونی را با کندی مواجه سازند. البته راهی وجود ندارد که بتوان پیش‌بینی کرد آیا یک مهاجم بی‌خطر و یا خطرناک است. در نتیجه حتی برای سیستم‌هائی که مدارک حساس بخصوصی را نگاه نمی‌دارند، انگیزه‌ای برای کنترل این مساله وجود دارد.

مثالی که بطور آشکار خطر چنین تهاجمی را نشان می‌دهد در دانشگاه A&M تکزاس رخ داد [SAFF93]. در ماه اوت ۱۹۹۲ میلادی، مرکز کامپیوتر آن دانشگاه متوجه شد که یکی از دستگاه‌های آن برای حمله به کامپیوترها در نقطه‌ای دیگر از طریق اینترنت بکار گرفته شده است. با کنترل عملیات، پرسنل مرکز کامپیوتر دریافتند که پای چندین مهاجم غیرخودی در کار است که برنامه‌هائی با هدف کشف کلمات عبور را روی تعدادی از کامپیوترها اجرا می‌کنند (سایت شامل ۱۲,۰۰۰ دستگاه کامپیوتری متصل بهم بود). مرکز کامپیوتر ماشین‌های آلوده را جدا کرده، حفره‌های امنیتی را بسته و عملیات نرمال را از سرگرفت. چند روز بعد، یکی از مدیران سیستم محلی دریافت که تهاجم از سر گرفته شده است. بعداً روشن شد که حمله بسیار پیچیده‌تر از آنست که در ابتدا تصور می‌شد. فایل‌هائی بدست آمدکه حاوی صدها کلمهٔ عبورکشف شده بودند که بعضی از آنها متعلق به سِرورهای اصلی و مثلاً امن بود. علاوه بر آن، یکی از ماشین‌های محلی بعنوان تابلوی اعلانات یک هَکِر مورد استفاده قرار گرفته بود که از آن هَکِرها برای ارتباط با هم و بحث در مورد تکنیک‌ها و پیشرفت کار استفاده می‌کردند.

تحلیل این حمله نشان داد که در واقع دو سطح از هَکِرها در این امر دخالت داشتند. هَکِرهای سطح بالا، کاربران پیچیده‌ای بودند که اطلاعات کافی از تکنولوژی داشتند و هَکِرهای سطح پائین «سربازان پیاده‌ای» بودند که صرفاً از برنامه‌های

عبور از سیستم استفاده کرده و نسبت به نحوهٔ عملکرد این برنامه‌ها اطلاعاتی نداشتند. این کار گروهی، دو سلاح عمده در زرادخانهٔ مهاجمین را با هم ترکیب کرده بود: اطلاعات پیچیده‌ای از اینکه چطور می‌توان مبادرت به هجوم کرد، و تمایل به صرف ساعات فراوان برای پیداکردن نقاط ضعف ورود به سیستم.

یکی از نتایج مرتبط با آشنائی فزاینده با مشکل تهاجم، ایجاد تعدادی از تیم های اورژانس کامپیوتری CERT (Computer Emergency Response Team) بوده است. این تیم‌ها، اطلاعات مربوط به نقاط آسیب‌پذیر سیستم‌ها را جمع کرده و آن را برای مدیران سیستم‌ها ارسال می‌کنند. متاسفانه هَکرها هم می‌توانند به این گزارشات CERT دسترسی یابند. در واقعهٔ A&M تکزاس، تحلیل‌های بعدی نشان داد که هَکرها برنامه‌هائی را ساخته بودند که با کمک آنها هر نقطهٔ ضعفی که بتوسط CERT گزارش شده بود مورد تجاوز قرار می‌گرفت. اگر حتی یک ماشین به توصیه‌های فوری CERT عمل ننموده بود، در مقابل این حملات آسیب‌پذیر باقی مانده بود.

علاوه بر برنامه‌های مربوط به دسترسی به کلمات عبور، مهاجمین سعی کردند تا نرم‌افزار ورود به سیستم را طوری دستکاری نمایند که بتوانند کلمه‌های عبور کاربرانی را که به آن سیستم متصل‌اند کشف کنند. این امر آنان را قادر ساخت تا مجموعهٔ عظیمی از کلمات عبور هَک شده را تهیه نموده و آن را در تابلوی اعلاناتی که خود در یکی از ماشین‌های قربانی ایجاد کرده بودند نصب کنند.

در این بخش به تکنیک‌های استفاده شده برای تهاجم نظر خواهیم کرد. آنگاه روش‌های تشخیص تهاجم را عنوان می‌کنیم. بالاخره به روش‌های مرتبط با کلمات عبور در جلوگیری از تهاجم توجه خواهیم نمود.

تکنیک‌های تهاجم

هدف مهاجم، کسب دست‌یابی به سیستم و یا افزایش محدودهٔ اختیارات وی در دست‌یابی به سیستم است. این امر نیاز به کسب اطلاعاتی از طرف مهاجم دارد که معمولاً حفاظت شده‌اند. در بیشتر موارد، این اطلاعات در کلمهٔ عبور کاربر خلاصه می‌شوند. با اطلاع از کلمهٔ عبور یک کاربر قانونی، یک مهاجم می‌تواند وارد سیستم شده و از تمام امتیازات مربوط به کاربر قانونی استفاده نماید.

معمولاً، یک سیستم بایستی فایلی را که شامل کلمات عبور تمام کاربران مجاز است در داخل خود نگهداری کند. اگر چنین فایلی بدون حفاظت در سیستم ذخیره شده باشد، آنگاه دسترسی به آن و پیدا کردن کلمات عبور برای هَکرها آسان خواهد بود. فایل کلمات عبور می‌تواند به دو طریق محافظت شود:

- **تابع یک- طرفه**: سیستم تنها یک فرم رمزشده از کلمهٔ عبور کاربر را نگهداری می‌کند. وقتی کاربر کلمهٔ عبوری را به سیستم عرضه می‌کند، سیستم این کلمهٔ عبور را به رمز درآورده و آن را با فرم ذخیره شدهٔ آن مقایسه می‌کند. در عمل، سیستم معمولاً یک تبدیل یک طرفه (برگشت‌ناپذیر) را انجام داده که درآن از کلمهٔ عبور برای تولید یک کلید رمزنگاری استفاده شده و یک خروجی با طول ثابت تولید می‌شود.
- **کنترل دست‌یابی**: دست‌یابی به فایل کلمات عبور، محدود به یک یا چند مشترک است.

اگر یکی از این دو روش مقابله با تهاجم و یا هردوی آنها در سیستم مستقر باشند، تلاش قابل توجهی لازم است تا یک مهاجم، هرچند قوی، بتواند کلمات عبور را پیدا کند. بر اساس یک بررسی انجام شده که شامل گفتگو با تعدادی هَکر کلمات عبور نیز بوده است [ALVA90]، تکنیک‌های پیداکردن کلمات عبور چنین‌اند:

۱- کلمات عبور پیش فرض مربوط به مشترکین که معمولاً به همراه سیستم عرضه می‌شوند، امتحان شود. بسیاری از مدیران سیستم، زحمت تغییر این کلمات عبور را بخود نمی‌دهند.
۲- تمام کلمات عبور کوتاه ممکن (بین ۱ تا ۳ حرف) امتحان شود.
۳- کلمات موجود در کتاب لغت حالت فعال سیستم و یا لیستی از کلمات عبور محتمل امتحان شود. نمونه‌هائی از کلمات عبور محتمل را می‌توان در تابلوی اعلانات هَکِرها پیدا کرد.
۴- اطلاعات مربوط به کاربران جمع‌آوری شود. این اطلاعات شامل نام کامل آنها، نام همسر و فرزندان آنها، نام عکس‌های موجود در دفتر آنها و نام کتاب‌های مورد علاقه و یا موجود در محل کار آنها که سرگرمی فرد محسوب می‌شوند، می‌باشد.
۵- شمارهٔ تلفن کاربر، شمارهٔ شناسنامه، کُد ملی، شمارهٔ اطاق وی و اطاق‌هائی که بیشتر با آنها مرتبط است امتحان شود.
۶- تمام شماره‌های قانونی اتومبیل‌ها در شهر محل سکونت کاربر امتحان شود.
۷- از یک اسب تروا برای دور زدن محدودیت‌های دست‌یابی به سیستم استفاده شود.
۸- روی خط ارتباطی سیستم مورد نظر با کاربران دور، شنود گذاشته شود.

شش روش اول، راه‌های مختلف حدس‌زدن کلمهٔ عبور است. اگر یک مهاجم قرار باشد تا با تلاش‌های مداوم و حدس‌زدن‌های مکرر به یک سیستم راه یابد، تلاش او تلاش خسته‌کننده‌ای خواهد بود که بسهولت می‌تواند تشخیص داده شود. بعنوان مثال، یک سیستم می‌تواند بسادگی پس از هر سه بار تلاش ناموفق در وصل شدن به سیستم، تلاش بعدی را انکار نماید و بنابراین مهاجم مجبور خواهد بود که پس از این مدت دوباره به سیستم متصل گردد تا تلاش خود را از سر بگیرد. تحت چنین شرایطی، امتحان کردن بیش از یک مشت کلمهٔ عبور عملی نخواهد بود. ولی احتمال اینکه یک مهاجم زیرک چنین روش خامی را انتخاب کند کم است. بعنوان مثال اگر مهاجم بتواند با سطح پائینی از امتیازات به یک فایل رمز شدهٔ کلمات عبور دست یابد، استراتژی او این خواهد بود که فایل را گرفته و در فرصت کافی به کشف رمز کلمات عبور بپردازد تا کلمهٔ عبوری با امتیازات بیشتر دست‌یابی را کشف کند.

حملات حدسی در جائی ممکن و حتی خیلی مؤثر خواهد بودکه بتوان حدس‌های زیادی را بصورت اتوماتیک به مرحلهٔ اجرا گذاشت و هر حدس را امتحان کرد بدون اینکه عملیات حدس زدن بتوسط مسئولین شبکه قابل تشخیص باشد. در بخش‌های بعدی این فصل در مورد خنثی‌سازی حملات حدسی صحبت خواهیم کرد.

روش هفتم در لیست بالا، یعنی استفاده از اسب تروا، کاری است که مقابله با آن بسیار مشکل است. مثالی از یک برنامه که مرحلهٔ کنترل دست‌یابی را دور می‌زند در [ALVA90] ذکر شده است. در این مثال، یک کاربر دارای سطح پائین دست‌یابی یک بازی کامپیوتری تهیه کرده و اپراتور سیستم را به استفاده از آن در اوقات فراغت وسوسه نمود. برنامه ظاهراً یک بازی کامپیوتری بود ولی در خفا شامل کُدی بود که فایل کلمات عبور که رمزشده نبوده ولی از نظر دست‌یابی حفاظت شده بود را به داخل فایل کاربر کپی می‌نمود. چون بازی در سطح بالای دست‌یابی مربوط به اپراتور اجرا می‌گردید، قادر بود تا به فایل کلمات عبور دست یابد.

حملهٔ هشتم ذکرشده در لیست بالا، یعنی شنود خط، در مقولهٔ حفاظت فیزیکی است. با این حمله می‌توان از طریق تکنیک‌های رمزنگاری پیوند، که در بخش رمزنگاری به آن اشاره شد، مقابله کرد.

سایر تکنیک‌های تهاجم نیازی به فراگیری کلمهٔ عبور ندارند. مهاجمین می‌توانند با سوء استفاده از حملاتی نظیر سرریزکردن حافظهٔ موقت روی برنامه‌ای که با امتیازات خاصی در حال اجراست، به سیستم دست یابند. ارتقاء امتیازات نیز می‌تواند بهمین روش انجام شود.

حال به بررسی دو روش مهم مقابله با این تهاجمات، یعنی تشخیص و جلوگیری، می‌پردازیم. تشخیص، مرتبط با آگاه شدن از حمله چه قبل و چه بعد از آن است. جلوگیری، یک چالش امنیتی و یک جنگ دائمی در همهٔ زمان‌هاست. مشکل از این حقیقت سرچشمه می‌گیرد که مدافع بایستی همهٔ حملات ممکن را دفع کند، در حالی که مهاجم آزاد است تا ضعیف‌ترین حلقه در زنجیرهٔ دفاعی را پیدا کرده و از آنجا حمله نماید.

۹-۲ تشخیص تهاجم (INTRUSION DETECTION)

بهترین سیستم‌های جلوگیری از تهاجم، ناچاراً روزی شکست خواهند خورد. خط دفاعی دوم سیستم در مقابل تهاجم، تشخیص تهاجم است و این مساله در سال‌های اخیر محور بسیاری از تحقیقات بوده است. ملاحظات متعددی محرک اینگونه بررسی‌ها بوده‌اند که از آن جمله‌اند:

۱- اگر تهاجم باندازهٔ کافی سریع تشخیص داده شود، مهاجم را می‌توان شناسائی و قبل از اینکه صدمه‌ای به سیستم وارد شده و یا داده‌هائی مورد تعرض قرار گیرند او را از سیستم بیرون راند. تشخیص تهاجم حتی اگر باندازه کافی سروقت انجام نشود تا بتوان مهاجم را قبل از هرگونه عملی از سیستم خارج کرد، بازهم هر چقدر زودتر صورت پذیرد میزان صدمات وارد به سیستم کمتر بوده و بازیابی سیستم سریعتر انجام خواهد پذیرفت.

۲- یک سیستم تشخیص تهاجم کارآمد، می‌تواند به عنوان یک سیستم بازدارنده عمل کرده و از هجمه‌ها جلوگیری کند.

۳- تشخیص تهاجم باعث می‌شود که بتوان مجموعه‌ای از اطلاعات مربوط به تکنیک‌های تهاجم را جمع‌آوری کرد که خود می‌تواند تسهیلات جلوگیری از تهاجم را توسعه بخشد.

تشخیص تهاجم بر این فرض قرار گرفته است که رفتار یک مهاجم با رفتار یک کاربر قانونی بنحوی متفاوت خواهد بود که می‌توان آن را اندازه‌گیری کرد. البته نمی‌توان انتظار داشت که یک وجه تمایز کاملاً مشخص و مرزبندی شده بین حملهٔ یک مهاجم و استفادهٔ معمول یک کاربر قانونی از منابع مجاز، وجود داشته باشد. بلکه باید انتظار داشته باشیم که بخشی از عملیات این دو یکسان و غیرقابل تشخیص باشند.

شکل ۹-۱ بطور خیلی کلی ماهیت وظیفه‌ای که بعهدهٔ یک طراح سیستم تشخیص تهاجم است را نشان می‌دهد. اگرچه رفتار نوعی یک مهاجم با رفتار معمول یک کاربر معتبر متفاوت است ولی این رفتارها نقاط مشترکی هم دارند. به همین دلیل یک تعبیر نسبتاً وسیع از رفتار تهاجمی که باعث بدام انداختن مهاجمین زیادتری گردد، خواه ناخواه منجر به متهم نمودن عده‌ای از کاربران مجاز، بعنوان مهاجم نیز خواهد گردید. از سوی دیگر کوشش برای مبرا نمودن همهٔ کاربران مجاز از اینکه بعنوان مهاجم شناخته شوند، نیاز به تعاریف سفت و سختی از رفتار تهاجمی داشته که در نتیجه آن برخی مهاجمین نیز از شناخته شدن مصون خواهند ماند. بنابراین عمل تشخیص تهاجم یک مصالحهٔ هنری بین دو مقولهٔ فوق‌الذکر است.

در بررسی آندرسن [ANDE80]، چنین فرض شده است که می‌توان با اطمینان معقول، بین یک مهاجم نقاب‌دار و یک کاربر قانونی تفاوت قایل شد. رفتار یک کاربر قانونی را می‌توان با مشاهدهٔ تاریخچهٔ رفتاری او در گذشته جمع‌آوری کرد و هر تغییر قابل ملاحظه در نحوهٔ رفتار او را می‌توان تشخیص داد. آندرسن بیان می‌کند که عمل تشخیص یک رفتار مشکوک (کاربر قانونی که رفتاری غیر قانونی دارد)، سخت‌تر از تشخیص تهاجم است زیرا فاصلهٔ بین رفتار نرمال و رفتار غیرنرمال یک کاربر قانونی ممکن است بسیار کم باشد. آندرسن نتیجه گرفت که این نوع تخطّی صرفاً از طریق جستجوی رفتارهای نامعقول،

غیر قابل تشخیص خواهد بود. با وجود این، رفتارهای مشکوک را می‌توان با تعریف هوشمندانهٔ دسته‌ای از عملیات که استفادهٔ غیرمعتبر از سیستم را مشخص می‌کنند تشخیص داد. بالاخره، تشخیص کاربر خفیه کاری فراتر از حوزهٔ روش‌های صرفاً اتوماتیک است. این مشاهدات که در سال ۱۹۸۰ میلادی مورد توجه بودند، امروز نیز همچنان معتبرند.

[PORR92] روش‌های زیر برای تشخیص تهاجم را تعریف می‌کند:

۱- **تشخیص آماری ناهنجاری**: شامل جمع‌آوری اطلاعات مربوط به کاربران قانونی در طول زمان است.آنگاه تست‌های آماری به رفتار یک کاربر مشکوک اعمال شده تا با اطمینان نسبتاً بالا تعیین شود که این رفتار، رفتار یک کاربر قانونی نیست.

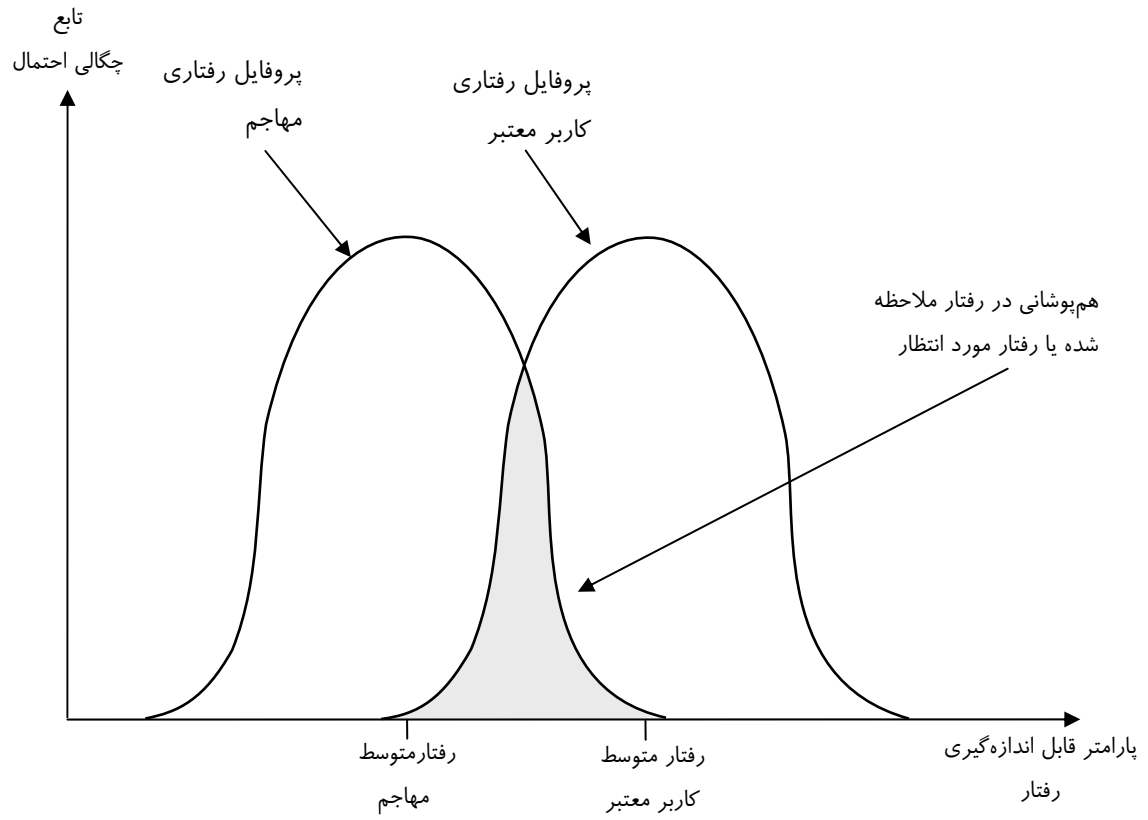
الف- تشخیص آستانه‌ای: این روش شامل تعریف آستانه‌هائی، مستقل از کاربر، برای تواتر وقوع اتفاقات مختلف است.

ب- تشخیص مبتنی بر پروفایل: یک پروفایل از فعالیت هرکاربر تهیه شده و برای تشخیص تغییرات رفتاری هر مشترک مجاز مورد استفاده قرار می‌گیرد.

۲- **تشخیص مبتنی بر قاعده**: شامل تلاش برای تعریف یک سری قواعد است که با استفاده از آن قواعد بتوان تصمیم گرفت که رفتار مشاهده شده، مربوط به یک مهاجم است.

الف- تشخیص موارد خلاف قاعده: قواعدی تهیه می‌شود که انحراف از رفتارهای نرمال را تشخیص دهد.

ب- تشخیص نفوذ: یک سیستم خبره است که به دنبال رفتارهای مشکوک می‌گردد.



شکل ۱-۹ پروفایل رفتاری مهاجمین و کاربران معتبر

در یک جمله، روش‌های آماری به دنبال تعریف رفتارهای هنجار و یا قابل انتظار هستند در حالی که روش‌های مبتنی بر قاعده، کوشش می‌کنند تا رفتارهای منظم را تعریف کنند.

بر اساس تعاریفی که قبلاً برای انواع مهاجمین داده شد، تشخیص آماری ناهنجاری در مورد مهاجمین نقاب‌دار، که به احتمال کم الگوی رفتاری کاربران مجاز را تقلید می‌کنند، مؤثر است. از سوی دیگر، این تکنیک‌ها ممکن است در مقابله با رفتارهای غیرمجاز مؤثر نباشند. برای چنین حملاتی، روش‌های مبتنی بر قاعده ممکن است قادر به شناسائی وقایع و تواتر وقوع آنها بوده و نفوذ بیگانه را مشخص سازند. در عمل یک سیستم ممکن است ترکیبی از هر دو روش را استفاده کرده تا بتواند در مقابل محدودهٔ وسیعی از حملات ایستادگی نماید.

سوابق ممیزی (Audit Records)

یکی از ابزارهای اصلی برای تشخیص تهاجم، سوابق ممیزی است. بعضی از سوابق فعالیت‌های جاری کاربران بایستی بعنوان ورودی یک سیستم تشخیص تهاجم نگهداری شوند. در این مورد از دو طرح استفاده می‌شود:

- **سوابق ممیزی بومی**: تقریباً تمام سیستم عامل‌های چندکاربره، دارای نرم‌افزار محاسبی می‌باشند که اطلاعات مربوط به فعالیت کاربران را جمع‌آوری می‌نماید. مزیت استفاده از این اطلاعات این است که هیچ نرم‌افزار جمع‌آوری دیگری مورد نیاز نیست. عیب آن نیز این است که سوابق بومی ممکن است اطلاعات مورد نیاز را نداشته و یا آن را به فرم مطلوب نداشته باشند.
- **سوابق ممیزی مخصوص تشخیص**: می‌توان یک تسهیلات جمع‌آوری اطلاعات را طوری برپا نمود که تنها سوابق مورد نیاز یک سیستم تشخیص تهاجم را جمع‌آوری نماید. یکی از مزایای چنین روشی این است که می‌توان آن را مستقل از سیستم تعریف کرد و به سیستم‌های مختلفی اعمال نمود. عیب آن میزان سربارهٔ اضافی، ناشی از داشتن دو بستهٔ نرم‌افزاری محاسباتی روی یک ماشین، است.

مثال خوبی از سوابق ممیزی مخصوص تشخیص، آن است که بتوسط Dorothy Denning [DENN87] خلق گردید. هر سابقهٔ ممیزی شامل میدان‌های زیر است:

- **عامل**: شروع‌کنندگان عملیات. یک عامل نوعاً کاربری در یک ترمینال است، ولی ممکن است پردازشی باشد که بجای یک کاربر و یا گروهی از کاربران عمل کند. تمام فعالیت‌ها از طریق فرمان‌هائی انجام می‌شود که از سوی عامل صادر شده است. عامل‌ها می‌توانند در گروه‌های مختلف با امتیازات دست‌یابی متفاوت طبقه‌بندی شده و این طبقات ممکن است هم‌پوشانی هم داشته باشند.
- **عمل**: عملیاتی است که بتوسط عامل روی موضوع انجام می‌شود. مثال‌هائی از این دست، ورود به سیستم، خواندن، عملیات I /O و غیره است.
- **موضوع**: چیزهائی که عمل روی آنها صورت می‌پذیرد. مثال‌های این مورد مانند فایل‌ها، برنامه‌ها، پیام‌ها، رکوردها، ترمینال‌ها، چاپگرها و یا ساختارهای ایجاد شده بتوسط کاربر یا برنامه است. وقتی یک عامل در معرض دریافت عملی، مانند دریافت e-mail، قرار می‌گیرد آنگاه همان عامل مبدل به یک موضوع می‌گردد. موضوع‌ها می‌توانند بر حسب نوعشان دسته‌بندی شوند. دسته‌بندی موضوعات ممکن است وابسته به شکل و یا محیط باشد. مثلاً عملیات روی یک پایگاه داده ممکن است براساس تأثیر روی کل پایگاه و یا از دید یک فایل آن پایگاه ممیزی گردد.

- **شرایط استثنائی**: در صورت وجود استثناء، آن را مشخص می‌سازد.
- **استفاده از منابع**: یک لیست کمّی است که مشخص می‌کند هر عنصر به چه میزان از منابع استفاده کرده است (مثلاً تعداد خطوطی که چاپ شده و یا نمایش داده شده است، تعداد رکوردهائی که خوانده و یا نوشته شده است، زمان پردازش، واحدهای I /O استفاده شده، زمان یک اجلاس).
- **زمان‌سنج**: یک برچسب زمانی یکتا که تاریخ و زمان وقوع عمل را نشان می‌دهد.

اکثر عملیات یک کاربر از تعدادی عمل ابتدائی تشکیل شده است. برای مثال، کپی کردن یک فایل شامل انجام فرمان کپی کاربر است که خود شامل اعمالی مانند کنترل اعتبار دست‌یابی به فایل، تنظیم کپی، خواندن از یک فایل و نوشتن فایل در جای دیگر است. فرمان زیر را در نظر بگیرید

COPY GAME.EXE TO <Library>GAME.EXE

که بتوسط آقای Smith برای کپی کردن یک فایل اجرائی GAME از فهرست جاری به فهرست دیگری بنام <Library> صادر شده است. سوابق ممیزی زیر ممکن است تولید شوند:

Smith	execute	<Library>COPY.EXE	0	CPU = 00002	11058721678

Smith	read	<Smith>GAME.EXE	0	RECORDS = 0	11058721679

Smith	execute	<Library>COPY.EXE	write-viol	RECORDS = 0	11058721680

در این مورد، عمل کپی ممکن است نادیده گرفته شود (abort)، زیرا Smith اجازه نوشتن فایلی در فهرست <Library> را نداشته باشد.

تجزیه عمل کاربر به عملیات ابتدائی‌تر دارای سه مزیت است:

۱- چون موضوعات، واحدهای قابل حفاظت در یک سیستم‌اند، استفاده از عملیات ابتدائی باعث می‌گردد تا ممیز بتواند تمام رفتارهائی که روی موضوع تأثیر می‌گذارند را ثبت کند. بنابراین سیستم می‌تواند تمام تلاش‌های ناموفق برای دست‌یابی را تشخیص داده (با ملاحظهٔ وضع غیرعادی موارد برگشتی) و همچنین تلاش‌های موفق را، با ملاحظهٔ وضع غیرعادی در مجموعهٔ موضوعاتی که عامل به آنها دسترسی دارد، پیدا کند.
۲- سوابق ممیزی مربوط به یک موضوع منفرد و یک عمل منفرد، مدل را ساده نموده و ساخت آن را آسان می‌سازند.
۳- بعلت ساختار ساده و یکنواخت سوابق ممیزی مخصوص تشخیص، کسب اطلاعات مربوط به آن و یا حداقل بخشی از این اطلاعات می‌تواند به سهولت، با انتقال سوابق ممیزی بومی به آن انجام شود.

تشخیص آماری ناهنجاری (Statistical Anomaly Detection)

همانطور که قبلاً بیان گردید، تکنیک‌های تشخیص آماری ناهنجاری در دو طبقهٔ وسیع جای می‌گیرند: سیستم‌های تشخیص آستانه‌ای و سیستم‌های مبتنی بر پروفایل. تشخیص آستانه‌ای شامل شمارش تعداد وقوع یک پیشامد بخصوص در محدودهٔ مشخصی از زمان است. اگر شمارش از تعداد معقولی که مورد انتظار است فراتر رود، آنگاه می‌توان فرض کرد که تهاجمی رخ داده است.

تحلیل آستانه‌ای به تنهائی یک تشخیص‌دهندهٔ مبتدی و غیرمؤثر، حتی در مورد حملات با پیچیدگی کم، است. هم اندازهٔ آستانه و هم فاصلهٔ زمانی بایستی تعیین شوند. بعلت تنوع رفتار کاربران، احتمال دارد که این آستانه‌ها نتایج مثبت اشتباه و یا نتایج منفی اشتباه قابل ملاحظه‌ای را ایجاد کنند. با وجود این، تشخیص‌دهنده‌های سادهٔ آستانه‌ای ممکن است در معیت تکنیک‌های پیچیدهٔ دیگر مفید واقع شوند.

تشخیص ناهنجاری مبتنی بر پروفایل، بر تعیین رفتارهای سابق تک‌تک کاربران و یا گروه کاربران تکیه کرده و سعی می‌کند انحراف از این نوع رفتار را نشان دهد. یک پروفایل ممکن است شامل مجموعه‌ای از پارامترها باشد تا تنها انحراف یک پارامتر از حالت نرمال آن، منجر به اعلام یک تهاجم نگردد.

تشخیص ناهنجاری مبتنی بر پروفایل بر اساس تحلیل سوابق ممیزی قرار دارد. سوابق ممیزی به دو طریق، ورودی تابع تشخیص تهاجم را تشکیل می‌دهند. اولاً، طراح بایستی نسبت به تعیین یک سری از معیارهای کمّی که می‌تواند رفتار کاربر را ارزیابی کند تصمیم بگیرد. تحلیل سوابق ممیزی در یک دوره از زمان می‌تواند برای تعیین پروفایل فعالیت یک کاربر معمولی مورد استفاده واقع شود. بنابراین سوابق ممیزی برای تعریف رفتار نوعی یک کاربر بکار می‌روند. ثانیاً، سابقهٔ ممیزی عملیات جاری، ورودی سیستم برای تشخیص تهاجم را تشکیل می‌دهند. یعنی مدل تشخیص تهاجم، سوابق ممیزی عملیات جاری را تحلیل کرده تا انحراف آن از رفتار متوسط کاربر را تشخیص دهد.

مثال‌هائی از معیارهای تشخیص، که در تشخیص تهاجم مبتنی بر پروفایل مفید هستند، بقرار زیراند:

- **شمارنده**: یک عدد صحیح غیرمنفی که می‌تواند زیاد شده ولی نمی‌تواند کم شود مگر اینکه با یک فرمان مدیریتی صفر گردد. معمولاً شمارش یک پیشامد در طول دورهٔ زمانی مشخصی مورد توجه است. نمونه‌هائی از این دست شامل تعداد تلاش‌ها برای ورود به سیستم از طرف یک کاربر مشخص در طول یک ساعت، تعداد دفعات اجرای یک فرمان در زمان یک اجلاس، و یا تعداد تلاش‌های ناموفق برای ورود به سیستم از طریق کلمات عبور غلط ظرف یک دقیقه است.

- **پیمانه**: یک عدد صحیح غیرمنفی که می‌تواند هم زیاد و هم کم شود. یک پیمانه معمولاً برای اندازه گیری مقدار یک چیز بکار می‌رود. مثال‌های این مورد نظیر تعداد اتصالات منطقی به یک برنامهٔ کاربردی و یا تعداد پیام‌های خروجی است که برای پردازش در صف انتظار قرار گرفته‌اند.
- **زمان سنج**: زمان بین وقوع دو پیشامد یکسان را نشان می‌دهد. مثال امر، فاصله زمانی بین دو ورود به سیستم از سوی یک کاربر مشخص است.
- **بکارگیری منابع**: اندازهٔ کمّی منابعی که در طول زمان مشخصی بکار گرفته شده‌اند را نشان می‌دهد. نمونهٔ این موارد عبارت از تعداد صفحات چاپ شده در خلال یکبار اتصال به سیستم و یا زمان صرف شده برای اجرای یک برنامه است.

با در دست داشتن این معیارها، تست‌های مختلفی را می‌توان برای تعیین اینکه فعالیت‌های جاری در محدودهٔ قابل قبول هستند و یا خیر، انجام داد. [DENN87] به عوامل زیر که می‌توانند مؤثر واقع شوند اشاره می‌کند:

- میانگین و انحراف معیار
- چندمتغیری
- فرآیند مارکوف
- سری‌های زمانی
- عملیاتی

ساده‌ترین تست آماری، اندازه‌گیری **میانگین و انحراف معیار** یک پارامتر در طول یک دورهٔ زمانی است. این اندازه‌ها منعکس‌کنندهٔ رفتار متوسط و تغییرات حول و حوش آن است. استفاده از میانگین و انحراف معیار قابل اعمال به شمارنده‌ها، زمان‌سنج‌ها و منابع دیگر می‌باشند. اما این معیارها معمولاً به تنهائی برای تشخیص تهاجم کافی نیستند.

یک مدل **چندمتغیری** برمبنای همبستگی بین دو یا چند متغیر قرار دارد. رفتار مهاجم را با ملاحظهٔ همبستگی بین وقایع (مثل همبستگی زمان پردازش با استفاده از یک منبع، و یا همبستگی تعداد دفعات ورود به سیستم با زمان استفاده از سیستم در هر بار)، با اطمینان بیشتری می‌توان تحلیل نمود .

یک مدل **فرآیند مارکوف** برای تعیین احتمال عبور بین حالات مختلف بکار می‌رود. بعنوان مثال این مدل ممکن است عبور از یک فرمان به فرمان مشخص دیگر را جستجو نماید.

یک مدل **سری زمانی** روی فواصل زمانی تکیه کرده و به دنبال زنجیره‌ای از وقایع می‌گردد که فاصلهٔ بین آنها یا خیلی زیادتر و یا خیلی کمتر از حد معمول می‌باشد. تست‌های آماری مختلفی می‌توانند برای تعیین زمان‌بندی غیرنرمال مورد استفاده قرار گیرند.

بالاخره یک **مدل عملیاتی**، بجای تحلیل اتوماتیک سوابق ممیزی، بر مبنای قضاوت در مورد آنچه غیرعادی تلقی می‌شود قرار دارد. در این روش نوعاً محدوده‌های ثابتی تعریف شده و اگر عملیاتی خارج از این محدوده‌ها انجام شود، شک نسبت به تهاجم قوت می‌پذیرد. چنین روشی وقتی خوب کار می‌کند که رفتار تهاجمی را بتوان از روی یک سری رفتارهای مشاهده شده نتیجه‌گیری کرد. بعنوان مثال، اگر در طول زمان کوتاهی تلاش زیادی برای ورود به سیستم انجام شود، می‌توان آن را نوعی تهاجم تلقی کرد.

بعنوان مثالی از بکارگیری این معیارها و مدل‌های مختلف، جدول ۱-۹ معیارهای متنوعی که در سیستم تشخیص تهاجم (IDES) انستیتوی تحقیقاتی استانفورد (SRI) بکار گرفته و آزمایش شده است را نشان می‌دهد [DENN87,JAVI91,LUNT88].

مزیت عمدهٔ استفاده از پروفایل‌های آماری این است که نیازی به داشتن معلومات قبلی از خطاهای امنیتی نیست. برنامهٔ تشخیص دهندهٔ تهاجم می‌آموزد که چه رفتاری نرمال است و آنگاه بدنبال انحراف از این رفتار نرمال می‌گردد. این روش متکی بر مشخصه‌های سیستم و نقاط آسیب‌پذیر آن نیست. در نتیجه این تکنیک می‌تواند بسهولت در سیستم‌های دیگر نیز بکار گرفته شود.

جدول ۱-۹ معیارهائی که می‌توانند برای تشخیص تهاجم بکار گرفته شوند

معیار	مدل	نوع تهاجم تشخیص داده شده
فعالیت مربوط به ورود به سیستم و اجلاس‌ها		
دفعات ورود به سیستم در روز و زمان	میانگین و انحراف معیار	مهاجمین احتمال دارد که در ساعات غیر اداری وارد سیستم شوند.
دفعات ورود به سیستم از مکان های مختلف	میانگین و انحراف معیار	مهاجمین ممکن است از مکانی وارد سیستم شوند که یک کاربر خاص بندرت و یا هیچگاه از آنجا وارد نمیشود.
زمان گذشته از آخرین ورود به سیستم	عملیاتی	وارد شدن از یک حساب مسدود.
زمان هر اجلاس	میانگین و انحراف معیار	انحراف قابل ملاحظه از زمان معمول ممکن است نشان یک مهاجم نقاب‌دار باشد.
میزان دیتای انتقال‌یافته به مکانی دوردست	میانگین و انحراف معیار	انتقال مقدار زیادی از دیتا به نقطه‌ای دور می‌تواند به نشان نشت اطلاعات حیاتی باشد.
بکارگیری منابع در هر اجلاس	میانگین و انحراف معیار	بکارگیری زیاد پردازشگر و I/O میتواند بعلت حضور مهاجم باشد.
دادن کلمهٔ عبور اشتباه در هنگام ورود به سیستم	عملیاتی	تلاش برای ورود به سیستم با حدس زدن کلمهٔ کلمهٔ عبور.
شکست در ورود به سیستم از ترمینال های بخصوص	عملیاتی	تلاش برای ورود غیرمجاز به سیستم.
فعالیت مربوط به اجرای برنامه‌ها و فرمان‌ها		
دفعات اجرای یک برنامه یا فرمان	میانگین و انحراف معیار	می‌تواند از طرف یک مهاجم باشد که از فرمان‌های مختلف استفاده می‌کند و یا ناشی از نفوذ موفق یک کاربر قانونی است که به فرمان‌های سطح بالا دسترسی یافته است.
بکارگیری منابع برنامه	میانگین و انحراف معیار	یک اندازهٔ غیر معقول میتواند نماینده ورود یک ویروس یا اسب تروا باشد که آثار جنبی آن بکارگیری پردازشگر یا بخش I/O است.
اجرا نشدن برنامه	عملیاتی	می‌تواند به تشخیص تهاجمی از سوی یک فرد برای دسترسی به امتیازات بالاتر منجر گردد.
فعالیت مربوط به دست یابی به فایل‌ها		
دفعات خواندن، نوشتن، ایجاد و حذف	میانگین و انحراف معیار	مقادیر غیرمعقول تلاش برای خواندن و نوشتن از طرف کاربران می‌تواند بیانگر حملهٔ بالماسکه‌ای یا مرورگری باشد.
رکوردهای خوانده شده و یا نوشته شده	میانگین و انحراف معیار	اندازه‌های غیرمعقول می‌تواند نمایش تلاش برای بدست آوردن اطلاعات حساس باشد.
شمارش دفعات تلاش ناموفق برای خواندن، نوشتن، ایجاد و یا حذف	عملیاتی	می‌تواند منجر به کشف کاربرانی شود که مصرانه قصد دسترسی به فایل‌های غیرمجاز را دارند.

تشخیص مبتنی بر قاعده تهاجم (Rule-Based Intrusion Detection)

روش‌های مبتنی بر قاعده، تهاجم را با مشاهدهٔ پیشامدها در یک سیستم و اعمال یک سری قواعد به آنها ردیابی می‌کنند تا به این تصمیم دست یابند که این مجموعهٔ فعالیت‌ها مشکوک و یا غیرمشکوک است. در بیان خیلی کلی، تمام روش‌ها را می‌توان به تشخیص ناهنجاری و یا شناسائی نفوذ دسته‌بندی کرد، اگرچه این دو گروه دارای نقاط مشترکی نیز هستند.

تشخیص مبتنی بر قاعدهٔ ناهنجاری (Rule-based anomaly detection)، از نظر روش برخورد و قدرت شبیه تشخیص آماری ناهنجاری است. در روش مبتنی بر قاعده، سوابق تاریخی فایل‌ها تحلیل شده تا رفتارهای کاربر شناسائی گشته و بصورت خودکار قواعدی که نمایش‌دهندهٔ این رفتارهاست تولیدگردند. قواعد ممکن است نشان‌دهندهٔ رفتارهای گذشته کاربران، برنامه‌ها، اولویت‌ها، شیارهای زمانی، ترمینال‌ها و غیره باشند. آنگاه رفتارهای زمان حال مشاهده شده و هر عمل با یک سری قواعد تهیه شده مقایسه گشته تا معلوم شود آیا این عمل جاری با تاریخچهٔ مربوط به اعمال گذشته همخوانی دارد یا خیر.

همانند تشخیص آماری ناهنجاری، تشخیص مبتنی بر قاعدهٔ ناهنجاری نیاز به شناخت نقاط آسیب‌پذیر امنیتی یک سیستم ندارد. بلکه روش، مبتنی بر مشاهدهٔ رفتار گذشته بوده و در واقع فرض می‌کند که آینده هم مثل گذشته خواهد بود. برای اینکه این روش اثربخش باشد، به یک پایگاه داده نسبتاً حجیم از قواعد نیازمندیم. برای مثال، روشی که در [VACC89] توصیف شده است بین $۱۰^{۴}$ تا $۱۰^{۶}$ قاعده دارد.

شناسائی مبتنی بر قاعده نفوذ (Rule-based penetration identification)، روش بسیار متفاوتی را برای تشخیص تهاجم بکار می‌برد که بر مبنای تکنولوژی سیستم‌های خبره قرار دارد. مشخصهٔ کلیدی چنین سیستم‌هائی استفاده از قواعد برای شناسائی نفوذهای شناخته شده و یا نفوذهائی که از نقاط ضعف شناخته شده استفاده می‌کنند می‌باشد. همچنین می‌توان قواعدی را تعریف کرد که رفتارهای مشکوک را، حتی وقتی که در محدودهٔ قانونی رفتارهای تعیین شده کاربردها هستند، شناسائی نماید. قواعدی که در این سیستم‌ها بکار می‌روند، نوعاً منحصر به ماشین و سیستم عامل بخصوصی می‌باشند. همچنین، چنین قواعدی بجای این‌که بر اساس تحلیل خودکار سوابق تولید شوند بتوسط «خبرگان» تدوین می‌شوند. روش معمول کار این است که با مدیران سیستم و تحلیل‌گران امنیتی مذاکره شده تا بر اساس تجربیات آنان مجموعه‌ای از سناریوهای نفوذ و وقایع کلیدی که امنیت سیستم هدف را به مخاطره می‌اندازند جمع‌آوری گردد. روشن است که کارآئی این روش، بستگی به مهارت افراد مرتبط در جمع‌آوری قواعد خواهد داشت.

مثال ساده‌ای از نوع قواعدی که می‌تواند مورد استفاده قرار گیرد را می‌توان در NIDX، یکی از اولین سیستم‌هائی که از شواهد تاریخی برای نسبت دادن درجاتی از شک و شبهه به فعالیت‌ها استفاده کرده است، پیدا کرد [BAUE88]. مثال‌هائی از این شواهد تاریخی چنین‌اند:

۱- کاربران نبایستی فایل‌هائی که در پوشه‌های شخصی کاربران دیگر است را بخوانند.
۲- کاربران نبایستی در فایل‌های کاربران دیگر بنویسند.
۳- کاربرانی که پس از گذشت چندساعت مجدداً وارد کامپیوتر می‌شوند معمولاً به همان فایل‌هائی رجوع می‌کنندکه قبلاً به آنها مراجعه کرده‌اند.
۴- کاربران معمولاً تجهیزات مرتبط با دیسک‌ها را مستقیماً باز نمی‌کنند بلکه متکی به قابلیت‌های سطوح بالاتر سیستم عامل هستند.
۵- کاربران نبایستی بیش از یکبار به یک سیستم وارد شوند.
۶- کاربران از برنامه‌های سیستمی کپی تهیه نمی‌کنند.

روش شناسائی نفوذ در IDES بیانگر استراتژی دنبال شده است. سوابق ممیزی در زمان جمع‌آوری بررسی شده و با قواعد مبنا مقایسه می‌شوند. اگر بین این دو شباهتی وجود داشته باشد *نرخ شک* کاربر افزایش می‌یابد. اگر تعداد تطبیق مشاهدات با قواعد مبنا از یک آستانه فراتر رود، آنگاه شناسائی یک بیگانه گزارش می‌شود.

روش IDES مبتنی بر بررسی سوابق ممیزی است. یک نقطهٔ ضعف این طرح، نداشتن قابلیت انعطاف است. برای یک سناریوی نفوذ، ممکن است بتوان سوابق متنوعی را که هر یک از آنها با دیگری اختلاف جزئی داشته باشد معیار قضاوت قرار داد. مشکل این است که شاید نتوان این مقدار تنوع زیاد را در قواعد صریحی جمع‌آوری نمود. روش دیگر تهیهٔ یک مدل سطح بالاتری است که مستقل از سوابق ممیزی عمل کند. مثالی از این دست، مدل گذر از حالات USTAT است [ILGU93]. USTAT بجای استفاده از رفتارهای مشخص با جزئیات کامل که در مکانیسم ثبت سوابق UNIX دیده شده است از رفتارهای کلی استفاده می‌کند. USTAT روی یک سیستم عامل SunOS ساخته شده که سوابق ممیزی ۲۳۹ پیشامد را فراهم می‌سازد. از این تعداد تنها ۲۸ تا برای پردازش اولیه بکار می‌روند که منجر به ۱۰ عمل مختلف می‌شوند (جدول ۲-۹). تنها با استفاده از این اعمال و پارامترهائی که با هر عمل بکار گرفته می‌شود، یک دیاگرام حالت که فعالیت مشکوک را تعیین می‌کند ساخته می‌شود. چون تعدادی از پیشامدهای قابل ممیزی مختلف به تعداد کمتری عمل نگاشت می‌شوند، روش خلق قواعد ساده‌تر است. علاوه براین، مدل دیاگرام حالت پس از مشاهدهٔ رفتارهای تهاجمی جدید، به فرم ساده‌تری قابل جرح و تعدیل خواهد بود.

جدول ۲-۹ عملیات USTAT در برابر انواع پیشامدهای SunOS

USTAT Action	SunOS Event Type
Read	open_r, open_rc, open_rtc, open_rwc open_rwtc, open_rt, open_rw, open_rwt
Write	truncate, ftruncate, creat, open_rtc, open_rwc, open_rwtc, open_rt, open_rw, open_rwt, open_w, open_wt open_wc, open_wct
Create	mkdir, creat, open_rc, open_rtc, open_rwc, open_rwtc, open_wc, open_wtc, mknod
Delete	rmdir, unlink
Execute	exec, execve
Exit	exit
Modify_Owner	chown, fchown
Modify_Perm	chmod, fchmod
Rename	rename
Hardlink	link

خطای نرخ پایهٔ (Base-Rate Fallacy)

یک سیستم تشخیص تهاجم برای اینکه کاربرد عملی داشته باشد، بایستی بتواند درصد قابل توجهی از تهاجم‌ها را تشخیص داده و در عین حال نرخ آلارم‌های کاذب را در حد قابل قبولی پائین نگاه دارد. اگر تنها درصد متوسطی از تهاجم‌های واقعی شناسائی شوند، سیستم حس امنیتی کاذبی را ایجاد خواهد کرد. از سوی دیگر اگر سیستم در مواقعی که تهاجمی وجود ندارد هشدار دهد (آلارم کاذب)، آنگاه یا مدیران سیستم آلارم‌ها را جدی نخواهند گرفت و یا زمان زیادی در راه تحلیل آلارم‌های کاذب تلف خواهد شد.

متاسفانه، بعلت احتمالی بودن وقایع مرتبط با این مساله، بسیار مشکل است که بتوان نرخ بالای کشف تهاجمات را با نرخ کم آلارم‌های کاذب همراه کرد. بطور کلی اگر تعداد واقعی تهاجمات در مقایسه با تعداد دفعات استفادهٔ قانونی از سیستم کم باشد، آنگاه نرخ آلارم‌های کاذب بالا خواهد بود مگر اینکه تست‌ها بتوانند بصورت فوق‌العاده‌ای بین صحت و سقم وقایع تفاوت قائل شوند. یک بررسی از سیستم‌های تشخیص تهاجم که در [AXEL00] گزارش شده است، نشان می‌دهد که سیستم‌های جاری بر مشکل خطای نرخ پایهٔ فایق نشده‌اند. برای آگاهی مختصری از ریاضی این بحث به ضمیمهٔ ۹- الف مراجعه شود.

تشخیص توزیع شدهٔ تهاجم

تا زمان‌های اخیر، کار بر روی سیستم‌های تشخیص تهاجم منحصر به تجهیزات یک سیستم منفرد و متکی به خود بود. سازمان‌های کنونی نوعاً نیاز به یک دفاع کارآمد از مجموعهٔ توزیع شدهٔ میزبان‌هائی را دارند که بتوسط شبکه‌های LAN و یا اینترنت بهم متصل‌اند. اگرچه می‌توان برای هر میزبان، یک عملیات دفاعی تشخیص تهاجم را بطور جداگانه بوجود آورد ولی با هماهنگی و همکاری بین سیستم‌های تشخیص تهاجم منصوبه بر روی شبکه، می‌توان دفاع مؤثرتری را ایجاد کرد.

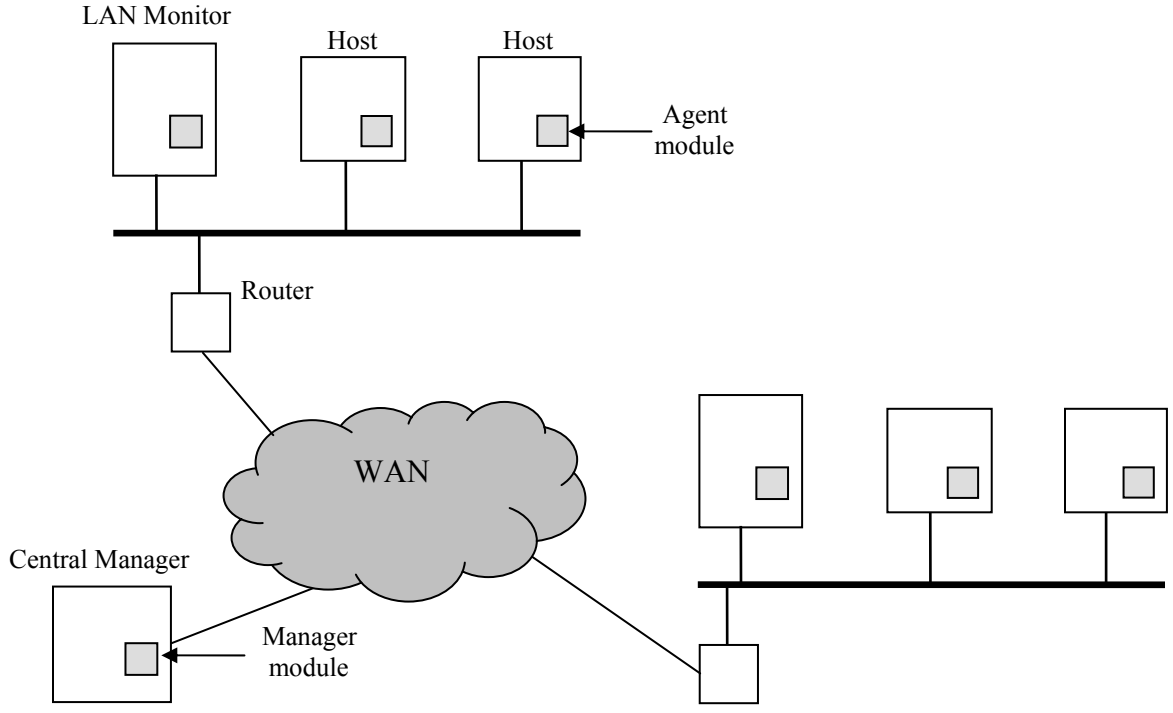
Porras مقوله‌های عمده زیر را در طراحی یک سیستم توزیع‌شدهٔ تشخیص تهاجم یادآوری می‌کند[PORR92]:

- یک سیستم توزیع‌شدهٔ تشخیص تهاجم ممکن است به استفاده از فرمت‌های مختلف سوابق ممیزی نیاز داشته باشد. در یک محیط نامتجانس، سیستم‌های مختلف از روش‌های جمع‌آوری سوابق بومی مختلفی استفاده کرده و اگراز سیستم تشخیص تهاجم استفاده نمایند، ممکن است فرمت‌های مختلفی از سوابق ممیزی امنیتی را بکار گیرند.
- یک و یا چند گره از شبکه بعنوان نقاط جمع‌آوری و تحلیل داده‌های مرتبط با سیستم‌های شبکه بکار خواهند رفت. بنابراین دیتای مربوط به سوابق ممیزی یا بصورت خام و یا بصورت خلاصه، بایستی در عرض شبکه منتقل گردد و لازم است که اصالت و محرمانگی این داده‌ها تضمین شود. اصالت داده‌ها از اینجهت مورد نیاز است که یک مهاجم نتواند فعالیت‌های خود را با تغییر دادن سوابق ممیزی انتقال یافته، در پشت نقابی پنهان نماید. محرمانگی داده‌ها نیز از اینجهت مهم است که اطلاعات ممیزی می‌توانند اطلاعات گرانقیمتی باشند.
- هم یک معماری متمرکز و هم یک معماری غیرمتمرکز می‌تواند برای این منظور بکار رود. در یک معماری متمرکز، تنها یک نقطهٔ مرکزی جمع‌آوری و تحلیل داده‌های ممیزی وجود دارد. این امر وظیفهٔ ارتباط دادن گزارشات ورودی به یکدیگر را تسهیل نموده ولی باعث ایجاد یک گلوگاه و یا یک نقطهٔ خرابی منفرد می‌شود. در یک معماری غیرمتمرکز بیش از یک مرکز جمع‌آوری و تحلیل داده‌ها وجود داشته که البته بایستی با هم هماهنگی نموده و اطلاعات را مبادله کنند.

مثال خوبی از یک سیستم توزیع‌شدهٔ تشخیص تهاجم، در دانشگاه کالیفرنیا در Davis طراحی شده است [HEBE92,SNAP91]. شکل ۲-۹ معماری کلی این طرح را نشان می‌دهدکه از سه مؤلفهٔ اصلی تشکیل می‌گردد:

- **مدول عامل میزبان**: یک مدول جمع‌آوری سوابق ممیزی که بصورت یک پردازش پشت پرده در یک سیستم پایش شده عمل می‌کند. هدف آن جمع‌آوری داده‌های مربوط به پیشامدهای مرتبط با امنیت در سیستم میزبان و انتقال آن به یک مدیر مرکزی است.
- **مدول عامل پایشگر LAN**: بهمان روش مدول عامل میزبان کار کرده بجز اینکه ترافیک LAN را تجزیه و تحلیل کرده و نتایج را به مدیر مرکزی گزارش می‌کند.
- **مدول مدیریت مرکزی**: گزارشات را از میزبان‌ها و پایشگر LAN گرفته، آنها را پردازش نموده، همبستگی آنها را جستجو کرده و تهاجم را تشخیص می‌دهد.

روش طوری طراحی شده است که مستقل از نوع سیستم عامل و یا روش جمع‌آوری سوابق ممیزی سیستم است. شکل ۳-۹ [SNAP91] روش کلی برخورد با مسأله را نشان می‌دهد. واحد عمل‌کننده هر سابقه‌ای را که بتوسط سیستم‌های ثبت سوابق بومی جمع‌آوری می‌شود می‌گیرد. فیلتر نشان داده شده طوری طراحی شده است که تنها سوابقی را که از نظر امنیتی دارای اهمیت هستند نگاه دارد. سوابق نگاهداری شده آنگاه طوری فرمتشان عوض می‌شود تا به فرم یک فرمت استاندارد که سوابق ممیزی میزبان (HAR) خوانده می‌شود درآیند. آنگاه یک مدول منطقی مبتنی بر الگو، سوابق را بمنظور کشف فعالیت‌های مشکوک تجزیه و تحلیل می‌کند. در پائین‌ترین سطح، واحد عمل‌کننده به دنبال پیشامدهائی می‌گردد که مستقل از هر پیشامد قبلی مورد توجه‌اند. شکست در دسترسی به یک فایل، تلاش برای دست‌یابی به فایل‌های سیستم، و تغییر



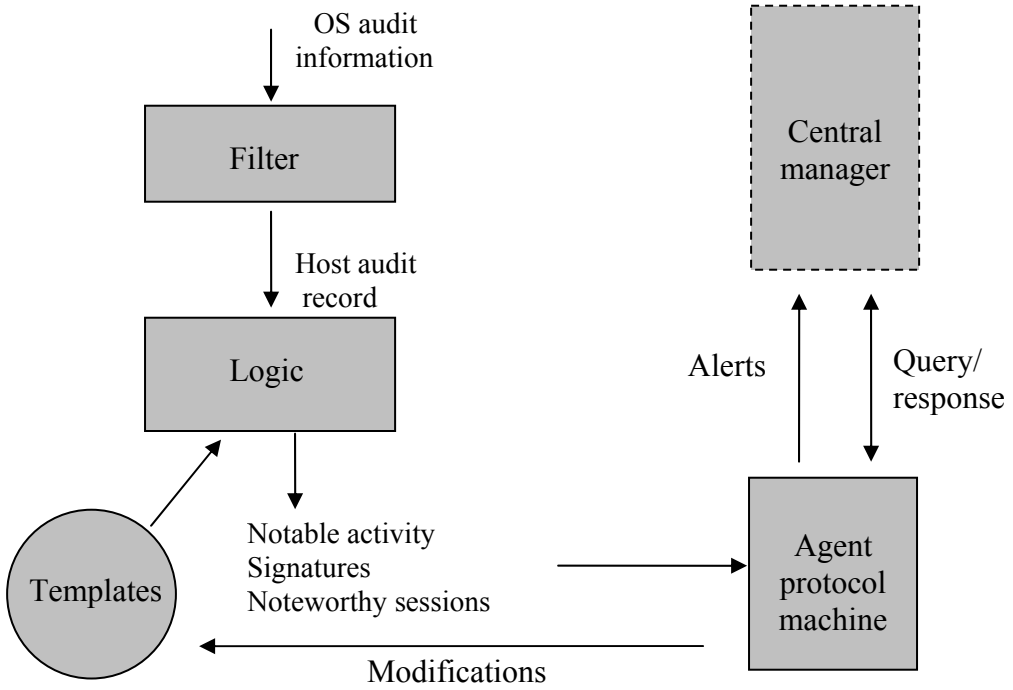
شکل ۲-۹ معماری برای تشخیص توزیع شدهٔ تهاجم

در کنترل دست‌یابی به یک فایل، مثال‌هائی از این نوع‌اند. در یک سطح بالاتر، واحد عمل‌کننده به دنبال زنجیره‌ای از پیشامدها مانند پترن حمله‌های شناخته شده (امضاءها) می‌گردد. بالاخره بر اساس پروفایل تاریخچهٔ رفتاری یک کاربر، رفتار مشکوک آن کاربر را مورد توجه قرار می‌دهد. در این مقوله، تعداد برنامه‌های اجراشده، تعداد فایل‌های مورد استفاده قرار گرفته و امثال آن مورد توجه خواهند بود.

وقتی فعالیت مشکوکی تشخیص داده می‌شود، یک هشدار برای مدیریت مرکزی ارسال می‌شود. مدیریت مرکزی شامل یک سیستم خبره بوده که می‌تواند از داده‌های دریافت شده، گمانی را استخراج نماید. مدیریت ممکن است از هر میزبان، نسخهٔ HAR او را درخواست نماید تا آنها را با یکدیگر مقایسه کند.

پایشگر LAN نیز اطلاعاتی را برای مدیریت مرکزی فراهم می‌سازد. این عامل، اتصالات میزبان‌ها به یکدیگر و سرویس‌های استفاده شده و حجم ترافیک را ثبت می‌کند. همچنین پیشامدهای قابل توجه از قبیل تغییر ناگهانی بار شبکه، استفاده از سرویس‌های مرتبط با امنیت، و فعالیت‌های شبکه مانند *rlogin* بتوسط همین عامل جستجو می‌شود.

معماری نشان داده شده در شکل‌های ۲-۹ و ۳-۹ کاملاً عام و قابل انعطاف است. این معماری زیربنای لازم برای برخوردی مستقل از سیستم، که می‌تواند از تشخیص تهاجم مربوط به یک سیستم منفرد تا ارتباط دادن فعالیت‌های سایت‌های مختلف شبکه برای تشخیص یک فعالیت مشکوک را بپوشاند، تأمین می‌کند.



شکل ۳-۹ معماری عامل

طعمه‌ها (Honeypots)

یک نوآوری نسبتاً جدید در تکنولوژی تشخیص تهاجم، honeypot است. honeypotها طعمه‌هائی هستند که با اغفال مهاجم او را از سیستم‌های حیاتی دور نگاه می‌دارند. honeypotها برای مقاصد زیر طراحی می‌شوند:

- منحرف کردن مهاجم از دست‌یابی به سیستم‌های حیاتی
- جمع‌آوری اطلاعات در مورد فعالیت‌های مهاجم
- تشویق مهاجم به باقی ماندن در سیستم تا زمانی که مدیران سیستم بتوانند عکس‌العمل نشان دهند

این سیستم‌ها با اطلاعات ساختگی که بظاهر ارزشمند جلوه می‌کنند طوری پر می‌شوند که کاربران قانونی معمولاً به آنها دسترسی پیدا نمی‌کنند. بنابراین هرگونه تلاش برای دست‌یابی به honeypot مشکوک تلقی خواهد شد. سیستم با پایشگرهای حساس و ثبت‌کننده‌های وقایع طوری تجهیز شده است که این نوع دست‌یابی‌های مشکوک را تشخیص داده و اطلاعات مربوط به فعالیت‌های تهاجمی را جمع‌آوری نماید. چون هر حمله‌ای بر علیه honeypot موفقیت‌آمیز بنظر خواهد رسید، مدیران شبکه فرصت کافی برای تجهیز شدن و دنبال کردن مهاجم، قبل از اینکه سیستم‌های اصلی در معرض خطر قرار گیرند، را خواهند داشت.

تلاش‌های اولیه، در استفاده از یک کامپیوتر honeypot و آدرس‌های IP مخصوصی که برای جلب مهاجمین انتخاب شده بود، خلاصه می‌شد. تحقیقات جدیدتر بر ساخت شبکه‌های honeynet که تمام سازمان را تقلید کرده و احتمالاً از دیتای واقعی یا شبیه‌سازی شده استفاده کنند متمرکز شده است. بمحض اینکه هَکرها در محدودهٔ شبکه قرار گیرند، مسئولین می‌توانند رفتار آنها را با جزئیات کامل مشاهده کرده و سیاست‌های دفاعی مناسب را بیابند.

فرمت مبادلهٔ تشخیص تهاجم (Intrusion Detection Exchange Format)

برای تسهیل ساخت سیستم‌های گستردهٔ تشخیص تهاجم که بتوانند در عرض محدودهٔ وسیعی از رایانه‌ها و محیط‌ها عمل نمایند، نیاز به استانداردهائی است که بتوانند عملیات مختلف بین‌شبکه‌ای را حمایت کنند. چنین استانداردهائی، کانون فعالیت گروه کاری تشخیص تهاجم IETF را تشکیل می‌دهد. هدف این گروه کاری تعریف فرمت‌های دیتا و مبادلهٔ روش‌هائی برای به اشتراک گذاشتن اطلاعات جالب در زمینهٔ تشخیص تهاجم و سیستم‌های پاسخگو به مساله و همچنین سیستم‌های مدیریتی است که ممکن است نیاز به تعامل با آنها داشته باشند. خروجی‌های این سیستم کاری، شامل موارد زیراند:

۱- یک سند نیازمندی‌ها که توصیف‌کنندهٔ نیازهای عملیاتی سطح بالا برای ارتباط بین سیستم‌های تشخیص تهاجم، و همچنین نیازهای ارتباطی بین سیستم‌های تشخیص تهاجم با سیستم‌های مدیریت، بهمراه دلایل مستدل و منطقی برای این نیازهاست. برای توجیه این نیازها بایستی از سناریوهای مناسب استفاده شود.

۲- تعیین مشخصه‌های یک زبان محاورهٔ مشترک برای تشخیص،که فرمت داده‌هائی که این نیازها را ارضاء می‌کنند را هم تعیین نماید.

۳- یک سند ساختاری که بهترین پروتکل‌های موجود برای ارتباط بین سیستم‌های تشخیص تهاجم را شناسائی نموده و مشخص می‌سازد که فرمت‌های انتخاب شده دیتا چگونه با آنها مرتبط‌اند.

در زمان نگارش این مطلب، تمام این اسناد در مرحلهٔ پیش‌نویس اسناد اینترنتی هستند.

۹-۳ مدیریت کلمهٔ عبور

حفاظت کلمهٔ عبور

خط مقدم دفاع در مقابل مهاجمین سیستم، کلمهٔ عبور است. تقریباً تمام سیستم‌های چندکاربره از کاربران می‌خواهندکه نه تنها یک نام یا شناسه (ID) را وارد سیستم کنند بلکه یک کلمهٔ عبور (password) را نیز عرضه نمایند. کلمهٔ عبور برای سنجش اعتبار ID فردی که می‌خواهد به سیستم وارد شود مورد استفاده قرار می‌گیرد. ID نیز بنوبهٔ خود، امنیت را به راه‌های زیر فراهم می‌آورد:

- ID تعیین می‌کند که آیا کاربر برای دست‌یابی به سیستم دارای اعتبار است. در بعضی سیستم‌ها، تنها کسانی که قبلاً ID آنها در سیستم ذخیره شده است می‌توانند به سیستم دست یابند.
- ID امتیازاتی را که به کاربر اختصاص داده شده است، تعیین می‌کند. کاربران محدودی ممکن است دارای وظایف سوپروایزری سیستم باشند که طبیعتاً بایستی به آنها این اجازه داده شود که فایل‌ها را خوانده و یا عملیاتی را انجام دهند که معمولاً از نظر پنهان است. بعضی سیستم‌ها دارای امکانات پذیرش میهمان و یا افراد ناشناس بوده و طبیعتاً این افراد بایستی محدودیت‌های زیادتر و یا امتیازات کمتری نسبت به سایرین داشته باشند.
- ID بصورتی که معمولاً کنترل دست‌یابی منصفانه نامیده می‌شود مورد استفاده قرار می‌گیرد. بعنوان مثال، یک کاربر ممکن است با لیست نمودن IDهای کاربران دیگر به آنها اجازه دهد تا فایل‌هائی که متعلق به اوست را بخوانند.

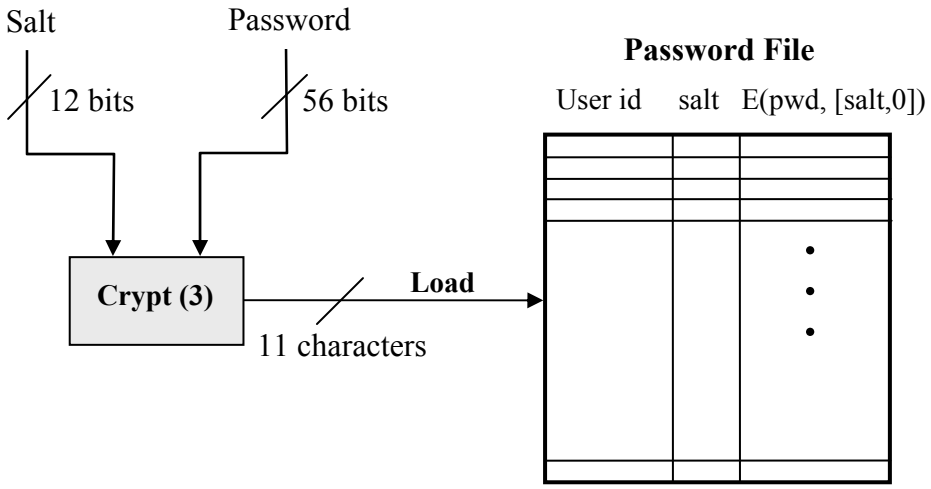
آسیب‌پذیری کلمات عبور

برای فهم ماهیت خطراتی که متوجه سیستم‌های مبتنی بر کلمهٔ عبور است، اجازه دهید روشی که بصورت گسترده در UNIX استفاده می‌شود و در آن هیچگاه کلمات عبور به فرم باز ذخیره نمی‌شوند را مطالعه کنیم. روش چنین است (شکل ۴-۹الف). هر کاربر یک کلمهٔ عبور که حداکثر دارای هشت کاراکتر قابل چاپ است را امتحان می‌کند. این کلمهٔ عبور به یک مقدار ۵۶- بیتی (با استفاده از کُد ۷- بیتی ASCII) تبدیل می‌شود که بعنوان کلید ورودی یک روش رمزنگاری بکار می‌رود. روش رمزنگاری که crypt (3) نام دارد مبتنی بر DES است. الگوریتم DES بنحوی دستکاری شده است که از یک "salt" ۱۲- بیتی استفاده می‌کند. نوعاً این مقدار وابسته به زمانی است که کلمهٔ عبور به کاربر اختصاص داده می‌شود. الگوریتم DES دستکاری شده، روی یک دیتای ورودی که یک بلوک ۶۴- بیتی از 0 های باینری است عمل می‌کند. خروجی الگوریتم سپس بعنوان ورودی یک رمزنگار دوم مورد استفاده قرار می‌گیرد. این پردازش مجموعاً برای ۲۵ بار رمزنگاری تکرار می‌شود. خروجی ۶۴- بیتی این پردازش آنگاه به یک دنبالهٔ ۱۱-کارکتری تبدیل می‌گردد. آنگاه hash کلمهٔ عبور بهمراه کپی متن ساده "salt" در فایل کلمهٔ عبور متناظر با ID کاربر ذخیره می‌گردد. نشان داده شده است که این متد در برابر انواع مختلفی از حملات شکستن رمز امن است [WAGN00].

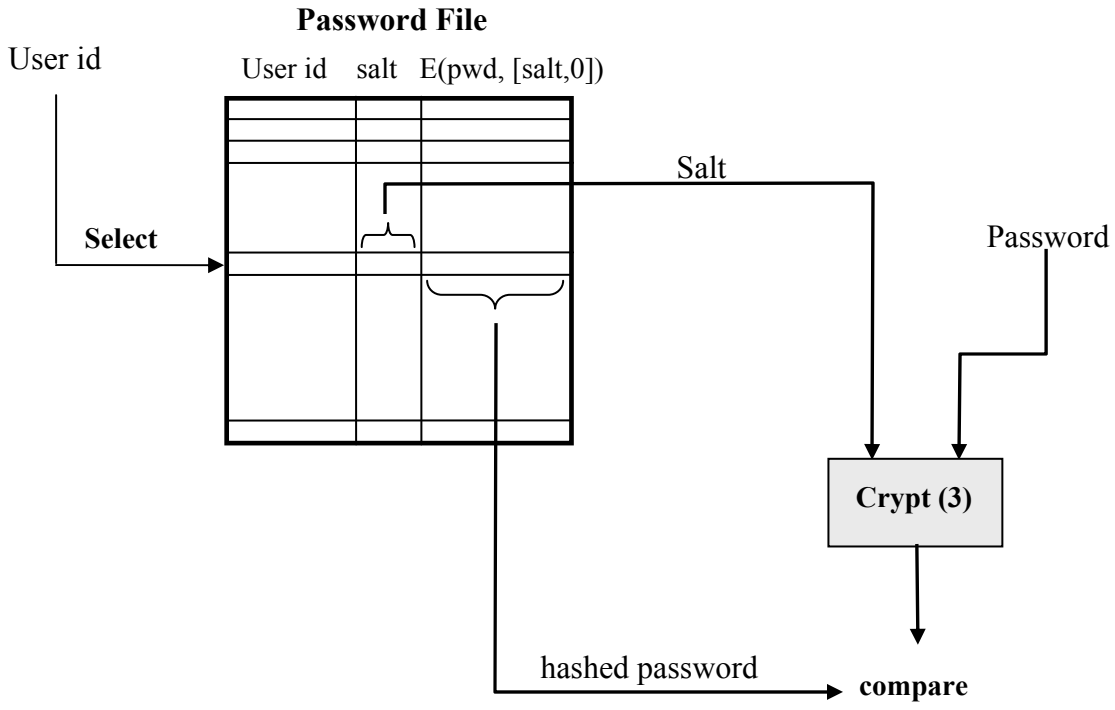
"salt" سه منظور را برآورده می‌سازد:

- نمی‌گذارد کلمات عبور مشابه در فایل کلمات عبور مشاهده شوند. حتی اگر دو کاربر کلمهٔ عبور یکسانی را انتخاب کنند، چون این کلمات عبور در زمان‌های متفاوتی به کاربران اختصاص یافته‌اند، بنابراین کلمات عبور پردازش شدهٔ انتهائی دو کاربر متفاوت خواهند بود.

- بطور مؤثری طول کلمات عبور را زیاد می‌کند بنحوی که کاربر لازم نیست دو کاراکتر اضافی را بخاطر بسپارد. در نتیجهٔ این عمل، تعداد کلمات عبور بمیزان ۴,۰۹۶ برابر زیاد شده و حدس زدن یک کلمهٔ عبور سخت‌تر می‌گردد.
- استفاده از سخت‌افزار آماده DES، که می‌تواند حملهٔ همه جانبه به کلمهٔ عبور را تسهیل کند، را غیرممکن می‌سازد.



(الف) بارگذاری یک کلمهٔ عبور جدید



(ب) تأئید یک کلمهٔ عبور

شکل ۴-۹ روش برخورد با کلمهٔ عبور در UNIX

وقتی یک کاربر می‌خواهد به یک سیستم UNIX وارد شود، او یک ID و یک کلمهٔ عبور را به سیستم عرضه می‌دارد. سیستم عامل با استفاده از ID وارد فایل لیست کلمات عبور شده و "salt" و کلمهٔ عبور رمزشده را بازخوانی می‌کند. "salt" و کلمهٔ عبور ارائه شده بتوسط کاربر، بعنوان ورودی‌های الگوریتم رمزنگاری مورد استفاده قرار می‌گیرند. اگر نتیجه امر برابر اندازهٔ ذخیره شدهٔ قبلی بود، کلمهٔ عبور پذیرفته می‌شود.

عمل رمزنگاری برای خنثی کردن حملاتی که استراتژی آنها حدس زدن کلمهٔ عبور است بکار می‌رود. اجرای نرم‌افزاری DES در مقایسه با نمونهٔ سخت‌افزاری آن کند بوده و تکرار ۲۵ مرتبهٔ آن، زمان لازم را ۲۵ برابر بیشتر می‌کند. از زمان طراحی اولیهٔ این روش تا کنون، دو تغییر در الگوریتم آن داده شده است. اولاً نسخه‌های جدیدتر الگوریتم، سرعت آن را افزایش داده‌اند. بعنوان مثال کِرم اینترنتی توصیف شده در فصل ۱۰ قادر شد تا با استفاده از یک الگوریتم رمزنگاری بهره‌ورتر از آنچه که در ابتدا بصورت استاندارد در سیستم‌های UNIX نصب شده بود، صدها کلمهٔ عبور را در طول مدت نسبتاً کوتاهی، در سیستم مورد تهاجم، حدس بزند. ثانیاً، عملکرد سخت‌افزارها روز به روز بهتر شده بطوری که اجازه می‌دهد الگوریتم‌های نرم‌افزاری نیز سریع‌تر عمل کنند.

بنابراین کلمهٔ عبور در سیستم UNIX با دو تهدید مواجه است. اول اینکه یک کاربر ممکن است با استفاده از تسهیلات پذیرش میهمان و یا روش دیگری به ماشین دسترسی پیدا کرده و آنگاه یک برنامهٔ حدس‌زنندهٔ کلمهٔ عبور، که شکنندهٔ کلمهٔ عبور خوانده می‌شود، را روی ماشین اجرا کند. مهاجم بایستی قادر باشد تا صدها و شاید هزارها کلمات عبور ممکن را با بکارگیری حداقل منابع امتحان کند. علاوه برآن، اگر یک دشمن قادر باشد تا یک نسخه از فایل کلمات عبور را بدست آورد، آنگاه یک برنامهٔ cracker (شکننده) خواهد توانست تا سر فرصت و روی ماشین دیگری کلمات عبور را کشف کند.

بعنوان مثال در ماه اوت سال ۱۹۹۳ میلادی، حضور یک شکنندهٔ کلمات عبور روی اینترنت گزارش گردید[MADS93]. این عمل با استفاده از یک کامپیوتر موازی Thinking Machines Corporation و به میزان ۱,۵۶۰ رمزنگاری در ثانیه برای هر واحد حاصل گردید. با استفاده از چهار واحد پردازش در هر گره پردازشگر (یک پیکربندی استاندارد)، این عمل به ۸۰۰,۰۰۰ رمزنگاری در ثانیه روی یک ماشین با ۱۲۸ گره (که اندازهٔ متوسطی است)، و ۶/۴ میلیون رمزنگاری در ثانیه روی یک ماشین با ۱,۰۲۴ گره افزایش یافت.

حتی این نرخ حدسیات، یک مهاجم صاحب عقل را قانع نمی‌سازد تا با استفاده از یک حملهٔ همه‌جانبه بخواهد تمام ترکیبات ممکن کاراکترها برای کشف یک کلمهٔ عبور را امتحان کند. بجای آن شکننده‌های کلمات عبور اغلب از این واقعیت استفاده می‌کنند که بعضی کاربران از کلمه‌های عبوری استفاده می‌کنند که به سهولت قابل حدس زدن هستند.

برخی کاربران وقتی اجازه دارند که کلمهٔ عبور را خود انتخاب کنند، کلمه‌ای را برمی‌گزینند که بطور ساده‌لوحانه‌ای کوتاه است. نتایج یک بررسی در دانشگاه Purdue در جدول ۳-۹ نشان داده شده است. در این بررسی، تغییر کلمات عبور روی ۵۴ ماشین مورد مطالعه قرار گرفت که تقریباً ۷,۰۰۰ شمارهٔ کاربری را در بر می‌گرفت. تقریباً ۳٪ کلمات عبور از ۳ کاراکتر و یا کمتر تشکیل شده بودند. یک مهاجم می‌توانست حملهٔ خود را با آزمایش کردن تمام کلمات عبور ممکن با طول ۳ و کمتر آغاز کند. یک علاج ساده این است که سیستم هر کلمهٔ عبور کمتر از مثلاً ۶ کاراکتر را نپذیرد و یا مثلاً کاربران را مجبور کند تا کلمات عبوری که حتماً ۸ کاراکتر داشته باشند را انتخاب نمایند. اکثر کاربران در مورد اعمال چنین محدودیتی شکایت نخواهند کرد.

طول کلمهٔ عبور تنها بخشی از مشکل است. بسیاری افراد وقتی اجازه دارند تا کلمهٔ عبور را خود انتخاب نمایند، کلمه‌ای را برمی‌گزینند که قابل حدس است. آنها مثلاً نام خود و یا نام خیابان منزل خود و یا یک کلمهٔ موجود در کتاب لغت و غیره را انتخاب می‌کنند. این امر کار شکستن کلمهٔ عبور را تسهیل می‌کند و شکنندهٔ کلمهٔ عبور فقط کافی است فایل کلمات عبور را

جدول ۳-۹ طول‌های مشاهده شده برای کلمات عبور [SPAF92a]

Length	Number	Fraction of Total
1	55	.004
2	87	.006
3	212	.02
4	449	.03
5	1260	.09
6	3035	.22
7	2917	.21
8	5772	.42
Total	13787	1.0

با کلمات عبور محتمل مقایسه نماید. چون بسیاری از مردم از کلمات عبور قابل حدس استفاده می‌کنند، این نوع استراتژی تقریباً در تمام سیستم‌ها موفق خواهد بود.

یک گزارش از موفقیت‌آمیزبودن حدس زدن کلمهٔ عبور در [KLEI90]، نمایش داده شده است. نویسندهٔ گزارش، فایل‌های کلمات عبور UNIX را از منابع متنوعی جمع‌آوری نموده که شامل تقریباً ۱۴,۰۰۰ کلمهٔ عبور رمزشده است. نتیجهٔ، که نویسنده بحق آن را رعب‌آور توصیف کرده است، در جدول ۴-۹ نشان داده شده است. رویهم‌رفته تقریباً یک چهارم کلمات عبور، لو رفته بودند. استراتژی مورد استفاده چنین بود:

۱- نام کاربر، حرف اول نام او، نام فامیل او، شمارهٔ حساب و سایر اطلاعات شخصی او را امتحان کنید. در مجموع ۱۳۰ تبدیل مختلف برای هر کاربر مورد آزمایش قرار گرفته بود.

۲- کلمات موجود در کتاب لغت‌های مختلف را امتحان کنید. نویسنده، یک کتاب لغت با بیش از ۶۰,۰۰۰ لغت را جمع‌آوری کرده است که شامل کتاب لغت برخط روی سیستم، و لیست‌های مختلفی برابر آنچه در جدول آمده است می‌باشد.

۳- تبدیل‌های متفاوتی روی لغات جمع‌آوری شده در بند ۲ انجام دهید. این شامل تبدیل حرف اول لغت به حرف بزرگ و یا یک کاراکتر کنترلی، تبدیل تمام لغت به حروف بزرگ، معکوس کردن کلمه و تبدیل حرف "O" به رقم "0" و غیره است. این تبدیل‌ها تقریباً یک میلیون کلمه به لیست اضافه می‌کنند.

۴- تبدیل‌های بزرگ کردن یک یا چند حرف که در بند ۳ انجام نشده است را به لغات بند ۲ اعمال کنید. این امر تقریباً دو میلیون کلمهٔ اضافی را به لیست اضافه می‌نماید.

بنابراین، تست یادشده تقریباً ۳ میلیون کلمه را در بر می‌گرفت. با استفاده از سریع‌ترین Thinking Machine که قبلاً از آن یاد شد، رمزنگاری تمام این کلمات با استفاده از تمام مقادیر "salt" به زمانی کمتر از یک ساعت نیاز داشت. بخاطر داشته باشید که چنین جستجوی جامعی تقریباً ۲۵٪ احتمال موفقیت داشته است در حالی که حتی یک مورد موفقیت هم ممکن است برای دست‌یابی به امتیازات وسیعی در سیستم کافی باشد.

جدول ۴-۹ کلمات عبور لو رفته از یک مجموعهٔ نمونهٔ با ۱۳,۷۹۷ حساب [KLEI90]

Type of Password	Search Size	Number of Matches	Percentage of Password Matched	Cost/Benefit Ratio *
User/account name	130	368	2.7%	2.830
Character sequences	866	22	0.2%	0.025
Numbers	427	9	0.1%	0.021
Chinese	392	56	0.4%	0.143
Place names	628	82	0.6%	0.131
Common names	2239	548	4.0%	0.245
Female names	4280	161	1.2%	0.038
Male names	2866	140	1.0%	0.049
Uncommon names	4955	130	0.9%	0.026
Myths&legends	1246	66	0.5%	0.053
Shakespearean	473	11	0.1%	0.023
Sport terms	238	32	0.2%	0.134
Science fiction	691	59	0.4%	0.085
Movies and actors	99	12	0.1%	0.121
Cartoons	92	9	0.1%	0.098
Famous people	290	55	0.4%	0.190
Phrases and patterns	933	253	1.8%	0.271
Surnames	33	9	0.1%	0.273
Biology	58	1	0.0%	0.017
System dictionary	9683	1027	7.4%	0.052
Machine names	9018	132	1.0%	0.015
Mnemonics	14	2	0.0%	0.143
King James Bible	7525	83	0.6%	0.011
Miscellaneous words	3212	54	0.4%	0.017
Yiddish words	56	0	0.0%	0.000
Asteroids	2407	19	0.1%	0.007
Total	62727	3340	24.2%	0.053

* Cost /Benefit Ratio = Number of Matches / Search Size

کنترل دست‌یابی

یکی از راه‌های مقابله با تهاجم به کلمات عبور، جلوگیری از دست‌یابی دشمن به فایل کلمات عبور است. اگر قسمتی از فایل که کلمات عبور رمزشده در آن نگهداری می‌شود تنها بتوسط یک کاربر دارای امتیازات ویژه قابل دست‌یابی باشد، آنگاه دشمن بدون دانستن کلمهٔ عبور آن کاربر ویژه، امکان دسترسی به فایل را نخواهد داشت. [SPAF92a] چند اشکال در این استراتژی را یادآوری می‌کند:

- بسیاری از سیستم‌ها، شامل بیشتر سیستم‌های UNIX، در معرض تعرض‌های پیش‌بینی نشده قرار دارند. همینکه مهاجمی توانست به نحوی به سیستم راه یابد، ممکن است بخواهد مجموعه‌ای از کلمات عبور را بدست آورد تا از طریق حساب‌های مختلف برای ورود به سیستم اقدام نموده و بدین نحو خطر تشخیص خود را کم کند. همچنین یک کاربر دارای اشتراک ممکن است بخواهد از حساب مشترک دیگری استفاده کرده تا داده‌های گران‌قیمت را خوانده و یا در سیستم خرابکاری نماید.

- یک رویداد حفاظتی ممکن است فایل کلمات عبور را ناخوانا ساخته و در نتیجه تمام اشتراک‌ها را به مخاطره اندازد.
- برخی کاربران، دارای اشتراک روی ماشین‌های دیگر در نواحی حفاظتی دیگری هستند ولی از همین کلمهٔ عبور در آنجا هم استفاده می‌کنند. بنابراین اگر فردی بتواند کلمهٔ عبور آنها روی یک ماشین را کشف کند، ماشین دیگری در محل دیگری نیز می‌تواند در معرض خطر واقع شود.

بهمین دلیل، یک استراتژی مؤثرتر این است که کاربران را مجبور کرد تا کلمهٔ عبوری را انتخاب کنند که حدس زدن آن مشکل باشد.

استراتژی‌های انتخاب کلمهٔ عبور

درسی که از دو آزمایش بالا (جداول ۳-۹ و ۴-۹) آموخته می‌شود این است که، در صورتی که امر به خود آنها واگذار شود، بسیاری از کاربران کلمهٔ عبوری را انتخاب می‌کنند که یا خیلی کوتاه بوده و یا حدس زدن آن خیلی آسان است. از سوی دیگر اگر به کاربران کلمات عبوری اختصاص یابد که از هشت کاراکتر قابل چاپ تصادفی تشکیل شده باشد، شکستن کلمهٔ عبور تقریباً غیرممکن خواهد بود. اما در این صورت بخاطر سپردن کلمهٔ عبور برای بیشتر کاربران امری غیرممکن است. خوشبختانه حتی اگر فضای کلمات عبور را به دنباله‌ای از کاراکترها که بطور معقول قابل بخاطر سپردن هستند محدود کنیم، باز هم اندازهٔ فضا بحدی بزرگ خواهد ماند که شکستن کلمهٔ عبور عملی نخواهد بود. بنابراین هدف ما این است که کلمات عبور قابل حدس را بنحوی حذف کنیم که باز هم کاربر قادر باشد کلمهٔ عبور خود را بخاطر بسپارد. در این مقوله از چهار تکنیک عمده استفاده می‌شود:

- تعلیم کاربر
- استفاده از کلمات عبوری که بتوسط کامپیوتر تولید می‌شود
- کنترل غیرفعال کلمهٔ عبور
- کنترل فعال کلمهٔ عبور

کاربران را بایستی از اهمیت بکاربردن کلمات عبوری که بسختی قابل حدس زدن هستند آگاه کرد و برای آنها خط‌مشی مناسبی برای انتخاب کلمات عبور قوی ارائه داد. این استراتژی **تعلیم کاربر** در اکثر سازمان‌ها، علی‌الخصوص در جاهائی که جمعیت زیاد بوده و تغییر و تحول زیادی در جریان است، احتمال موفقیت کمی دارد. بسیاری از کاربران خط‌مشی‌ها را بسهولت زیرپا می‌گذارند. برخی دیگر ممکن است قضاوت صحیحی نسبت به یک کلمهٔ عبور محکم نداشته باشند. مثلاً کاربران زیادی (اشتباهاً) معتقدند که معکوس کردن حروف یک کلمه، و یا بزرگ نوشتن حرف آخر یک کلمه، آن را غیرقابل حدس خواهد ساخت.

کلمات عبور تولید شده بتوسط کامپیوتر نیز دارای مشکل‌اند. اگر کلمهٔ عبور طبیعت کاملاً تصادفی داشته باشد، کاربر قادر به بخاطر سپردن آن نخواهد بود. حتی اگر کلمهٔ عبور قابل تلفظ هم باشد، کاربر ممکن است در بخاطر سپردن آن مشکل داشته و وسوسه شود که آن را یادداشت نماید. بطور کلی، روش ساخت کلمهٔ عبور بتوسط کامپیوتر دارای تاریخچه‌ای است که عدم تمایل به پذیرش آن از سوی کاربران را نشان می‌دهد. FIPS PUB 181 یکی از بهترین طراحی‌های مربوط به تولید اتوماتیک کلمهٔ عبور را ارائه داده است. این استاندارد نه تنها روش عمل را توصیف می‌کند بلکه کُد برنامه نوشته شده به

زبان C را نیز ارائه می‌دهد. الگوریتم با ایجاد بخش‌های قابل تلفظ، آنها را به‌هم چسبانده تا یک کلمهٔ عبور را ایجاد نماید. یک مولد اعداد تصادفی برای تولید دنبالهٔ کاراکترها جهت ساخت بخش‌های کلمهٔ عبور بکار می‌رود.

یک استراتژی **کنترل غیرفعال کلمهٔ عبور** این است که خود سیستم هرچندوقت یکبار برنامهٔ شکستن کلمهٔ عبور داخلی خود را اجرا کرده تا کلمات عبور قابل حدس را پیدا کند. سیستم اگر بتواند یک کلمهٔ عبور را با حدس بدست آورد آن را حذف کرده و کاربر را از این موضوع مطلع می‌کند. این تاکتیک چندین نقطهٔ ضعف دارد. اول اینکه اگر قرار باشد تا عمل درست انجام شود، منابع سیستم را بشدت بکار خواهد گرفت. دوم اینکه یک دشمن مصمم که قادر به ربودن یک فایل کلمات عبور باشد می‌تواند ساعتها و یا حتی روزها تمام زمان CPU را برای دسترسی به هدف خود بکار گیرد. علاوه بر این هر کلمهٔ عبور موجود تا زمانی که کنترل غیرفعال کلمهٔ عبور، آن را کشف کند قابل تعرض خواهد ماند.

امیدوارکننده‌ترین روش برای امنیت کلمهٔ عبور، یک **کنترل‌کنندهٔ فعال کلمهٔ عبور** است. در این روش به یک کاربر اجازه داده می‌شود تا کلمهٔ عبور خود را انتخاب کند. ولی در هنگام انتخاب، سیستم به دنبال این می‌گردد که آیا این انتخاب مجاز است و در صورتی که این شرط برقرار نباشد آن را نمی‌پذیرد. چنین کنترل‌کننده‌هائی بر اساس این فلسفه قرار دارند که با هدایت کافی از طرف سیستم، کاربران می‌توانند کلمات عبور قابل بخاطرسپردن را از بین مجموعهٔ بزرگی از کلمات عبور که احتمال حدس زدن آنها در یک حملهٔ لغت‌نامه‌ای محتمل است، انتخاب کنند.

حیلهٔ کنترل‌کنندهٔ فعال کلمهٔ عبور این است که میخواهد توازنی بین خواست کاربر با استحکام کلمهٔ عبور ایجاد کند. اگر سیستم کلمات عبور زیادی را دفع کرده و قبول نکند، کاربران از این شکایت خواهند کرد که انتخاب کلمهٔ عبور کاری مشکل است. اگر سیستم از یک الگوریتم ساده برای تعریف آنچه قابل قبول است استفاده کند، این خود هدایت‌کنندهٔ شکنندگان رمز عبور برای بهبود بخشیدن به روش‌های گمانه‌زنی خود خواهد بود. در بقیهٔ این قسمت، نگاهی به برخوردهای مختلف در زمینهٔ کنترل فعال کلمهٔ عبور خواهیم انداخت.

اولین روش، یک سیستم ساده اعمال قانون است. بعنوان مثال قوانین زیر را می‌توان اعمال کرد:

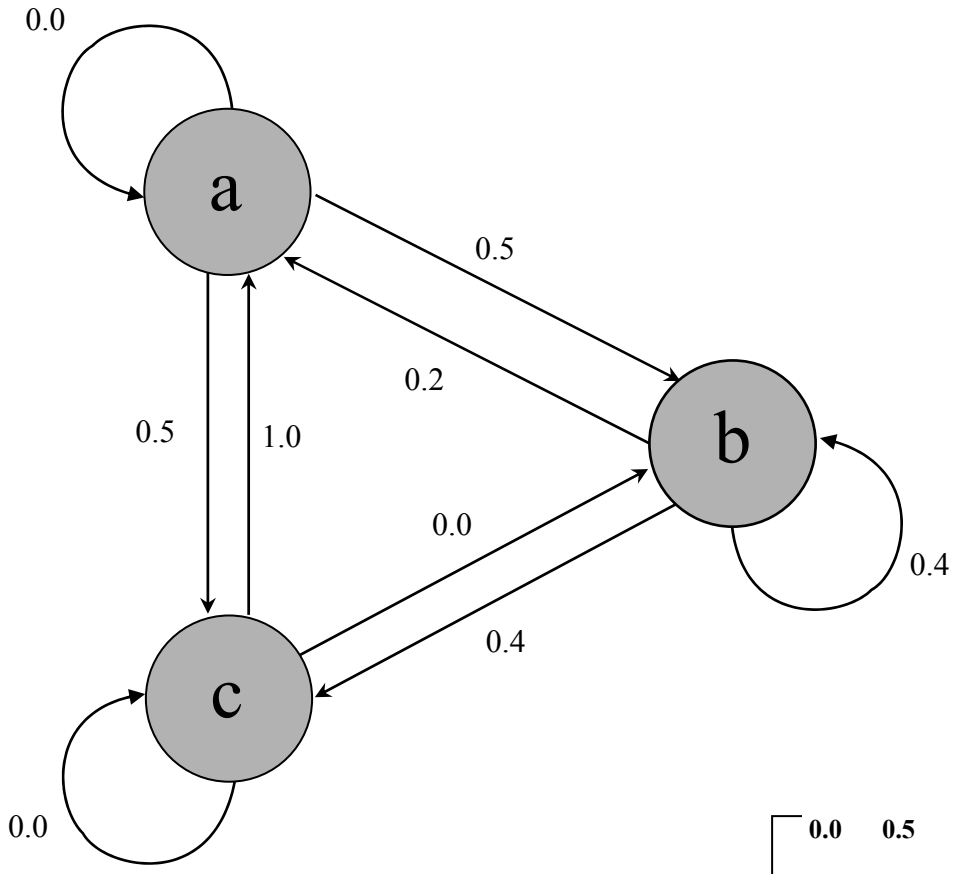
- تمام کلمات عبور بایستی حداقل ۸ کاراکتر طول داشته باشند.
- در ۸ کاراکتر اول کلمهٔ عبور بایستی حداقل یک حرف بزرگ، یک حرف کوچک، یک عدد و یک علامت وجود داشته باشد.

این قوانین را می‌توان بهمراه نصایح دیگر برای کاربر بیان نمود. اگرچه این نوع برخورد نسبت به تعلیم سادهٔ کاربران ارجحیت دارد، ولی برای ناامید کردن شکنندگان کلمهٔ عبور ممکن است کافی نباشد. این روش به قفل‌شکنان هشدار می‌دهد که کدام کلمات عبور را امتحان نکنند ولی بازهم ممکن است منجر به شکستن کلمهٔ عبور شود.

روش ممکن دیگر، جمع‌آوری یک لیست از کلمات عبور «بد» است. وقتی کاربری کلمهٔ عبوری را انتخاب می‌کند، سیستم آن را آزمایش کرده تا اطمینان یابد که در لیست غیرقابل قبول قرار نداشته باشد. این روش دارای دو مشکل است:

- **فضا**: برای مؤثر بودن روش، لیست بایستی خیلی بزرگ باشد. بعنوان مثال، لیستی که در پروژهٔ Purdue [SPAF92a] مورد استفاده قرار گرفت بیش از ۳۰ مگابایت حافظه را اشغال می‌کرد.
- **زمان**: زمان لازم برای جستجو در چنین لیست بزرگی، خود ممکن است خیلی زیاد باشد. علاوه بر آن برای کنترل کردن تمام تبدیلات ممکن روی لغات لیست، یا آن کلمات بایستی در لیست وجود داشته باشند، که حجم لیست را عظیم خواهد کرد، و یا برای هر کلمه پردازش دیگری نیز مورد نیاز خواهد بود.

دو تکنیک امیدوارکننده برای ساخت یک کنترل‌کنندهٔ فعال مؤثر و بهره‌ور، که بر مبنای نپذیرفتن لغات یک لیست قرار دارد، وجود دارد. یکی از این دو روش از یک مدل مارکوف (Markov) برای تولید کلمات عبور قابل حدس استفاده می‌کند [DAVI93]. شکل ۵-۹ یک نسخهٔ ساده شده از چنین مدلی را نشان می‌دهد. این مدل زبانی را نشان می‌دهد که الفبای آن دارای سه حرف است. حالت سیستم در هر لحظه برابر آخرین حرفی است که در سیستم پردازش شده است. مقدار نشان داده شده روی هر فلش این احتمال را نشان می‌دهد که حرفی به دنبال حرف دیگر قرار گیرد. بنابراین اگر مثلاً حرف فعلی a باشد، احتمال این که حرف بعدی b باشد 0.5 است.



$$M = \{3, \{a, b, c\}, T, 1\} \quad \text{where} \quad T = \begin{bmatrix} 0.0 & 0.5 & 0.5 \\ 0.2 & 0.4 & 0.4 \\ 1.0 & 0.0 & 0.0 \end{bmatrix}$$

مثالی از دنباله‌ای که مربوط به این زبان است : abbcacaba

مثالی از دنباله‌ای که مربوط به این زبان نیست : aacccbaaa

شکل ۵-۹ مثالی از مدل مارکوف

یک مدل مارکوف در حالت کلی دارای چهار عنصر [$m, A, \mathbf{T}, k$] است که درآن m تعداد حالات، A فضای حالات، $\mathbf{T}$ ماتریس احتمالات عبور از یک حالت به حالت دیگر و k مرتبهٔ مدل است. برای یک مدل مرتبهٔ k ام احتمال عبور به یک حرف بخصوص بستگی به نظم k حرف تولیدشدهٔ قبلی دارد. شکل ۵-۹ یک مدل سادهٔ مرتبهٔ اول را نشان می‌دهد.

دست اندرکاران از تولید و استفاده از یک مدل مرتبهٔ دوم نیز خبر می‌دهند. برای شروع، از تمام کلمات عبور قابل حدس یک لغت‌نامه ساخته می‌شود. آنگاه ماتریس عبور به طریق زیر ساخته می‌شود:

۱- ماتریس فرکانس $\mathbf{f}$ را بسازید، که در آن $\mathbf{f}(i,j,k)$ تعداد وقوع سه حرفی‌های شامل کاراکترهای i ام، j ام و k ام است. مثلاً کلمه عبور *parsnips* سه حرفی‌های par , ars , rsn , sni , nip , ips را تولید می‌کند.

۲- برای هر دوحرفی ij، $\mathbf{f}(i,j,\infty)$ را بعنوان تعداد کل سه حرفی‌هائی که با ij شروع می‌شود حساب کنید. مثلاً $\mathbf{f}(a,b,\infty)$ شامل تعداد کل سه حرفی‌های بصورت aba، abb، abc و غیره خواهد بود.

۳- مؤلفه‌های ماتریس $\mathbf{T}$ را بصورت زیر محاسبه کنید:

$$T_{(i,j,k)} = \frac{f_{(i,j,k)}}{f_{(i,j,\infty)}}$$

نتیجهٔ امر مدلی است که ساختار کلمات لغت‌نامهٔ ساخته شده را نشان می‌دهد. با این مدل، سؤال «آیا این کلمهٔ عبور بد است؟» به سؤال «آیا این دنباله (کلمهٔ عبور) از این مدل مارکوف تولید می‌شود؟» تبدیل می‌گردد. برای یک کلمهٔ عبور داده شده، احتمالات تمام سه حرفی‌های آن را می‌توان جستجو کرد. آنگاه می‌توان نوعی تست آماری استاندارد برای تعیین اینکه آیا این کلمهٔ عبور بتوسط این مدل قابل تولید هست یا نیست را مورد استفاده قرار داد. کلمات عبوری که تولید آنها بتوسط این مدل محتمل است، پذیرفته نمی‌گردند. نویسندگان مقاله نتایج خوبی برای مدل مرتبهٔ دوم را گزارش داده‌اند. سیستم آنها تقریباً تمام کلمات موجود در لغت نامهٔ آنها را بدام انداخته و بسیاری از کلمات عبور مناسبی که برای کاربر راحت هستند را می‌پذیرد.

یک روش کاملاً متفاوت بتوسط Spafford [SPAF92a,SPAF92b] گزارش شده است. این روش بر اساس استفاده از یک فیلتر Bloom [BLOO70] قرار دارد. برای شروع، عمل فیلتر Bloom را توضیح می‌دهیم. یک فیلتر Bloom از مرتبهٔ k از یک مجموعهٔ k تائی از توابع hash مانند $H_1(x), H_2(x), \ldots, H_k(x)$ تشکیل شده است که در آن هر تابع، یک کلمهٔ عبور را به اندازهٔ hash آن در محدودهٔ 0 تا N-1 تبدیل می‌کند. یعنی

$$H_i(X_j) = y \qquad 1 \le i \le k\,; \qquad 1 \le j \le D \qquad 0 \le y \le N\text{-}1$$

که در آن

X_j = کلمهٔ j ام در لغت نامهٔ کلمات عبور

D = تعداد کلمات در لغت نامهٔ کلمات عبور

آنگاه روش زیر به لغت‌نامه اعمال می‌شود:

۱- یک جدول hash از N بیت تعریف می‌شود که تمام بیت‌ها در ابتدا 0 اند.

۲- برای هر کلمهٔ عبور، اندازه‌های k مقدار hash آن محاسبه شده و بیت‌های نظیر آن در جدول hash به 1 تعویض می‌شوند. بنابراین اگر برای مقداری از i و j، $H_i(X_j) = 67$ باشد، آنگاه بیت شصت و هفتم جدول hash مساوی 1 می‌شود و اگر بیت قبلاً 1 بوده است 1 باقی می‌ماند.

وقتی کلمهٔ عبور جدیدی به کنترل‌کنندهٔ کلمات عبور عرضه می‌شود، اندازه‌های k مقدار hash آن محاسبه می‌گردد. اگر تمام بیت‌های نظیر جدول hash مساوی 1 باشند، آنگاه کلمهٔ عبور پذیرفته نخواهد شد. تمام کلمات عبوری که در لغت‌نامه وجود دارند رد می‌شوند. اما بازهم تعدادی جواب مثبت اشتباه خواهیم داشت (یعنی کلمات عبوری که در لغت‌نامه قرار ندارند ولی جدول hash آنها تطبیق دارد). برای روشن شدن مطلب، روشی با دو تابع hash را درنظر بگیرید. فرض کنید که کلمات عبور *undertaker* و *halkhogan* در لغت‌نامه هستند اما *xG%#jj98* در آن نیست. علاوه بر آن فرض کنید

$H_1(\text{undertaker}) = 25$ $H_1(\text{hulkhogan}) = 83$ $H_1(\text{xG\%\#jj98}) = 665$

$H_2(\text{undertaker}) = 998$ $H_2(\text{hulkhogan}) = 665$ $H_2(\text{xG\%\#jj98}) = 998$

اگر کلمهٔ عبور xG%#jj98 به سیستم عرضه شود، با وجود این که در لغت‌نامهٔ سیستم قرار ندارد رد خواهد شد. اگر تعداد زیادی از این جواب‌های مثبت غلط از سیستم بیرون داده شود، انتخاب کلمهٔ عبور برای کاربران سخت خواهد شد. بنابراین علاقه‌مندیم روشی را انتخاب کنیم که این جواب‌های مثبت غلط را به حداقل برساند. می‌توان نشان داد که احتمال یک جواب مثبت غلط برابر مقدار تقریبی زیر است

$$P \approx (1- e^{kD/N})^k = (1- e^{k/R})^k$$

و یا بصورت معادل آن

$$R \approx \frac{-k}{\ln(1- P^{1/k})}$$

که در آن

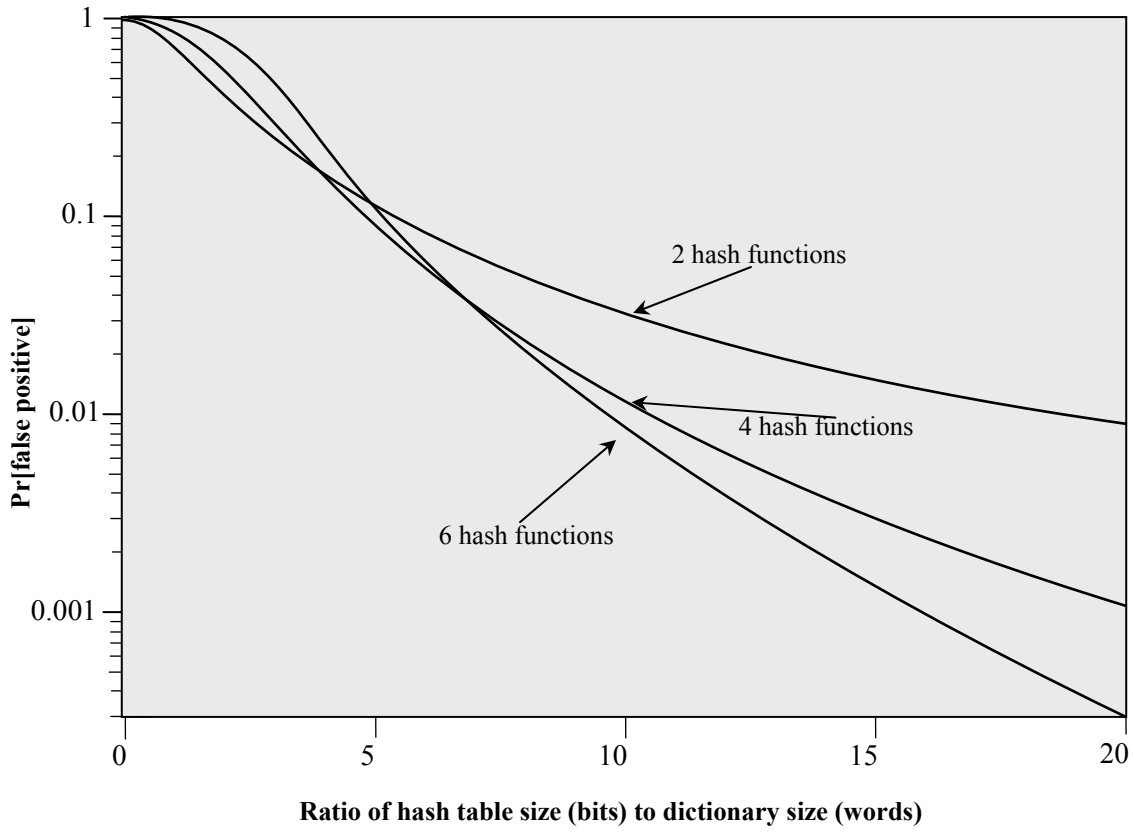
k = تعداد توابع hash

N = تعداد بیت‌های جدول hash

D = تعداد کلمات موجود در لغت‌نامه

$R = N/D$ نسبت اندازهٔ جدول hash (bits) به اندازهٔ لغت‌نامه (words)

شکل ۶-۹، P را بر حسب تابعی از R برای مقادیر مختلف k رسم کرده است. فرض کنید که یک لغت‌نامه یک میلیون لغت داشته و می‌خواهیم احتمال نپذیرفتن یک کلمهٔ عبور که در این لغت‌نامه قرار ندارد برابر ۰/۰۱ باشد. اگر شش تابع hash انتخاب کنیم، نسبت مورد نیاز $R = ۹/۶$ است. بنابراین یک جدول hash با $۹/۶ \times ۱۰^۶$ بیت و یا تقریباً ۱/۲ مگابایت حافظه مورد نیاز است. در مقایسه، ذخیره کردن تمام لغت‌نامه به حدود ۸ مگابایت حافظه نیازمند است. بنابراین فشردگی حاصل تقریباً با فاکتور ۷ خواهد بود. علاوه بر این، کنترل کردن کلمهٔ عبور شامل محاسبهٔ آسان شش تابع hash بوده و مستقل از حجم لغت‌نامه است در حالی که در صورت استفاده از کل لغت‌نامه، حجم جستجو بسیار وسیع‌تر خواهد شد.



شکل ۶-۹ عملکرد فیلتر Bloom

۹-۴ منابع مطالعاتی

دو منبع بررسی کامل تشخیص تهاجم [BACE00] و [PROC01] هستند. یک بررسی مختصرتر ولی کاملاً ارزشمند در [BACE01] ارائه شده است. دو مقالهٔ تحقیقی کوتاه ولی سودمند [KENT00] و [MCHU00] می‌باشند. [NING04] آخرین دستاوردهای تکنیک‌های تشخیص تهاجم را بررسی کرده است. [HONE01] توصیف محکمی از honeypots نموده و تحلیلی از ابزارها و روش‌های هَکرها را ارائه می‌دهد.

BACE00 Bace, R. *Intrusion Detection*. Indianapolis, IN: Macmillan Technical Publishing, 2000.
BACE01 Bace, R., and Mell, P. *Intrusion Detection System*. NIST Special Publication SP 800-31, November 2000.
HONE01 The Honeynet Project. *Know Your Enemy: Revealing the Security Tools, Tactics, and Motives of the Blackhat Community*. Reading, MA: Addison-Wesely, 2001.
KENT00 Kent, S. "On the Trail of Intrusions into Information Systems." *IEEE Spectrum*, December 2000.
MCHU00 McHugh, J.; Christie, A.; and Allen, j."The Role of Intrusion Detection Systems." *IEEE Software*, September/October 2000.
NING04 Ning, P., et al. "Techniques and tools for analyzing Intrusion Alerts." *ACM Transactions on Information and system Security*, May 2004.
PROC01 Proctor, P. *The Practical Intrusion Detection Handbook*. Upper Saddle River, NJ: Prentice Hall, 2001.

وِب سایت‌های مفید

- **CERT Coordination Center**: سازمانی که از دل تیم پاسخگوئی به فوریت‌های کامپیوتری، بتوسط آژانس پروژه‌های پیشرفتهٔ تحقیقاتی وزارت دفاع آمریکا بیرون آمد. این سایت اطلاعات مفیدی در مورد تهدیدهای امنیتی اینترنت، آسیب‌پذیری‌ها و آمار حملات دارد.
- **Honeynet Project**: یک پروژهٔ تحقیقاتی که مطالعهٔ تکنیک‌های هَکرها، و ساخت محصولات honeypot را بعهده دارد.
- **Honeypots**: مجموعهٔ خوبی از مقالات تحقیقاتی و گزارشات تکنیکی.
- **Intrusion Detection Working Group**: شامل کلیهٔ اسنادی است که بتوسط گروه تولید شده است.

۹-۵ واژه‌های کلیدی، سؤالات مرورکنندهٔ بحث و مسائل

واژه‌های کلیدی

audit record	سوابق ممیزی- گزارشات ممیزی
Bayes' theorem	قضیهٔ Bayes
base-rate fallacy	خطای نرخ پایه
honeypot	طعمه
intruder	مهاجم
intrusion detection	تشخیص تهاجم
intrusion detection exchange format	فرمت مبادلهٔ تشخیص تهاجم
password	کلمهٔ عبور
rule-based intrusion detection	تشخیص مبتنی بر قاعدهٔ تهاجم
salt	
statistical anomaly detection	تشخیص آماری ناهنجاری

سؤالات مرورکننده بحث

۹-۱ سه دسته از مهاجمین را نام برده و عملکرد آنها را بطور مختصر تعریف کنید.

۹-۲ دو تکنیک معمول برای حفاظت از فایل کلمات عبور کدامند؟

۹-۳ سه نتیجهٔ مفید استفاده از یک سیستم تشخیص تهاجم کدامند؟

۹-۴ تفاوت بین تشخیص آماری ناهنجاری و تشخیص مبتنی بر قاعده تهاجم چیست؟

۹-۵ چه مقادیری در تشخیص مبتنی بر پروفایل تهاجم مفید می‌باشند ؟

۹-۶ فرق بین تشخیص مبتنی بر قاعدهٔ ناهنجاری و شناسائی مبتنی بر قاعدهٔ نفوذ چیست؟

۹-۷ یک honeypot چیست ؟

۹-۸ در مقولهٔ مدیریت کلمهٔ عبور در سیستم عامل UNIX، یک "salt" چیست؟

۹-۹ چهار تکنیک مورد استفاده جلوگیری از حدس زدن کلمهٔ عبور را نام برده و عملکرد آنها را بطور مختصر تعریف کنید.

مسائل

۹-۱ یک رانندهٔ تاکسی در یک تصادف منجر به فوت شبانه، فردی را مصدوم نموده و فرار می‌کند. دو شرکت تاکسیرانی سبز و آبی در شهر فعالیت دارند. به شما گفته می‌شود:

- ۸۵٪ تاکسی‌های شهر سبز و ۱۵٪ آنها آبی هستند.
- یک شاهد، رنگ تاکسی باعث تصادم را آبی گزارش کرده است.

دادگاه میزان اعتماد به شاهد در تحت شرایط شب حادثه را سنجیده و نتیجه گرفته است که شاهد احتمالاً ۸۰٪ در شناسائی رنگ تاکسی موفق بوده است. احتمال اینکه تاکسی مورد نظر آبی بوده و سبز نباشد، چقدر است؟

۹-۲ فرض کنید که کلمات عبور از یک ترکیب چهارتائی کاراکترها از یک الفبای دارای ۲۶ کاراکتر انتخاب می‌شوند. فرض کنید که یک مهاجم بتواند کلمات عبور با نرخ یک کلمه در ثانیه را امتحان کند.

الف- با فرض اینکه تا پایان هر تلاش مهاجم هیچ فیدبکی به او داده نشود، زمان مورد انتظار برای کشف کلمهٔ عبور صحیح چقدر است؟

ب- با فرض اینکه بمحض وارد نمودن یک کاراکتر اشتباه، مهاجم پیام خطا دریافت نماید، زمان مورد انتظار برای کشف کلمهٔ عبور صحیح چقدر است؟

۹-۳ فرض کنید که عناصر یک منبع k تائی بصورت یکنواخت روی عناصر یک هدف p تائی نگاشت می‌شوند. اگر هر رقم بتواند یکی از r مقدار را داشته باشد، آنگاه تعداد عناصر منبع r^k و تعداد عناصر هدف مقدار کمتر r^p خواهد بود. یک عنصر مشخص منبع مثل x_i به یک عنصر مشخص هدف مثل y_j نگاشت می‌شود.

الف- احتمال اینکه با یکبار تلاش، عنصر صحیح منبع بتوسط مهاجم انتخاب شود چقدر است؟

ب- احتمال اینکه یک عنصر متفاوت منبع x_k $(x_i \neq x_k)$ که به همان عنصر هدف y_j نگاشت می‌شود انتخاب شود چقدر است؟

ج- احتمال اینکه با یکبار تلاش، عنصر صحیح هدف بتوسط مهاجم انتخاب شود چقدر است؟

۹-۴ یک تولیدکنندهٔ کلمات عبور قابل تلفظ، برای هر کلمهٔ عبور شش حرفی بطور تصادفی دو بخش را انتخاب می‌کند. فرم هر بخش CVC (consonant,vowel,consonant) است که در آن $V = < a,e,i,o,u >$ و $C = \overline{V}$ است

الف- تعداد کلمات عبور ممکن چندتاست؟

ب- احتمال اینکه یک مهاجم یک کلمهٔ عبور را درست حدس بزند چقدر است؟

۹-۵ فرض کنید که کلمات عبور منحصر به استفاده از ۹۵ کاراکتر قابل چاپ کُد ASCII بوده و هر کلمهٔ عبور از ۱۰ کاراکتر تشکیل شده باشد. فرض کنید که یک شکنندهٔ کلمات عبور با نرخ رمزنگاری ۶/۴ میلیون رمزنگاری در هر ثانیه برای کشف کلمهٔ عبور مشغول بکار شود. روی یک سیستم UNIX چقدر طول خواهد کشید تا این رمزشکن بتواند همهٔ کلمات عبور ممکن را امتحان کند؟

۹-۶ نظر به ریسک‌های شناخته شده سیستم کلمات عبور UNIX، اسناد SunOS-4.0 پیشنهاد می‌کند که فایل کلمات عبور را برداشته و بجای آن یک فایل قابل خواندن بتوسط عموم که /etc/publickey نامیده می‌شود را جایگزین نمائیم. برای کاربر A، اطلاعات ورودی فایل شامل یک شناسهٔ کاربر ID_A، کلید عمومی کاربر PU_a و کلید خصوصی نظیر آن PR_a است. کلید خصوصی با استفاده از DES و با کلیدی که از کلمهٔ عبور login کاربر (P_a) استخراج می‌شود رمز می‌شود. وقتی کاربر A به سیستم وصل می‌شود، سیستم $E [P_a , PR_a]$ را رمزگشائی کرده تا PR_a را بدست آورد.

الف- سیستم آنگاه تائید می‌کند که P_a بطور صحیح عرضه شده است. چگونه؟

ب- یک دشمن چگونه می‌تواند به این سیستم حمله کند؟

۹-۷ روش رمزنگاری کلمات عبور مورد استفاده در سیستم UNIX یک طرفه است و ممکن نیست که آن را بصورت معکوس بکار برد. در اینصورت آیا صحیح است که بگوئیم این روش در حقیقت یک کُد hash بوده و رمزنگاری کلمهٔ عبور نیست.

۹-۸ بیان کردیم که منظور کردن "salt" در روش کلمات عبور UNIX مشکل حدس زدن کلمهٔ عبور را بمیزان ۴,۰۹۶ برابر بالا می‌برد. اما "salt" بصورت متن ساده در همان جائی ذخیره می‌شود که کلمهٔ عبور رمزشده قرار می‌گیرد. بنابراین آن دو کاراکتر برای مهاجم معلوم بوده و نیازی به حدس زدن ندارد. در اینصورت چرا بیان می‌شود که "salt" امنیت را افزایش می‌دهد؟

۹-۹ با فرض اینکه شما مسالهٔ قبل را بطور صحیح پاسخ داده و اهمیت "salt" را درک کرده‌اید، به این سؤال پاسخ دهید. آیا این امکان وجود ندارد که با افزایش بسیار زیاد "salt" ، مثلاً به ۲۴ یا ۴۸ بیت بتوان همهٔ cracker ها را مأیوس کرد؟

۹-۱۰ فیلتر Bloom مورد بحث در بخش ۳-۹ را در نظر بگیرید. k را برابر تعداد توابع hash، N را برابر تعداد بیت‌های جدول hash و D را برابر تعداد کلمات لغت‌نامه فرض کنید.

الف- نشان دهید که تعداد بیت‌های 0 مورد انتظار در جدول hash از رابطهٔ زیر بدست می‌آید

$$\phi = (1- k / N)^{D}$$

ب- نشان دهید که احتمال اینکه یک کلمهٔ ورودی که در لغت‌نامه وجود ندارد بطور اشتباه بعنوان یک کلمهٔ موجود در لغت‌نامه پذیرفته شود مساوی است با

$$P = (1- \phi)^{k}$$

ج- نشان دهید که رابطهٔ قبل را می‌توان بصورت تقریبی $P \approx (1 - e^{-kD/N})^k$ نشان داد.

۹-۱۱ یک سیستم دست‌یابی به فایل طرح کنید که به کاربران معینی حق دست‌یابی خواندن و نوشتن به یک فایل را بر مبنای اجازه‌ای که بتوسط سیستم تعیین می‌گردد، بدهد. دستورات برنامه بایستی به فرمت زیر باشند:

تلاش کاربر A برای خواندن فایل F: READ (F, User A)

تلاش کاربر A برای ذخیره کردن فایل F که احتمالاً دستکاری هم شده است: WRITE (F, USER A)

هر فایل دارای یک header record است که شامل امتیازات اعتبارسنجی است. یعنی این سرآیند شامل لیستی از افرادی است که مجاز به خواندن و/ یا نوشتن هستند. این فایل قرار است با کلیدی رمزنگاری شود که در اشتراک کاربران نبوده و فقط برای سیستم شناخته شده است.

ضمیمهٔ ۹- الف خطای نرخ پایه (The Base-Rate Fallacy)

ابتدا نتایج مهمی از تئوری احتمالات را یادآوری نموده و سپس خطای نرخ پایهٔ را نشان می‌دهیم.

احتمال شرطی و پیشامدهای مستقل

اغلب لازم است تا احتمال پیشامدی را که مشروط به پیشامد دیگر است بدانیم. اثر این شرط این است که بعضی از وقایع از فضای نمونه حذف می‌شوند. مثلاً احتمال اینکه در انداختن دو طاس جمع خال‌ها ۸ باشد در صورتی که بدانیم یکی از خال‌ها حتماً زوج است چیست؟ می‌توان چنین استدلال کرد: چون یک طاس زوج است و جمع خال‌ها نیز زوج است بنابراین حتما طاس دوم هم بایستی زوج باشد. بنابراین، سه نتیجه موفق متساوی‌الاحتمال وجود دارد: (۶و۲) ، (۴و۴) و (۲و۶) که ازبین کل تعداد حالات ممکن ۲۷ = ۳×۳ - ۳۶ = (تعداد پیشامدهائی که هر دو تاس فردانند) - ۳۶ محاسبه می‌شوند. احتمال نتیجه شده برابر ۱:۹ = ۳:۲۷ است.

در حالت کلی، **احتمال شرطی** پیشامد A با فرض اینکه پیشامد B واقع شده باشد با Pr[A|B] نمایش داده شده و بصورت عبارت زیر تعریف می‌شود

$$\Pr[A \mid B] = \frac{\Pr[AB]}{\Pr[B]}$$

که در آن فرض می‌شود Pr[B] غیرصفر است.

در مثال ما، A = {جمع ۸ باشد} و B = {حداقل یکی از خالها زوج باشد} است. Pr[AB] تمام پیشامدهائی که در آنها جمع برابر ۸ و حداقل یکی از خالها زوج است را می‌پوشاند. همانطور که دیدیم سه پیشامد این‌چنینی وجود دارد. بنابراین Pr[AB]=3/36=1/12 . حال می‌توانیم Pr[A|B] را حساب کنیم:

$$Pr[A|B]=(1/12)/(3/4)=1/9$$

این نتیجه با استدلال قبلی همخوان است.

دو پیشامد A و B را **مستقل** ازهم گوئیم اگر Pr[AB]= Pr[A]Pr[B] باشد. بسولت می‌توان مشاهده نمود که اگر A و B مستقل باشند، Pr[A|B]=Pr[A] و Pr[B|A]=Pr[B] است.

قضیهٔ Bayes

یکی از مهم‌ترین نتایج تئوری احتمالات، قضیهٔ Bayes است. در ابتدا باید فرمول احتمال کلّ را بیان کنیم. هرگاه مجموعه‌ای از پیشامدهای دو به دو ناسازگار را داشته باشیم که اجتماع آنها همهٔ پیشامدهای ممکن را در بر بگیرد، و همچنین اگر یک پیشامد فرضی A را در نظر بگیریم، آنگاه میتوان نشان داد که

$$\Pr[\ A\ ] = \sum_{i=1}^{n} \Pr[\ A\ |\ E_i\ ] \Pr[\ E_i\ ] \qquad (۹-۱)$$

قضیهٔ Bayes را می‌توان بصورت زیر بیان کرد:

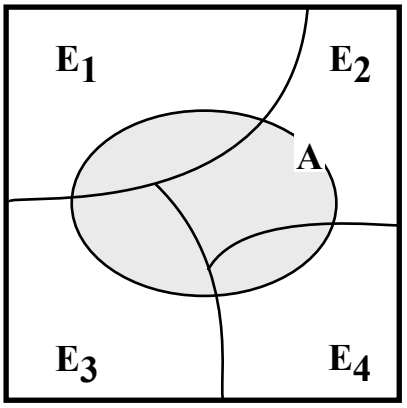
$$\Pr[E_i|A] = \frac{\Pr[A \mid E_i]\Pr[E_i]}{\Pr[A]} = \frac{\Pr[A \mid E_i]\Pr[E_i]}{\sum_{j=1}^{n} \Pr[A \mid E_j]\Pr[E_j]} \qquad (۹-۲)$$

شکل ۷-۹ الف مفهوم احتمال کلّ و قضیهٔ Bayes را نشان می‌دهد.

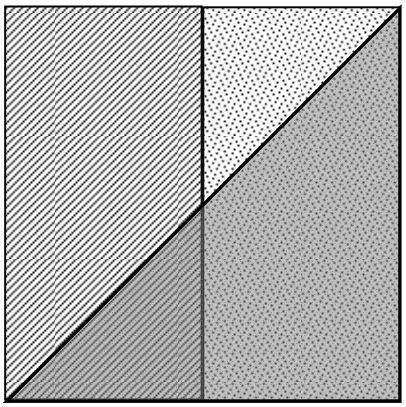
قضیهٔ Bayes برای محاسبهٔ «عواقب آینده»، یعنی احتمال وقوع پیشامدی با در دست داشتن شواهد مثبت در آن رابطه بکار می‌رود. برای مثال فرض کنید که دنباله‌ای از صفرها و یک‌ها از یک کانال نویزی منتقل می‌شوند. فرض کنید S0 و S1 به ترتیب پیشامدهای مرتبط با ارسال یک 0 و یک 1 در زمان معینی بوده و R0 و R1 نیز بترتیب پیشامدهای دریافت این 0 و 1 باشند. بازهم فرض کنید که ما احتمالات خروج این علائم از منبع را می‌دانیم و مثلاً $\Pr[S1] = P$ و $\Pr[S0] = 1\text{-}P$ است. حال خط را می‌پائیم تا ببینیم اگر 0 ارسال شود و یا 1 ارسال شود، هرچندوقت یکبار در دریافت آنها خطا خواهیم داشت و در این رابطه احتمالات $\Pr[R0 \mid S1] = P_a$ و $\Pr[R1 \mid S0] = P_b$ را حساب می‌کنیم. اگر یک 0 دریافت شود، ما می‌توانیم با استفاده از قضیهٔ Bayes احتمال شرطی یک خطا را محاسبه کنیم، یعنی احتمال شرطی این که یک 1 ارسال شده باشد در صورتی که یک 0 دریافت شده است:

$$\Pr[S1|R0] = \frac{\Pr[R0 \mid S1]\Pr[S1]}{\Pr[R0|S1]\Pr[S1] + \Pr[R0|S0]\Pr[S0]} = \frac{PaP}{PaP + (1-Pa)(1-P)}$$

شکل ۷-۹ب معادلهٔ بالا را نشان می‌دهد. در شکل، فضای نمونه با یک مربع واحد نمایش داده شده است. نصف مربع متناظر با S0 و نصف دیگر آن متناظر با S1 بوده و بنابراین $\Pr[S0] = \Pr[S1] = 0.5$ است. بهمین ترتیب نصف مربع متناظر با R0 و نصف آن متناظر با R1 بوده و در نتیجه $\Pr[R0] = \Pr[R1] = 0.5$ است. در ناحیه‌ای که نمایشگر S0 است، 1/4 آن ناحیه متناظر با R1 بوده و بنابراین $\Pr[R1 \mid S0] = 0.25$.است. سایر احتمالات شرطی بهمین ترتیب روشن‌اند.



(الف) نمودار نشان دهندهٔ مفاهیم



(ب) مثال

=S0; 0 sent = R0; 0 received
=S1; 1 sent = R1; 1 received

شکل ۷-۹ نمایش احتمال کلّ و قضیهٔ Bayes

نمایش خطای نرخ پایه

حالت زیر را در نظر بگیرید. از فرد بیماری برای تشخیص یک نوع مرض، آزمایشی گرفته شده است که نتیجهٔ آن مثبت است (یعنی این مرض را دارد). به شما گفته می‌شود:

- صحت آزمایش ۸۷٪ است (یعنی اگر بیمار این مرض را داشته باشد در ۸۷٪ موارد نتیجهٔ آزمایش مثبت است، و اگر بیمار این مرض را نداشته باشد بازهم در ۸۷٪ موارد نتیجهٔآزمایش صحیح است).
- احتمال بروز این مرض در یک جمعیت برابر ۱٪ است.

با فرض این که نتیجهٔ آزمایش مثبت است، احتمال این‌که بیمار این مرض را نداشته باشد چقدر است؟ بعبارت دیگر احتمال اینکه این یک اعلام خطر کاذب باشد چقدر است؟ برای یافتن جواب صحیح، نیاز به قضیهٔ Bayes داریم:

$$\Pr[\ well\ /\ positive\ ] = \frac{\Pr[\ positive\ /\ well\ ]\Pr[\ well\ ]}{\Pr[\ positive\ /\ disease\ ]\Pr[\ disease\ ] + [\Pr[\ positive\ /\ well\ ]\Pr[\ well\ ]}$$

$$= \frac{(0.13)(0.99)}{(0.87)(0.01) + (0.13)(0.99)} = 0.937$$

بنابراین در اکثریت وسیعی از موارد، وقتی شرایط مرض تشخیص داده می‌شود، این تشخیص یک تشخیص کاذب است.

این مسأله، در یک تحقیق [PIAT91]، به تعدادی افراد عرضه شد. بیشتر سوژه‌ها جوابشان ۱۳٪ بود. اکثریت بزرگی که شامل تعدادی از پزشکان نیز می‌شدند، عددی زیر ۵۰٪ را تخمین می‌زدند. خیلی از پزشکانی که حدسشان اشتباه بود با تأسف بیان می‌کردندکه «اگر شما صحیح می‌گوئید پس دلیلی برای آزمایشات کلینیکی وجود ندارد!» علت اینکه بیشتر افراد در پاسخ به این سؤال دچار اشتباه شده بودند این است که در محاسبهٔ ذهنی خود نرخ اصلی پیشامد(نرخ پایه) را منظور نکرده بودند. این خطا به نام *خطای نرخ پایه (base-rate fallacy)* مشهور است.

چگونه این معضل را میتوان حل کرد؟ فرض کنید بتوانیم هر دو نتیجه صحیح را به ۹۹/۹٪ برسانیم. یعنی فرض کنید بخواهیم که $Pr[positive/disease] = 0.999$ و $Pr[negative/well] = 0.999$ باشد. با قراردادن این دو عدد در فرمول (۲-۹)، $Pr[well/positive] = 0.09$ بدست می‌آید. بنابراین اگر بتوانیم تشخیص مریض بودن و یا مریض نبودن را با احتمال ۹۹/۹٪ درست پیش‌بینی نمائیم، آنگاه نرخ سیگنال کاذب تنها ۹٪ خواهد بود. این نتیجهٔ خیلی بهتری است ولی هنوز ایده‌آل نیست. بار دیگر دقت ۹۹/۹٪ را در نظر گرفته ولی فرض کنید که احتمال بروز مرض در یک جمعیت تنها ۰/۰۰۰۱ = ۱۰,۰۰۰ : ۱ باشد. در اینصورت نرخ آلارم کاذب ۹۱٪ خواهد شد. در حالات واقعی، [AXWL00] چنین نتیجه‌گیری کردکه احتمالات مرتبط با سیستم‌های تشخیص تهاجم آنچنان‌اند که نرخ آلارم کاذب رضایت‌بخش نمی‌باشد.

فصل ۱۰

نرم افزارهای بداندیش

ین فصل نرم‌افزارهای بداندیش(malware)، علی‌الخصوص ویروس‌ها و کِرم‌ها را مورد بررسی قرار می‌دهد.

۱۰-۱ ویروس‌ها و تهدیدهای مرتبط با آنها

شاید پیچیده‌ترین تهدیدها برای سیستم‌های کامپیوتری، بتوسط برنامه‌هائی صورت می‌پذیرد که از نقاط آسیب‌پذیر این سیستم‌ها سوءاستفاده می‌کنند. در این زمینه هم با برنامه‌های کاربردی و هم با برنامه‌های کمکی همانند ویرایش‌گرها (editors) و کامپایلرها (compilers) سروکار داریم.

این بخش را با مروری بر طیف این تهدیدهای نرم‌افزاری شروع می‌کنیم. بقیهٔ بخش به ویروس‌ها و کِرم‌ها اختصاص داشته و پس از نگاهی به ماهیت آنها، راه‌های مقابله با آنها را بررسی می‌کنیم.

برنامه‌های موذی

بعلت عدم وجود یک اجماع جهانی روی واژه‌های مربوط به این بخش و اختلاط بعضی از گروه‌ها با یکدیگر، تعریف اصطلاحات مربوط به این مقوله دشواری‌هائی را بوجود می‌آورد. جدول ۱۰-۱ که عمدتاً از [SZOR05] اقتباس شده است، راهنمای خوبی در این مورد است.

نرم‌افزارهای بداندیش را می‌توان به دو دسته تقسیم کرد: آنهائی که به یک برنامهٔ میزبان نیاز دارند و آنهائی که بطور مستقل عمل کرده و نیاز به محملی ندارند. دستهٔ اول ضرورتاً بخش‌هائی از یک برنامه بوده و نمی‌توانند مستقل از یک برنامهٔ کاربردی، برنامهٔ کمکی و یا برنامهٔ سیستمی وجود داشته باشند. ویروس‌ها، بمب‌های لاجیک و درب‌های مخفی مثال‌هائی از این دسته‌اند. دستهٔ دوم برنامه‌های کاملی هستندکه می‌توانند مستقلاً بتوسط سیستم عامل کامپیوتر برنامه‌ریزی و اجرا شوند. کِرم‌ها و زامبی‌ها از این مقوله‌اند.

همچنین می‌توان بین آن دسته از تهدیدهای نرم‌افزاری که تکثیر نمی‌شوند در مقابل آنهائی که تکثیر می‌شوند، تفاوت قائل شد. اولی برنامه‌ها و یا تکه‌هائی از برنامه هستند که با یک چاشنی فعال می‌شوند. مثال‌های این مورد، بمب‌های لاجیک، درب‌های مخفی و زامبی‌ها هستند. دستهٔ بعد تکه‌ای از یک برنامه و یا برنامهٔ مستقلی است که وقتی اجرا شود ممکن است کپی‌های متعددی از خود را تولید نموده و در آینده بتوسط همان سیستم و یا سیستم‌های دیگر مرتبط با آن سیستم فعال شوند. ویروس‌ها و کِرم‌ها از این مقوله‌اند.

در بقیهٔ این بخش بطور مختصر به تشریح هریک از این نرم‌افزارهای بداندیش، بجز ویروس‌ها و کِرم‌ها که بطور مستقل تشریح خواهند شد، می‌پردازیم.

جدول ۱-۱۰ واژه‌های مربوط به نرم‌افزارهای بداندیش

نام	توصیف
Virus	خود را به یک برنامه متصل کرده و کپی‌هائی از خود را به برنامه‌های دیگر منتقل می‌کند
Worm	برنامه‌ای که کپی‌های خود را به کامپیوترهای دیگر منتقل می‌کند
Logic Bomb	وقتی فعال می‌شود که پیشامد خاصی روی دهد
Trojan Horse	برنامه‌ای که شامل قابلیت‌های اضافی غیرمنتظره است
Backdoor (trapdoor)	دستکاری یک برنامه بطوری که دست‌یابی غیرمجاز به عملیاتی را امکان‌پذیر نماید
Exploits	کُد مختص به یک آسیب‌پذیری منفرد و یا مجموعه‌ای از آسیب‌پذیری‌ها
Downloaders	برنامه‌ای که اقلام جدیدی را روی ماشین مورد تهاجم نصب می‌کند. یک downloader معمولاً با یک نامهٔ الکترونیک ارسال می‌شود
Auto-rooter	ابزارهای یک نفوذگر بداندیش که از آنها برای ورود به ماشین‌های جدید از راه دور استفاده می‌کند
Kit (virus generator)	مجموعه‌ای از ابزارها برای تولید ویروس‌های جدید بصورت خودکار
Spammer programs	برای ارسال حجم زیادی از هَرزنامه‌های الکترونیک بکار می‌رود
Flooders	برای حمله به شبکه‌های کامپیوتری از طریق ایجاد حجم بالائی از ترافیک بکار می‌رود تا یک حملهٔ انکار سرویس (DoS) را سازمان دهد
Keyloggers	حرکات صفحه کلید در یک کامپیوتر مورد حمله را می‌پاید
Rootkit	مجموعه‌ای از ابزارهای نفوذگری که پس از اینکه نفوذگر به سیستم راه یافت از آنها برای دسترسی به root-level استفاده می‌کند
Zombie	برنامه‌ای که روی یک ماشین آلوده شده فعال می‌شود تا حملات بر روی ماشین‌های دیگر را سامان دهد

درب مخفی (Backdoor)

یک درب مخفی، منفذی سرّی برای ورود به یک برنامه است که به فردی که از وجود این درب مطلع است اجازه می‌دهد که بدون عبور از موانع امنیتی طراحی شده به برنامه راه یابد. درب‌های مخفی برای سالیان متمادی بتوسط برنامه‌نویسان برای تدابیر تِست و اِشکال‌زدائی برنامه‌ها بطور قانونی مورد استفاده قرار گرفته‌اند. تعبیهٔ این درب‌ها معمولاً به این دلیل است که اجرای عادی برنامه نیاز به مراحل تأیید هویت و یا عبور از مراحل وقت‌گیر اجرائی و وارد نمودن مقادیر متعدد دارد. برای اِشکال‌زدائی از برنامه، تولیدکنندهٔ برنامه ممکن است علاقه‌مند به کسب امتیازاتی برای ورود به برنامه و اجتناب از طی همهٔ مراحل اعتبارسنجی لازم باشد. همچنین برنامه نویس ممکن است بخواهد مطمئن شود که در صورت ایجاد اختلال در مراحل تأئید هویت، خود خواهد توانست از طریق دیگری وارد برنامه شده و آن را اصلاح نماید. درب مخفی در حقیقت کُدی برای شناخت دنبالهٔ مخصوصی از ورودی‌ها بوده و می‌تواند یک شمارهٔ کاربری خاص و یا دنبالهٔ غیرمحتملی از وقایع را شامل گردد.

درب‌های مخفی، وقتی‌که برای دسترسی غیرقانونی بتوسط برنامه‌نویس‌های غیرمسئول مورد استفاده قرار گیرند، تهدیدی جدی بحساب می‌آیند. درب مخفی، ایدهٔ اصلی نمایش آسیب‌پذیری در فیلم *War Games* بوده است. مثال دیگری از این تهدید آن بود که در جریان توسعهٔ Multics، تست‌های نفوذ در برنامه بتوسط «تیم ببر» نیروی هوائی امریکا انجام شد (نوعی شبیه‌سازی دشمن). یکی از تکنیک‌های بکار رفته این بود که یک برنامه جعلی بروزرسانی سیستم عامل، برای یکی از سایت‌هائی که از Multics استفاده می‌کرد ارسال شود. این برنامه حاوی یک اسب تروا (بعداً تشریح خواهد شد) بود که می‌توانست از طریق یکی از درب‌های مخفی وارد شده و امکان دسترسی به سیستم را در اختیار تیم عمل‌کننده قرار دهد. این تهدید آنقدر زیرکانه برنامه‌ریزی شده بود که طراحان Multics حتی بعد از اینکه از وجود آن مطلع شدند نتوانستند آن را پیدا نمایند[ENGE80].

ساخت کنترل‌هائی در سیستم عامل برای جلوگیری از سوءاستفاده از درب‌های مخفی کاری مشکل است. معیارهای امنیت بایستی بر توسعهٔ برنامه و فعالیت‌های مربوط بروزرسانی نرم‌افزار، نظارت دقیق داشته باشند.

بمب لاجیک (Logic Bomb)

یکی از قدیمی‌ترین انواع تهدیدهای نرم‌افزاری که قبل از ویروس‌ها و کِرم‌ها وجود داشت و هم اکنون نیز بکار گرفته می‌شود، بمب لاجیک است. بمب لاجیک بصورت یک کُد در یک برنامه کاملاً قانونی تعبیه شده و طوری تنظیم می‌گردد که در صورت حصول شرایط خاصی «منفجر گردد». مثال‌هائی از شرایط خاص که می‌توانند بمب را منفجر سازند، حضور و یا عدم حضور فایل‌های مشخص، روز خاصی از هفته و یا تاریخ مشخصی از سال و یا حتی هنگام استفادهٔ فرد مشخصی از برنامه است. وقتی چاشنی بمب روشن شود، بمب ممکن است بخشی از دیتا و یا تمام فایل‌ها را تغییر داده و یا پاک کند، باعث توقف سیستم شده و یا صدمات دیگری را بجا گذارد. مثال خیره‌کننده‌ای از اینکه چگونه بمب‌های لاجیک می‌توانند عمل کنند مربوط به مورد Tim Lloyd است که با ارسال یک بمب لاجیک، کارفرمای خود یعنی شرکت Omega Engineering را با بیش از ده میلیون دلار خسارت مواجه کرد. عمل او استراتژی رشد سازمان را مختل نموده و نهایتاً به بیکار شدن ۸۰ کارگر انجامید[GAUD00]. Lloyd نهایتاً به ۴۱ ماه زندان و پرداخت دو میلیون دلار غرامت محکوم گردید.

اسب‌های تروا (Trojan Horses)

یک اسب تروا، یک برنامه یا مجموعه‌ای از فرمان‌های مفید و یا ظاهراً مفید است که شامل کُدهای پنهانی بوده و وقتی بکار گرفته شود عمل ناخواسته و یا مضری را انجام دهد.

برنامه‌های اسب تروا می‌توانند بمنظور انجام غیرمستقیم عملی بکار روند که کاربر غیرمجاز نمی‌تواند آن را بصورت مستقیم انجام دهد. بعنوان مثال، برای دست‌یابی به فایل‌های کاربر دیگری در یک سیستم اشتراکی، یک کاربر ممکن است یک برنامهٔ اسب تروا را طوری طراحی نماید که وقتی اجرا شود محدودیت‌های خواندن فایل‌های کاربر دیگر را از بین برده و فایل‌ها بتوسط همهٔ کاربران دیگر قابل دست‌یابی شوند. نویسندهٔ برنامه آنگاه می‌تواند با قراردادن برنامهٔ خود بعنوان یک برنامهٔ مفید در یک فهرست به اشتراک گذاشته شده و دادن نام اغواکننده‌ای به آن، کاربران دیگر را به اجرای برنامه ترغیب نماید. مثالی از این مورد برنامه‌ای است که بتواند بطور واضح لیستی از فایل‌های کاربر را با فرمت دلخواهی تهیه نماید. پس از اینکه کاربر دیگری این برنامه را اجرا نمود، آنگاه نویسندهٔ برنامه خواهد توانست به اطلاعات موجود در فایل او دست یابد. مثالی از یک برنامهٔ اسب تروا که تشخیص آن مشکل خواهد بود، برنامهٔ کامپایلری است که طوری دستکاری شده باشد که در هنگام کامپایل کردن برنامه‌های خاصی همانند برنامهٔ ورود به سیستم، یک کُد را وارد آن نماید[THOM84]. این کُد یک درب

مخفی در برنامهٔ ورود به سیستم(login) ایجاد کرده که به نویسندهٔ برنامه اجازه می‌دهد که با استفاده از کلمهٔ عبور بخصوصی وارد سیستم شود. با خواندن متن اصلی برنامهٔ ورود به سیستم، هرگز نمی‌توان به وجود این اسب تروا پی‌برد.

محرک بسیار معمول دیگر در استفاده از اسب تروا، تخریب داده‌هاست. ظاهراً بنظر خواهد رسید که برنامه کار مفیدی را انجام می‌دهد(مثلاً یک برنامهٔ ماشین حساب)، ولی برنامه در خفا فایل‌های کاربر را پاک می‌کند. بعنوان مثال، یکی از مدیران شبکهٔ CBS با اسب تروائی مورد حمله قرار گرفت که تمام حافظهٔ کامپیوترش را پاک نمود [TIME90]. اسب تروا در یک برنامهٔ گرافیکی که در یک BBS الکترونیک آگهی شده بود قرار داشت.

زامبی (Zombie)

یک زامبی برنامه‌ای است که بطور مخفیانه کامپیوتر دیگری را که به اینترنت وصل است در اختیار گرفته و از طریق آن کامپیوتر حملاتی را انجام دهد که حتی برای خود خلق‌کنندهٔ زامبی نیز دنبال کردن آن دشوار خواهد بود. از زامبی‌ها در حملات انکار سرویس و معمولاً علیه وِب سایت‌ها استفاده می‌شود. زامبی روی صدها کامپیوتر متعلق به افراد غیرمشکوک لانه کرده و آنگاه با ایجاد یک حملهٔ همه جانبه از ترافیک اینترنتی روی وِب سایت هدف، کار آن را مختل می‌سازد. بخش ۳-۱۰ زامبی‌ها را در حوزهٔ حملات انکار سرویس مورد بحث قرار می‌دهد.

ماهیت ویروس‌ها

یک ویروس، یک برنامه است که می‌تواند سایر برنامه‌ها را از طریق دستکاری آنها «آلوده» سازد. در این جرح و تعدیل کپی دیگری از ویروس ایجاد می‌شود که بعداً می‌تواند برنامه‌های دیگر را آلوده کند.

ویروس‌های بیولوژیک، تکه‌های کوچکی از کُدهای ژنتیک هستند- DNA یا RNA - که می‌توانند سازوکار یک سلول زنده را در اختیار گرفته و با دوز و کلک آن را وادار سازند تا هزاران نمونه معیوب از ویروس اولیه را تولید کند. همانند همزاد بیولوژیکی خود، یک ویروس کامپیوتری در کُد خود دستورالعمل ساخت نمونه‌های یکسانی از خود را حمل می‌کند. با استقرار در کامپیوتر میزبان، ویروس کنترل موقت سیستم عامل دیسک(DOS) را در اختیار می‌گیرد. سپس هرگاه این کامپیوتر آلوده با نرم‌افزار غیرآلوده‌ای تماس یابد، یک کپی جدید از ویروس به برنامهٔ جدید منتقل می‌گردد. در نتیجه آلودگی می‌تواند بتوسط کاربران غیرمشکوک که مشغول تبادل برنامه روی یک شبکه هستند از یک کامپیوتر به کامپیوتر دیگر گسترش یابد. در محیط یک شبکه، قابلیت دست‌یابی به کاربردها و سرویس‌های سیستم که روی کامپیوترهای مختلف قرار دارند، استعداد بالائی را برای گسترش یک ویروس به وجود می‌آورد.

یک ویروس می‌تواند هر کاری را که برنامه‌های دیگر انجام می‌دهند، انجام دهد. تنها فرق آن با برنامه‌های دیگر این است که خود را به برنامهٔ دیگری چسبانده و وقتی آن برنامه اجرا می‌شود ویروس نیز مخفیانه کار خود را انجام می‌دهد. وقتی ویروس فعال شود، می‌تواند هر عملی مثل پاک کردن فایل‌ها و برنامه‌ها را انجام دهد.

در طول زمان حیات خود، یک ویروس از چهار فاز مختلف عبور می‌کند:

- **فاز خفتن**: در این فاز ویروس خفته است ولی بالاخره بتوسط واقعه‌ای مانند رسیدن تاریخ مشخصی، حضور برنامه و یا فایل دیگری، و یا عبور ظرفیت دیسک سخت از حد معینی بیدار و فعال می‌شود. همه ویروس‌ها از این مرحله عبور نمی‌کنند.
- **فاز انتشار**: ویروس یک کپی کاملاً مشابه با خود را در برنامه‌های دیگر و یا بخش‌های معینی از دیسک بوجود می‌آورد. هر برنامه آلوده شده خود شامل ویروس مشابه‌ای بوده که می‌تواند وارد فاز انتشار شود.

- **فاز شروع به فعالیت**: ویروس برای انجام عملی که برای آن خلق شده است، فعال می‌شود. همانند فاز خفتن، این فاز نیز می‌تواند بتوسط وقایع متنوعی از قبیل شمارش تعداد دفعاتی که این ویروس شبیه خود را ایجاد کرده است، فعال شود.
- **فاز اجرا**: ویروس کار خود را انجام داده است. نتیجهٔ این عمل ممکن است بدون خطر، همانند ظاهرشدن یک پیام روی مونیتور، و یا خطرناک مانند آسیب زدن و یا تخریب برنامه‌ها و یا فایل‌ها باشد. مثال‌های دیگر از تخریب، اشغال فضای حافظه، ایجاد تناقضات نرم‌افزاری و سخت‌افزاری، و یا رفتار غیرنرمال سیستم است.

بیشتر ویروس‌ها کار خود را طوری انجام می‌دهندکه مختص سیستم عامل خاص و یا در بعضی موارد پایهٔ سخت‌افزاری مشخصی است. بنابراین آنها طوری طراحی می‌شوند که از جزئیات و نقاط ضعف سیستم‌های بخصوص استفاده کنند.

ساختار ویروس

یک ویروس می‌تواند به ابتدای یک برنامهٔ اجرائی و یا انتهای یک برنامهٔ اجرائی وصل گردد و یا ممکن است بطریق دیگری در درون برنامه جای داده شود. نکتهٔ اصلی در این مورد این است که وقتی برنامهٔ آلوده به ویروس بکار گرفته می‌شود، اول کُدهای مربوط به ویروس اجرا شده و سپس کُدهای برنامه اجرا می‌گردند.

یک فرم بسیار عمومی از ساختار ویروس در شکل ۱-۱۰ نشان داده شده است (بر مبنای [COHE94]). در این مورد کُد ویروس V در ابتدای برنامه اجرائی قرار گرفته است و فرض بر این است که نقطهٔ ورود به برنامه، خط اول آن است.

```
        program V :=

{goto main;
      1234567;

          subroutine infect-executable :=
              {loop:
              file := get-random-executable-file;
              if (first-line-of-file = 1234567)
                      then goto loop
                     else prepend V to file; }

          subroutine do-damage :=
              {whatever damage is to be done}

          subroutine trigger-pulled :=
              {return true if some condition holds}

main:       main-program :=
                    {infect-executable;
                    if trigger-pulled then do-damage;
                    goto next;}
next:

 }
```

شکل ۱-۱۰ یک ویروس ساده

یک برنامهٔ آلوده با کُد مربوط به ویروس شروع شده و چنین عمل می‌کند. اولین خط، کُد پرش به برنامهٔ اصلی ویروس است. خط دوم نشانگر خاصی است که بتوسط ویروس مورد استفاده قرار گرفته تا مشخص شود که آیا قربانی مورد توجه قبلاً با این ویروس آلوده شده است یا خیر. وقتی برنامه احضار می‌شود،کنترل بلافاصله به برنامهٔ ویروس منتقل می‌شود. برنامهٔ ویروس به دنبال فایل‌های اجرائی آلوده نشده می‌گردد و آنها را آلوده می‌سازد. سپس ویروس ممکن است عملی را انجام دهد که معمولاً برای سیستم زیان‌آور است. این عمل می‌تواند هربار که برنامه احضار می‌شود صورت پذیرفته و یا یک بمب لاجیک باشد که تنها تحت شرایط خاصی عمل کند. بالاخره ویروس کنترل را به برنامهٔ اولیه منتقل می‌سازد. اگر فاز آلوده‌سازی برنامه بطور معقولی سریع انجام پذیرد، احتمال اینکه کاربر متوجه شود که برنامه آلوده و یا غیرآلوده است کم خواهد بود.

ویروسی همانند آنچه تشریح گردید بسهولت تشخیص داده می‌شود زیرا یک نسخهٔ آلوده از یک برنامه، از یک نسخهٔ آلوده نشده طولانی‌تر است. یک روش برای اینکه از این سهولت تشخیص اجتناب شود این خواهد بود که فایل اجرائی را طوری فشرده نمود که نسخه‌های آلوده و غیرآلوده دارای طول مساوی باشند. شکل ۲-۱۰ [COHE94] منطق لازم را بطور کلی نشان می‌دهد. خطوط کلیدی برنامهٔ ویروس شماره‌گذاری شده و شکل ۳-۱۰ [COHE94] عملیات مربوط را نشان می‌دهد. فرض کنید که برنامهٔ P_1 با ویروس CV آلوده شده است. وقتی این برنامه بکار گرفته می‌شود، کنترل به ویروس منتقل شده و مراحل زیر طی می‌شود:

۱- برای هر فایل آلوده نشده P_2 که پیدا شود، ویروس ابتدا این فایل را فشرده کرده و P'_2 را تولید می‌کند که از برنامهٔ اولیه باندازهٔ برنامهٔ ویروس کوتاه‌تر است.
۲- یک کپی از ویروس اول به برنامهٔ فشرده شده اضافه می‌شود.
۳- نسخهٔ فشرده شدهٔ برنامهٔ آلوده شده اولیه P'_1 از فشردگی خارج می‌گردد.
۴- برنامهٔ اولیهٔ غیرفشرده اجرا می‌شود.

در این مثال، ویروس کاری بجز انتشار انجام نمی‌دهد. همانند مثال قبل، ویروس می‌تواند حاوی یک بمب لاجیک باشد.

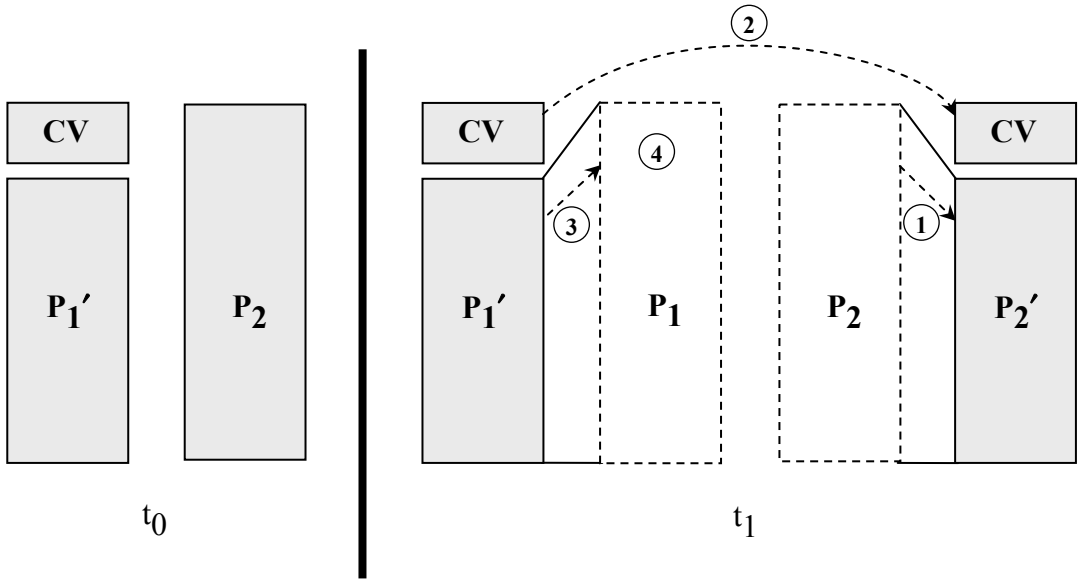
```
        program CV : =

{goto main;
      01234567;

        subroutine infect-executable :=
                {loop:
                        file := get-random-executable-file;
                    if (first-line-of-file = 01234567) then goto loop;
            (1)  compress file;
            (2)  prepend CV to file;
                }

main:  main-program :=
                    {if ask-permission then infect-executable;
            (3)  uncompress rest-of-file;
            (4)  run uncompressed file;}
                }
```

شکل ۲-۱۰ منطق یک ویروس فشرده‌سازی شده



شکل ۳-۱۰ یک ویروس فشرده‌سازی شده

آلودگی اولیه

همینکه یک ویروس با آلوده کردن یک برنامه وارد سیستم شد، در موقعیتی است که می‌تواند بعضی و گاهی تمام فایل‌های دیگر آن سیستم را آلوده نماید. بنابراین برای جلوگیری از آلودگی‌های ویروسی باید از اول از ورود ویروس جلوگیری کرد. متاسفانه پیشگیری کاری بس دشوار است زیرا ویروس‌ها می‌توانند بخشی از هریک از برنامه‌های خارجی باشند. بنابراین مگر اینکه کسی از صفر شروع کرده و تمام برنامه‌های سیستمی و کاربردی را خود بنویسد والا همیشه خطر ویروس وجود دارد.

انواع ویروس ها

از زمانیکه ویروس‌ها برای اولین بار هویدا شدند، جنگ تسلیحاتی بین نویسندگان ویروس و نویسندگان نرم‌افزارهای ضد ویروس دائمی شد. تا برنامهٔ ضد ویروس مؤثری برای مقابله با ویروس‌های موجود آماده می‌گردید، ویروسی از نوع دیگر خلق می‌شد[STEP93]. موارد ذیل را بعنوان چشمگیرترین انواع ویروس معرفی می‌نماید:

- **ویروس انگلی(parasitic)**: این ویروس قدیمی‌ترین و هنوز معمول‌ترین نوع ویروس است. یک ویروس انگلی خود را به فایل‌های اجرائی چسبانده و وقتی آن فایل‌ها اجرا می‌گردند، فایل‌های اجرائی دیگر را پیدا کرده و آنها را آلوده می‌سازد.
- **ویروس ساکن درحافظه(memory-resident)**: بعنوان بخشی از یک برنامهٔ سیستم، در حافظهٔ اصلی کامپیوتر لانه می‌کند. ویروس از این نقطه هر برنامه‌ای را که اجرا می‌شود آلوده می‌سازد.

- **ویروس بخش راه‌اندازی(boot sector)**: یک رکورد و یا بخش اصلی راه‌اندازی را آلوده کرده و وقتی سیستم از روی دیسک مخصوص راه‌انداز که حاوی ویروس است بالا می‌آید، گسترش می‌یابد.
- **ویروس پنهان‌شونده(stealth)**: نوعی از ویروس است که مخصوصاً برای مخفی ماندن از دید نرم‌افزارهای ضد ویروس طراحی شده است.
- **ویروس چندچهره(polymorphic)**: ویروسی است که با هربار آلوده سازی فایل‌ها ظاهر خود را عوض کرده و بنابراین تشخیص آن با مقایسه با یک نمونهٔ قبل، غیرممکن است.
- **ویروس دگردیس(Metamorphic)**: این ویروس نیز همانند ویروس چندچهره در هر بار آلوده‌سازی ظاهر خود را عوض می‌کند. تفاوت در این است که این ویروس در هر بار آلوده‌سازی کاملاً خود را بازنویسی کرده و بنابراین کار تشخیص را دشوارتر می‌کند. ویروس‌های دگردیس ممکن است علاوه بر تغییر دادن چهرهٔ خود، رفتار خود را نیز عوض کنند.

قبلاً به نمونه‌ای از **ویروس پنهان‌شونده** اشاره گردید: ویروسی که از فشرده‌سازی استفاده کرده و حجم برنامه آلوده را دقیقاً به همان اندازهٔ حجم برنامهٔ اولیه نگاه می‌دارد. تکنیک‌های بسیار پیچیده‌تری نیز در مورد ساخت این ویروس ممکن است. بعنوان مثال، یک ویروس می‌تواند منطق حایل شدن در روال I/O دیسک را طوری تغییر دهد که وقتی کوششی برای خواندن بخش‌های مشکوک دیسک که شامل این روال‌هاست صورت می‌پذیرد، ویروس برنامهٔ اولیهٔ غیرآلوده را عرضه نماید.

یک **ویروس چندچهره** ویروسی است که در هنگام تکثیر کپی‌هائی را ایجاد می‌کند که از نظر عمل همانند نوع اولیهٔ آن بوده ولی از نظر ظاهر، الگوی بیت‌های آن تغییر کرده است. همانند یک ویروس پنهان شونده، هدف این ویروس علاوه بر تخریب، شکست دادن برنامه‌هائی است که به دنبال ویروس می‌گردند. در این مورد «امضاء» ویروس در هر کپی متفاوت خواهد بود. برای کسب چنین قابلیتی ویروس ممکن است بطور تصادفی دستوراتی را وارد برنامه کرده و یا نظم دستورالعملهای مستقل را بهم بزند. روش مؤثر دیگری که ممکن است پیش‌گرفته شود، استفاده از رمزنگاری است. قسمتی از ویروس که معمولاً *موتور تغییر (mutation engine)* نامیده می‌شود یک کلید رمز تصادفی را تولید کرده و بتوسط آن بقیهٔ ویروس به رمز در می‌آید. کلید بهمراه ویروس ذخیره شده، و خود موتور جستجو نیز تغییر می‌کند. وقتی یک برنامهٔ آلوده احضار می‌شود، ویروس از کلید پنهان شده استفاده کرده و ویروس را از رمز درمی‌آورد. وقتی ویروس تکثیر شد، یک کلید تصادفی دیگر انتخاب می‌شود.

سلاح دیگری که در زرادخانهٔ نویسندگان ویروس‌ها موجود است، جعبه ابزار تولید ویروس است. چنین جعبه ابزاری یک فرد نسبتاً تازه کار را قادر می‌سازد تا در مدت زمان کوتاهی، تعدادی ویروس خلق نماید. اگرچه ویروس‌هائی که به کمک این جعبه ابزارها خلق می‌شوند پیچیدگی کمتری نسبت به ویروس‌های دست‌ساز دارند، ولی با وجود این تعداد ویروس‌هائی که سریعاً بتوسط آن بوجود می‌آیند خود مشکلی برای روش‌های مبارزه با ویروس خواهد بود.

ویروس‌های ماکرو (Macro Viruses)

در اواسط دههٔ ۱۹۹۰ میلادی، ویروس‌های ماکرو فزاینده‌ترین نوع ویروس‌ها بودند. ویروس‌های ماکرو به دلایل متعددی تهدیدکننده‌های عمده بشمار می‌آیند:

۱- یک ویروس ماکرو مستقل از سیستم عامل کامپیوتر است. تقریباً تمام ویروس‌های ماکرو اسناد Microsoft Word را آلوده می‌سازند. هر پایه سخت‌افزاری و سیستم عاملی که Word را حمایت نماید می‌تواند آلوده شود.

۲- ویروس‌های ماکرو اسناد، و نه بخش‌های قابل اجرای کُد برنامه، را آلوده می‌سازند. بیشتر اطلاعات ورودی یک کامپیوتر را اسناد، و نه برنامه‌ها، تشکیل می‌دهند.
۳- ویروس‌های ماکرو به سهولت گسترش می‌یابند. یکی از روش‌های بسیار معمول از طریق پست الکترونیک است.

ویروس‌های ماکرو از خصیصه‌ای که در برنامهٔ Word و برنامه‌های دیگر Microsoft Office مانندExcel وجود داشته و ماکرو نامیده می‌شود استفاده می‌کنند. یک ماکرو اصالتاً یک برنامهٔ اجرائی است که در یک سند مربوط به Word یا فایل نوع دیگری جاسازی شده است. معمولاً کاربران ماکروها را برای خودکار کردن عملیات تکراری و بنابراین استفادهٔ کمتر از صفحه کلید بکار می‌گیرند. زبان ماکرو معمولاً نوعی از زبان برنامه‌نویسی Basic است. یک کاربر ممکن است دنباله‌ای از حرکات کلیدها را در یک ماکرو تعریف نموده و آن را طوری تنظیم نماید که وقتی یک کلید عملیاتی و یا ترکیب کوتاهی از چند کلید بکار گرفته می‌شوند، برنامهٔ ماکرو اجرا گردد..

نسخه‌های پی‌درپی Word که بعداً به بازار آمده‌اند، حفاظت بیشتری را در برابر ویروس‌های ماکرو فراهم نموده‌اند. بعنوان مثال مایکروسافت یک ابزار اختیاری Macro Virus Protection عرضه نموده است که فایل‌های مشکوک Word را تشخیص داده و به مشتری خطر جدی بازکردن یک فایل شامل ماکرو را گوشزد می‌کند. سازندگان محصولات متنوع ضد ویروس نیز ابزارهائی را برای تشخیص و تصحیح ویروس‌های ماکرو تولید کرده‌اند. همانند سایر انواع ویروس‌ها، مسابقهٔ تسلیحاتی در زمینهٔ ویروس‌های ماکرو نیز جریان دارد ولی آنها دیگر جزو تهدیدکننده‌های اصلی به شمار نمی‌آیند.

ویروس‌های پست الکترونیک (E-mail Viruses)

نوآوری جدید دیگر در مقولهٔ نرم‌افزارهای بداندیش، بتوسط ویروس پست الکترونیک حاصل گردید. اولین ویروس‌های پست الکترونیک که بسرعت تکثیر می‌شدند، همانند Melissa، از یک ماکروی Microsoft Word پنهانی در یک فایل پیوست به یک نامهٔ الکترونیک بهره می‌گرفتند. اگر دریافت‌کنندهٔ نامه، پیوست نامه را باز نماید ماکروی Word فعال می‌شود. آنگاه:

۱- ویروس پست الکترونیک، خود را برای کلیهٔ کسانی که نامشان در لیست آدرس کاربر بازکنندهٔ نامه موجود باشد می‌فرستد.
۲- ویروس تخریب محلی انجام می‌دهد.

در پایان سال ۱۹۹۹میلادی، نسخهٔ قدرتمندتری از ویروس پست الکترونیک ظاهرگردید. این نسخهٔ جدیدترمی‌توانست صرفاً با بازکردن یک نامهٔ الکترونیک حاوی ویروس فعال گردد و نیازی به بازکردن پیوست نامه نمی‌بود (مثل ویروس I Love You). این ویروس از دستورالعمل‌های زبان برنامه‌نویسی Visual Basic که تسهیلات پست الکترونیک را فراهم می‌سازند استفاده می‌کند.

بنابراین نسل جدیدی از بَدافزارها را مشاهده می‌کنیم که از طریق نامهٔ الکترونیک وارد شده و از خصوصیات نرم‌افزاری پست الکترونیک استفاده کرده تا خود را در سراسر اینترنت گسترش دهند. ویروس به محض فعال شدن به انتشار خود می‌پردازد (چه با بازکردن e-mail و چه با بازکردن پیوست e-mail) و خود را به تمام آدرس‌هائی که در صندوق پستی میزبان آلوده قرار دارد ارسال می‌کند. در نتیجه در حالی که قبلاً ماه‌ها و یا سال‌ها طول می‌کشید تا یک ویروس منتشر گردد، حال در ظرف مدت کوتاهی ویروس به همه جا راه می‌یابد. این موضوع کار نرم‌افزارهای ضدویروس را بسیار مشکل می‌کند تا بتوانند قبل از اینکه ویروس لطمات زیادی به محدودهٔ بزرگی وارد نماید، آن را کشف و نابود کنند. بالاخره، درجهٔ بالاتری از امنیت بایستی در نرم‌افزارهای کمکی و کاربردی اینترنتی روی کامپیوترهای شخصی تعبیه شود تا با این تهدید فزاینده مبارزه نمایند.

کِرم‌ها (Worms)

یک کِرم یک برنامه است که می‌تواند خود را تکثیر کرده و کپی‌های تکثیرشده را در عرض یک شبکهٔ کامپیوتری از رایانه‌ای به رایانهٔ دیگر منتقل نماید. پس از ورود به یک سیستم، کِرم ممکن است فعال شده، شروع به تکثیر نموده و مجدداً انتشار یابد. علاوه بر انتشار، کِرم معمولاً کارهای ناخواسته‌ای را نیز انجام می‌دهد. یک ویروس پست الکترونیک بعضی از مشخصات یک کِرم را داراست زیرا خود را از یک سیستم به سیستم دیگر منتقل می‌سازد. ولی با وجود این ما آن را یک ویروس می‌گوئیم زیرا برای انتقال آن واسطهٔ انسانی لازم است. یک کِرم بطور فعال به دنبال آلوده کردن سیستم‌های دیگر است و هر ماشینی که آلوده می‌شود خود بعنوان یک پایگاه خودکار برای حمله به ماشین‌های دیگر بکار می‌رود.

برنامه‌های مربوط به کِرم‌ها، از ارتباطات شبکه‌ای برای انتقال از یک سیستم به سیستم دیگر استفاده می‌کنند. همینکه یک کِرم شبکه در یک سیستم فعال گردید، می‌تواند بصورت یک ویروس یا باکتری عمل نموده، یا یک اسب تروا را در سیستم وارد کرده و یا به هر میزان عملیات تخریبی و یا قطع سرویس انجام دهد.

برای تکثیر خود، یک کِرم شبکه از برخی قابلیت‌های شبکه استفاده می‌کند. مثال‌هائی از این مورد چنین‌اند:

- **تسهیلات پست الکترونیک**: یک کِرم کپی خود را به سایر سیستم‌ها پست می‌کند.
- **قابلیت اجرا در دوردست**: یک کِرم یک کپی خود را در سیستم دیگر اجرا می‌کند.
- **قابلیت ورود به سیستم در دوردست**: یک کِرم بعنوان یک کاربر در یک سیستم دوردست وارد شده و سپس فرامینی را بکار می‌گیرد که خود را از یک سیستم به سیستم دیگر کپی نماید.

کپی جدید برنامهٔ کِرم آنگاه روی سیستم دوردست اجرا شده که علاوه بر عملیاتی که درآن سیستم انجام می‌دهد، بهمان ترتیب قبل خود را گسترش هم می‌دهد.

یک کِرم شبکه همان خصوصیات یک ویروس کامپیوتری را از خود نشان می‌دهد: یک فاز خفته، یک فاز انتشار، یک فاز شروع به فعالیت و یک فاز اجرا. فاز انتشار معمولاً عملیات زیر را انجام می‌دهد:

۱- با بررسی جداول کامپیوتر میزبان و یا سایر آدرس‌های موجود در فایل‌های سیستم دوردست، به دنبال سیستم‌هائی می‌گردد که آنها را آلوده سازد.
۲- یک اتصال با سیستم دوردست برقرار می‌کند.
۳- خود را در سیستم دوردست کپی کرده و ترتیبی اتخاذ می‌کند که کپی اجرا شود.

کِرم شبکه همچنین ممکن است قبل از آلوده‌کردن سیستم تلاش کند تا بفهمد که آیا سیستم قبلاً آلوده شده است یا خیر؟ در یک سیستم با برنامه‌های متعدد، کِرم ممکن است با پوشیدن لباس مبدّل خود را بجای یک پردازش و یا نام دیگری که برای اپراتور سیستم آشنا باشد جا بزند.

همانند ویروس‌ها، مبارزه با کِرم‌های شبکه کاری دشوار است.

کِرم موریس (Morris)

تا تولید نسل جدید کِرم‌ها، مشهورترین کِرم شناخته شده کِرمی بود که در سال ۱۹۸۸ میلادی بتوسط Robert Morris در اینترنت رها گردید. کِرم Morris برای گسترش در سیستم‌های UNIX طراحی شده بود و تکنیک‌های متفاوتی برای انتشار را بکار می‌بست. وقتی یک نسخهٔ آن اجرا می‌گردید، اولین وظیفهٔ آن شناسائی میزبان‌های مرتبط با میزبان

جاری بود که ورود به خود از طرف میزبان جاری را اجازه می‌دادند. کِرم این عمل را با آزمایش لیست‌ها و جداول متعددی که شامل جداول سیستم‌ها که مشخص می‌نماید کدام ماشین‌ها طرف اعتماد سیستم جاری بوده، فایل‌های پستی کاربر برای انتقال نامه‌های الکترونیک، جداولی که کاربران برای دسترسی به حساب‌های دوردست بکار می‌برند، و همچنین برنامه‌ای که اتصالات شبکه را نشان می‌دهد، شروع می‌کرد. برای هر میزبان جدید کشف شده، کِرم روش‌های متعددی برای کسب دست‌یابی را امتحان می‌نمود:

۱- سعی میکرد تا بعنوان یک کاربر قانونی به میزبان دور وارد شود. در این روش، کِرم اول تلاش می‌نمود تا فایل کلمه عبور محلی را شکسته (cracking) و سپس از ID و کلمهٔ عبور کشف شده استفاده کند. فرض براین بود که کاربران متعددی همین کلمهٔ عبور را در سیستم‌های مختلف بکار می‌برند. برای بدست آوردن کلمات عبور، کِرم یک برنامهٔ شکستن کلمهٔ عبور را اجرا می‌نمود که شامل مراحل زیر بود:
(الف) نام حساب شخص و همهٔ جایگشت‌های آن را امتحان می‌کرد.
(ب) یک لیست داخلی ۴۳۲- تائی از کلمات عبور که Morris آنها را محتمل می‌دانست آزمایش می‌شد.
(ج) تمام کلمات موجود در لغت‌نامهٔ محلی سیستم امتحان می‌شد.

۲- از یک اِشکال در پروتکل finger سوءاستفاده کرده تا محل یک کاربر دور را حدس بزند.

۳- از یک درب مخفی در گزینهٔ اشکال‌زدائی پردازش دور، که نامه‌ها را ارسال و دریافت می‌نماید سوءاستفاده می‌کرد.

اگر هریک از عملیات فوق به موفقیت می‌انجامید، کِرم ارتباط با مترجم فرامین سیستم عامل را حاصل می‌نمود. سپس کِرم به این مترجم یک برنامهٔ bootstap ارسال کرده، فرمانی برای اجرای این برنامه صادر کرده و از سیستم خارج می‌شد. برنامهٔ bootstrap در حین اجرا، برنامهٔ مادر را صدا زده و بقیه کِرم را پیاده می‌کرد. کِرم جدید سپس اجرا می‌گردید.

حملات جدید کِرم‌ها

دورهٔ مدرن تهدید کِرم‌ها با رها شدن کِرم Code Red در ماه ژوئیهٔ سال ۲۰۰۱ میلادی آغاز شد. Code Red از یک حفرهٔ امنیتی در IIS (Microsoft Internet Information Server) برای نفوذ و گسترش سوءاستفاده می‌کند. این کِرم همچنین کنترل‌کنندهٔ فایل‌های سیستمی در Windows را غیرفعال می‌کند. کِرم بصورت تصادفی از آدرس‌های IP استفاده کرده تا میزبان‌های دیگر را آلوده سازد. در خلال دورهٔ زمانی مشخصی، کِرم فقط انتشار می‌یابد. سپس با بمباران کردن یک وِب‌سایت با بسته‌های دیتا از میزبان‌های مختلف، یک حملهٔ انکار سرویس (Denial of Service) را آغاز می‌نماید. کِرم آنگاه فعالیت خود را به حال تعلیق درآورده ولی بطور تناوبی فعال می‌شود. در موج دوم حملات، Code Red تقریباً ۳۶۰,۰۰۰ سِرور را در ظرف ۲۴ ساعت آلوده نمود. علاوه بر فاجعه‌ای که برای سِرور هدف ایجاد می‌کند، Code Red می‌تواند مقادیر حجیمی از ظرفیت اینترنت را در اختیار گرفته و سرویس را مختل سازد.

Code Red II نوع دیگری از این کِرم است که Microsoft IIS هدف آن است. علاوه براین، کِرم جدید یک درب مخفی در کامپیوتر قربانی ایجاد کرده که به یک هَکر اجازه می‌دهد تا فعالیت‌های این کامپیوتر را هدایت نماید.

در اواخر سال ۲۰۰۱، کِرم دیگری با قابلیت‌های متنوع بنام Nimda هویدا گردید. Nimda برای گسترش از سازوکارهای متعددی استفاده می‌کند:

- از یک کلاینت به کلاینت دیگر، از طریق پست الکترونیک.
- از یک کلاینت به کلاینت دیگر، از طریق قابلیت‌های اشتراکی یک شبکهٔ باز.
- از یک سِرور وِب به کلاینت، از طریق مرورکردن سایت‌های مورد حمله قرارگرفته.
- از یک کلاینت به سِرور وِب، از طریق اِسکن کردن فعال و سوءاستفاده از نقاط آسیب‌پذیر فهرست‌های Microsoft IIS 4.0/5.0.
- از یک کلاینت به سِرور وِب، ازطریق جستجو برای درب‌های مخفی بجامانده بتوسط کِرم‌های "Code Red II".

این کِرم، اسناد وِب (مثل فایل‌های با پسوند htm.، html. و asp.) و فایل‌های اجرائی دیگر در سیستم آلوده شده را تغییر داده و نسخه‌های متعددی از خود تحت نام‌های متفاوت را خلق می‌کند.

در اوائل سال ۲۰۰۳، کِرم SQL Slammer ظاهر گردید. این کِرم از نقطهٔ ضعف سرریزشدن حافظهٔ موقت سِرور Microsoft SQL سوءاستفاده می‌کرد. Slammer بطور فوق‌العاده‌ای متراکم بود و بسرعت گسترش می‌یافت بطوری که در ظرف ده دقیقه ۹۰٪ میزبان‌های آسیب‌پذیر را آلوده می‌کرد. پایان سال ۲۰۰۳ شاهد ظهور کِرم Sobig.f بود که از سِرورهای باز پروکسی سوءاستفاده کرده و آنها را به پایگاهی برای ارسال هَرزنامه تبدیل می‌کرد. در اوج فعالیت خود، Sobig.f برابر گزارش‌های اعلام شده، عامل ارسال یک پیام در هر ۱۷ پیام بود و در اولین ۲۴ ساعت حضور، یک میلیون کپی از خود برجای گذاشت.

Mydoom یک کِرم پرحجم پست الکترونیک بود که در سال ۲۰۰۴ ظاهر گردید. این کرم از شگرد فزایندهٔ ایجاد یک درب مخفی در کامپیوترهای آلوده شده استفاده می‌کرد که به هَکرها اجازه می‌داد تا از این طریق به داده‌های مهمی همچون کلمات عبور و شمارهٔ کارت‌های اعتباری دست یابند. Mydoom تا هزاربار در هر دقیقه تکثیر می‌شد و برابر گزارشات، اینترنت را با ارسال ۱۰۰ میلیون پیام در عرض ۳۶ ساعت آلوده کرد.

وضعیت تکنولوژی کِرم‌ها

وضعیت فعلی تکنولوژی کِرم‌ها شامل موارد زیر است:

- **حمله به سیستم عامل‌های مختلف**: کِرم‌های جدیدتر منحصر به رایانه‌هائی با سیستم عامل Windows نبوده بلکه می‌توانند به سیستم عامل‌های متعددی حمله نمایند که از آن جمله انواع UNIX می‌باشد.
- **استثمار چندگانه**: کِرم‌های جدید به طرق مختلف به سیستم‌ها نفوذ کرده و از سِرورهای وِب، مرورگرها، پست الکترونیک، به اشتراک گذاشتن فایل‌ها و سایر کاربردهای مبتنی بر شبکه سوءاستفاده می‌کنند.
- **گسترش فوق سریع**: یکی از روش‌هائی که برای سرعت بخشیدن به گسترش کِرم از آن استفاده می‌شود یک پیش اِسکن اینترنت برای جمع‌آوری آدرس ماشین‌های آسیب‌پذیر است.
- **چندچهرگی**: برای فرار از کشف شدن، عبور از فیلترها و خنثی کردن تحلیل‌های برخط، کِرم‌ها از تکنیک ویروس‌های چندچهره استفاده می‌کنند. هر کپی کِرم یک کُد جدید ایجاد کرده که از نظر عملکرد دارای فرامین مشابه و تکنیک‌های رمزنگاری یکسان هستند.
- **دگردیسی**: علاوه بر عوض کردن ظاهر خود، کِرم‌های دگردیس دارای یک فهرست از الگوهای رفتاری متفاوت‌اند که در مراحل مختلف انتشار از آنها استفاده می‌کنند.

- **وسیلهٔ حمل و نقل**: چون کِرم‌ها می‌توانند به سرعت سیستم‌های زیادی را آلوده نمایند، اغلب محمل خطرناکی برای حمل سایر ابزارهای حملات گسترده از قبیل زامبی‌ها در حملات گستردهٔ انکار سرویس هستند.
- **استثمار غافلگیرانه**: برای گسترش ماکزیمم و ایجاد حداکثر غافل‌گیری، یک کِرم باید از یک آسیب‌پذیری ناشناخته که تنها امکان کشف آن در یک محیط عمومی شبکه امکان‌پذیر است استفاده کند.

۱۰-۲ روش‌های مقابله با ویروس‌ها

بکارگیری آنتی‌ویروس‌ها

راه حل ایده‌آل در برابر تهدید ویروس‌ها، جلوگیری از ورود آنهاست: در وهلهٔ اول به ویروس اجازه ندهید که وارد سیستم شود. دست‌یابی به این هدف معمولاً غیرممکن است ولی پیش‌گیری، حملات موفق ویروس‌ها را کاهش خواهد داد. بهترین عمل بعدی انجام کارهای زیر است:

- **تشخیص**: وقتی ویروس سیستمی را آلوده کرد، از این اتفاق آگاه شوید و محل ویروس را کشف کنید.
- **شناسائی**: وقتی مشخص شد که ویروسی وجود دارد، نوع ویروس را تعیین کنید.
- **پاک‌سازی**: وقتی ویروس شناسائی گردید، همهٔ آثار ویروس را از برنامهٔ آلوده شده پاک کرده و برنامه را بحالت اولیه برگردانید. ویروس را از تمام سیستم‌های آلوده شده طوری برانید که آلودگی به جاهای دیگر سرایت نکند.

اگر حضور ویروس تشخیص داده شد ولی شناسائی و یا پاک‌سازی آن میسر نگردید، آنگاه چارهٔ امر آن است که برنامهٔ آلوده را نابود کرده و یک نسخهٔ جدید و پاک را جانشین آن نمائیم.

تکنولوژی ساخت ویروس‌ها و آنتی‌ویروس‌ها دست در دست هم بجلو می‌روند. ویروس‌های اولیه، کُدهای نسبتاً ساده‌ای بودند و می‌توانستند با بسته‌های نرم‌افزاری ضدویروس نسبتاً ساده شناسائی و دفع شوند. همینطور که جنگ تسلیحاتی ویروس‌ها تکامل یافت، هم ویروس‌ها و هم الزاماً آنتی ویروس‌ها تکامل‌یافته‌تر و پیچیده‌تر شدند.

[STEP93] چهار نسل از نرم‌افزارهای ضدویروس را شناسائی نموده است:

- نسل اول: اِسکنرهای ساده
- نسل دوم: اِسکنرهای کشف‌کننده
- نسل سوم: دام‌های شکارکنندهٔ فعالیت
- نسل چهارم: محافظت تمام عیار

یک اِسکنر **نسل اول** برای شناسائی یک ویروس، نیاز به امضاء آن ویروس دارد. ویروس ممکن است شامل "wildcard" بوده ولی الزاماً دارای همان ساختار و همان پترن بیت‌ها در همهٔ نسخ خود است. چنین اِسکنرهای مخصوص امضاء، فقط قابلیت تشخیص ویروس‌های شناخته شده را داشتند. نوع دیگری از اِسکنرهای نسل اول سابقه‌ای از اندازهٔ برنامه را نگاه داشته و به دنبال تغییر اندازه برنامه می‌گردند.

یک اِسکنر **نسل دوم** متکی بر امضاء خاصی نیست. در عوض اِسکنر از قوانین ذهنی کشف‌کننده استفاده کرده و بدنبال آلودگی‌های محتمل ویروس می‌گردد. یک نوع از این اِسکنرها به جستجوی کُدهائی می‌پردازد که معمولاً با ویروس‌ها مرتبط می‌باشند. بعنوان مثال، یک اِسکنر ممکن است به جستجوی ابتدای یک حلقهٔ رمزنگاری که در ویروس چندچهره مورد استفاده

قرار می‌گیرد پرداخته و کلید رمز را پیدا کند. همینکه کلید کشف شد، اِسکنر برای شناسائی ویروس آن را رمزگشائی نموده و سپس آلودگی را برطرف کرده و برنامه را اصلاح می‌کند.

روش دیگری در نسل دوم آنتی‌ویروس‌ها، کنترل صحت برنامه است. یک جمع کنترلی می‌تواند به هر برنامه الصاق شود. اگر ویروسی برنامه را بدون تغییردادن جمع کنترلی آلوده نماید، آنگاه یک آزمایش کنترل صحت، تغییر را آشکار می‌سازد. برای مقابله با ویروسی که آنقدر پیچیده باشد که بتواند در هنگام آلوده‌سازی برنامه، جمع کنترلی آن را تغییر دهد می‌توان از یک تابع درهم‌ساز رمزشده استفاده کرد. کلید رمزنگاری در جائی جدا از برنامه ذخیره شده تا ویروس نتواند یک کُد جدید hash تولید کرده و آن را به رمز درآورد. با استفاده از یک تابع درهم‌ساز بجای یک جمع کنترلی ساده، ویروس از جرح و تعدیل برنامه بنحوی که بتواند همان کُد hash سابق را تولید کند، عاجز می‌ماند.

نسل سوم آنتی‌ویروس‌ها، برنامه‌های ساکن در حافظه هستند که ویروس را با فعالیت آن، و نه با ساختار آن، در یک برنامهٔ آلوده شده شناسائی می‌کنند. حسن چنین برنامه‌هائی این است که لازم نیست تا امضاءها و قواعد ذهنی برای ردیف وسیعی از ویروس‌ها را تولید کرد، بلکه تنها کافی است که مجموعهٔ کوچکی از فعالیت‌ها که نمایشگر تلاش برای آلوده‌سازی سیستم است را شناسائی نموده و سپس برای پاک‌سازی مداخله کرد.

محصولات **نسل چهارم**، بسته‌های نرم‌افزاری شامل مجموعه‌ای از آنتی‌ویروس‌ها بوده که به همراه هم مورد استفاده قرار می‌گیرند. اینها شامل اِسکنرها و دام‌های شکارکننده فعالیت می‌باشند. بعلاوه چنین بسته‌ای شامل قابلیت‌های کنترل دست‌یابی هستند که توان ویروس‌ها در ورود به یک سیستم را کم کرده و آنگاه قابلیت یک ویروس برای بروزرسانی فایل‌ها برای گسترش آلودگی را محدود می‌سازند.

جنگ تسلیحاتی ادامه دارد. با بسته‌های نرم‌افزاری آنتی‌ویروس‌های نسل چهارم، استراتژی دفاعی سازمان یافته‌تری بکار گرفته می‌شود و حوزهٔ دفاع به معیارهای عام‌تری از امنیت کامپیوتر اعمال می‌گردد.

تکنیک‌های پیشرفتهٔ آنتی‌ویروس‌ها

در زمینهٔ آنتی‌ویروس‌ها، مرتباً روش‌های پیچیده‌تر و محصولات پیشرفته‌تری پدیدار می‌گردند. در اینجا به دو نمونه از مهم‌ترین آنها اشاره می‌کنیم.

رمزگشائی ژنریک

تکنولوژی رمزگشائی ژنریک (Generic Decryption) GD، برنامه‌های آنتی‌ویروس را قادر می‌سازد که حتی پیچیده‌ترین ویروس‌های چندچهره را تشخیص داده و در عین حال سرعت اِسکن نمودن برای یافتن ویروس‌ها را بالا نگاه دارند[NOCH97]. بیاد آورید‌که وقتی فایلی که حاوی یک ویروس چندچهره است اجرا می‌شود، ویروس بایستی خود را رمزگشائی کرده و فعال شود. برای تشخیص چنین ساختاری، فایل‌های اجرائی از یک اسکنر GD که شامل عناصر زیر است استفاده می‌کنند:

- **مقلّد پردازشگر مرکزی (CPU Emulator)**: یک کامپیوتر مجازی مبتنی بر نرم‌افزار است. فرامین مربوط به یک فایل اجرائی بجای اجرا روی پردازشگر اصلی بتوسط این مقلّد ترجمه می‌گردند. مقلّد شامل نسخه‌های نرم‌افزاری همهٔ رجیسترها و سایر بخش‌های سخت‌افزاری پردازشگر می‌باشد بطوری که پردازشگر اصلی تحت تاثیر برنامه‌هائی که بتوسط مقلّد مورد تعبیر قرار می‌گیرند واقع نمی‌شود.

- **اسکنر امضاء ویروس (Virus signature scanner)**: مدولی که کُد برنامهٔ هدف، بمنظور یافتن امضاء ویروس‌های شناخته شده، را اسکن می‌کند.
- **مدول کنترل مقلّد (Emulation control module)**: اجرای کُد برنامهٔ هدف را کنترل می‌کند.

در ابتدای هر شبیه‌سازی، مقلّد شروع به ترجمهٔ خط به خط فرامین برنامهٔ هدف می‌کند. در نتیجه اگر کُد برنامه شامل یک بخش رمزگشائی که ویروس را رمزگشائی و آشکار می‌کند باشد، این کُد ترجمه می‌گردد. در واقع ویروس کار برنامهٔ آنتی‌ویروس را با ظاهر کردن ویروس انجام می‌دهد. هرچند وقت یکبار نیز، مدول کنترل عمل ترجمه را متوقف نموده و برنام را بمنظور یافتن امضاء ویروس‌ها اسکن می‌نماید.

در هنگام ترجمه، کُد برنامهٔ هدف نمی‌تواند هیچ آسیبی به محیط حقیقی کامپیوتر شخصی وارد کند زیرا برنامه در یک محیط کاملاً کنترل شده ترجمه می‌شود.

مشکل‌ترین مقولهٔ طراحی در استفاده از یک اسکنر GD، تعیین زمان لازم برای ترجمه است. معمولاً عناصر یک ویروس در فاصلهٔ کوتاهی پس از شروع اجرای یک برنامه فعال می‌شوند، ولی الزاماً اینطور نیست. هرچقدر زمان تقلید یک برنامهٔ بخصوص بتوسط اسکنر زیادتر باشد، احتمال شکار یک ویروس مخفی زیادتر است. ولی برنامهٔ آنتی‌ویروس نمی‌تواند زمان و منابع زیادی را برای مدت طولانی در اختیار گیرد زیرا در اینصورت کاربر از تأخیر طولانی شاکی خواهد شد.

سیستم مصون دیجیتال (Digital Immune System)

سیستم مصون دیجیتال، یک سیستم حفاظتی مفصّل ضد ویروس است که بتوسط IBM طراحی شده است [KEPH97a, KEPH97b]. محرک تولید این سیستم، تهدید فزایندهٔ انتشار ویروس‌های اینترنتی بوده است. ابتدا مختصری راجع به این نوع تهدید صحبت کرده و آنگاه روش IBM را تشریح می‌کنیم.

در قدیم، تهدید ویروس‌ها دارای این مشخصهٔ بود که گسترش ویروس‌های جدید و تغییر شکل آنها کند بود. نرم‌افزارهای آنتی‌ویروس معمولاً بصورت ماهیانه به روز درآمده و این امر برای کنترل مساله کافی بود. همچنین در فرم سنّتی، اینترنت نقش نسبتاً کوچکی در گسترش ویروس‌ها را ایفاد می‌نمود. اما همانطور که [CHES97] خاطرنشان می‌سازد، دو روند عمده در تکنولوژی اینترنت اثر فزاینده‌ای در سرعت گسترش ویروس در سال‌های اخیر داشته است:

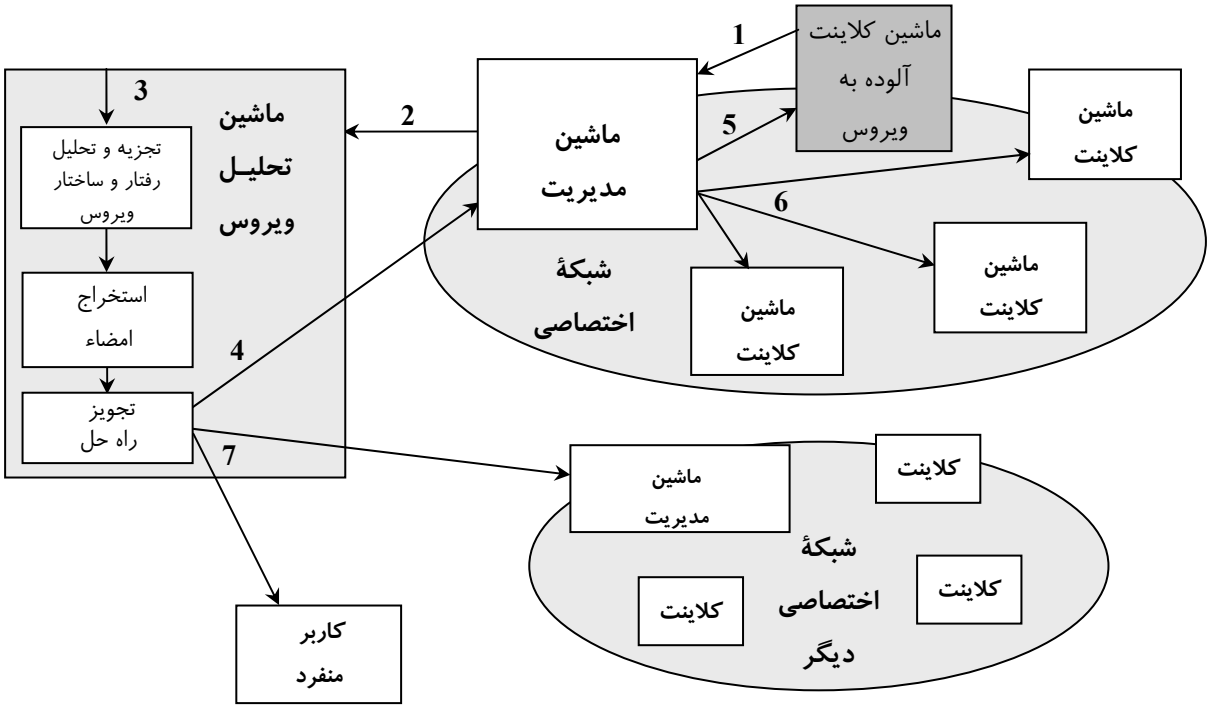
- **سیستم‌های یکپارچهٔ پستی**: سیستم‌هائی همانند Lotus Notes و Microsoft Outlook امر ارسال هر چیزی به هرکسی، و کار با اقلام دریافت شده را بسیار آسان کرده است.
- **سیستم‌های برنامه‌های سیار**: قابلیت‌هائی همچون Java و Active X به برنامه‌ها اجازه می‌دهد تا بخودی‌خود از یک سیستم به سیستم دیگر عبور نمایند.

در پاسخ به تهدیدهائی که بتوسط قابلیت‌های جدید اینترنت حاصل شده‌اند، IBM یک سیستم مصون دیجیتال مربوط به خود را تولید کرده است. این سیستم، استفاده از مقلّد برنامه که در قسمت قبل از آن یاد شد را توسعه داده و یک سیستم مقلّد و تشخیص ویروس را ساخته است. هدف این سیستم، عکس‌العمل سریع و ایزوله کردن ویروس‌ها بمحض ورود به سیستم است. همینکه یک ویروس جدید وارد سازمانی شود، سیستم مصون بطور خودکار آن را محاصره کرده، تجزیه و تحلیل نموده، تشخیص داده، محاصره کرده، آن را از بین برده و اطلاعات مربوط به ویروس را به سیستم‌هائی که از آنتی‌ویروس IBM استفاده می‌کنند ارسال کرده تا ویروس قبل از اینکه بتواند در جای دیگری اجرا شود، کشف و خنثی گردد.

شکل ۴-۱۰ مراحل اصلی عملیات در سیستم مصون دیجیتال را نشان می‌دهد:

۱- یک برنامهٔ پایشگر روی هر PC، از یک سری ذهنیات در رابطه با رفتار سیستم، تغییرات مشکوک برنامه‌ها، و یا امضاء محلی در مورد اینکه ممکن است ویروسی وجود داشته باشد استفاده می‌کند. برنامهٔ پایشگر یک نسخه از هر برنامه‌ای را که فکر می‌کند آلوده است به یک ماشین مدیریت در سازمان ارسال می‌کند.
۲- ماشین مدیریت، نسخهٔ برنامه را بصورت رمز درآورده و آن را برای یک ماشین مرکزی تجزیه و تحلیل ویروس ارسال می‌دارد.
۳- این ماشین محیطی را ایجاد می‌کند که در آن برنامهٔ آلوده می‌تواند با امنیت کامل اجرا شود. روش‌های اجرائی در این مورد شامل تقلید و یا خلق محیط حفاظت شده‌ای است که در آن محیط، برنامهٔ مشکوک بتواند اجرا و پایش گردد. ماشین تجزیه و تحلیل ویروس آنگاه داروئی برای شناسائی و حذف ویروس تجویز می‌کند.
۴- داروی تجویز شده به ماشین مدیریت برگشت داده می‌شود.
۵- ماشین مدیریت، داروی تجویز شده را به کلاینت می‌دهد.
۶- داروی تجویز شده همچنین برای سایر کلاینت‌های سازمان ارسال می‌گردد.
۷- مشترکین سراسر دنیا بطور منظم آنتی‌ویروس‌های جدید را دریافت می‌کنند تا آنها را از ویروس‌های جدید حفظ کنند.

موفقیت سیستم مصون دیجیتال وابسته به توانائی ماشین تجزیه و تحلیل ویروس در تشخیص ویروس‌های جدید و ابتکاری می‌باشد. با تحلیل مداوم و پائیدن ویروس‌هائی که در محیط پیدا می‌شوند بایستی بتوان نرم‌افزار مصون دیجیتال را بصورت دائم برای مبارزه با تهدیدها به روز نگاه داشت.



شکل ۴-۱۰ سیستم مصون دیجیتال

نرم افزار سدکنندهٔ رفتار (Behavior-Blocking Software)

بر خلاف اِسکنرهائی که به تاریخچهٔ ویروس‌ها و یا بعبارتی به جستجوی اثر انگشت ویروس‌ها می‌پردازند، نرم‌افزار سدکنندهٔ رفتار با سیستم عامل یک کامپیوتر میزبان جفت شده و رفتار برنامه را بمنظور کشف رفتارهای بداندیشانه بصورت بلادرنگ می‌پاید. نرم‌افزار سدکنندهٔ رفتار آنگاه جلوی رفتارهای موذیانه را قبل از اینکه بتوانند روی سیستم تأثیر منفی گذارند، سد می‌کند. رفتارهای پایش شده می‌تواند شامل موارد زیر باشد:

- تلاش برای بازکردن، مشاهده، حذف و یا تعویض فایل‌ها.
- تلاش برای فرمت کردن درایوها و سایر عملیات غیرقابل برگشت.
- ایجاد تغییرات در منطق فایل‌های اجرائی برنامه‌های ماکرو.
- تغییرات در تنظیمات حیاتی سیستم مانند تغییر در تنظیمات بالاآمدن کامپیوتر.
- تغییرات در نامهٔ الکترونیک و پیام‌های فوری کلاینت.
- تحریک به شروع ارتباطات شبکه‌ای.

اگر سدکنندهٔ رفتار تشخیص دهد که اجرای یک برنامه باعث بروز رفتارهای موذیانه خواهد شد، می‌تواند این رفتارها را در هنگام اجرا مسدود کرده و یا نرم‌افزار بداندیش را خاتمه دهد. این عمل رجحان عمده‌ای نسبت به روش‌های تشخیص آنتی‌ویروس‌های مبتنی بر اثر انگشت و یا سوابق ویروس‌ها دارد. در حالی که عملاً میلیاردها روش مختلف برای سردرگم‌کردن و تغییر دستورات در یک ویروس یا کِرم وجود دارد، که بسیاری از آنها نمی‌توانند از طرف یک اِسکنر عمل‌کننده بر اساس سوابق مورد تشخیص واقع شوند، ولی بالاخره یک برنامهٔ بداندیش باید موضوع کاملاً تعریف‌شده‌ای را از سیستم عامل درخواست کند. با فرض اینکه سدکنندهٔ رفتار بتواند چنین درخواستی را تشخیص دهد، خواهد توانست عملیات بداندیشانه را صرف نظر از اینکه منطق برنامه چقدر پیچیده و سردرگم‌کننده باشد، تشخیص و جلوی آن را سد کند.

روشن است که توانائی پایش نرم‌افزار در حال اجرا در زمان حال، مزیت بزرگی را برای سدکنندهٔ رفتار به ارمغان می‌آورد ولی این امر دارای نقاط ضعفی نیز هست. چون برنامهٔ بداندیش قبل از اینکه همهٔ رفتارهای آن مشاهده شود بایستی روی سیستم هدف اجرا گردد، ممکن است قبل از اینکه به توسط سیستم سدکنندهٔ رفتار شناسائی و مسدود گردد، صدمات زیادی را به سیستم وارد نماید. بعنوان مثال، یک ویروس جدید ممکن است تعدادی از فایل‌های ظاهراً غیرمهم را در دیسک سخت جابجا نموده و سپس به یک فایل تنها حمله کرده و آن را آلوده سازد که همین آلودگی اخیر باعث تشخیص و سدشدن راه آن گردد. اگرچه آلودگی اصلی مسدود شده است ولی کاربر ممکن است در پیداکردن فایل‌های خود دچار مشکل شود که خود باعث از دست رفتن کارآئی و یا بدتر از آن خواهد بود.

۳-۱۰ حملات توزیع شدهٔ انکار سرویس

حملات توزیع‌شدهٔ انکار سرویس DDoS (Distributed Denial of Service) یک تهدید مهم امنیتی برای سازمان‌ها محسوب گردیده و بنظر می‌رسد که اینگونه حملات در حال افزایش‌اند[VIJA02]. در یک بررسی سه هفته‌ای در سال ۲۰۰۱، محققین بیش از ۱۲,۰۰۰ حمله به بیش از ۵,۰۰۰ هدف مشخص را شناسائی کردند. هدف‌ها انواع متفاوتی داشته و کمپانی‌های شناخته شدهٔ بزرگی مانند Amazon و Hotmail تا ISPهای کوچک خارجی و اتصالات تلفنی را می‌پوشاندند

[MOOR01]. حملات DDoS با بمباران کردن سِرورها، شبکه‌ها و یا حتی کاربران انتهائی با ترافیک بی‌خاصیت باعث می‌شوند که کاربران قانونی نتوانند به آن منابع دسترسی یابند. در یک حملهٔ DDos معمول، تعداد زیادی از میزبان‌های مورد تهاجم قرارگرفته گردهم می‌آیند تا بسته‌های دیتای بی‌حاصل را ارسال نمایند. در سال‌های اخیر، روش‌های حمله و ابزارهای بکارگرفته شده برای این منظور پیچیده‌تر، مؤثرتر و دنبال کردن آن تا هدف نهائی سخت‌تر شده‌اند بطوری که تکنولوژی‌های دفاع، در برابر حملات بسیار گسترده قادر به مقاومت نیستند[CHAN02].

یک حملهٔ انکار سرویس (DoS) تلاشی برای ناکام گذاشتن کاربران قانونی در استفاده از آن سرویس است. وقتی این حمله از یک میزبان منفرد و یا یک گرهٔ شبکه شروع می‌شود آن را به سادگی یک حملهٔ DoS می‌خوانند. یک تهدید جدی‌تر حملهٔ DDoS است. در یک حملهٔ DDoS، یک مهاجم قادر است تا تعدادی از میزبان‌های روی اینترنت را به نحوی سازمان‌دهی نماید که همه با هم و یا بصورت متوالی به یک هدف حمله کنند. این بخش به حملات DDoS می‌پردازد. در ابتدا به ماهیت و انواع حمله اشاره می‌کنیم. سپس به نحوهٔ آماده‌سازی شبکه‌ای از میزبان‌ها برای انجام حمله می‌پردازیم. در انتها، روش‌های مقابله با این حملات را بررسی خواهیم کرد.

توصیف حملهٔ DDoS

یک حملهٔ DDoS تلاش می‌کند تا منابع هدف حمله را طوری بکار گیرد تا از دادن سرویس باز مانند. یک روش برای طبقه‌بندی حملات DDoS دسته‌بندی آنها بر اساس منبع بکارگرفته شده است. در حالت کلی، منبع بکارگرفته شده منابع داخلی سیستم هدف و یا ظرفیت انتقال دیتای شبکهٔ محلی سیستم هدف است.

یک مثال ساده از **حمله به منابع داخلی**، حملهٔ SYN flood است. شکل ۵-۱۰الف مراحل این حمله را نشان می‌دهد:

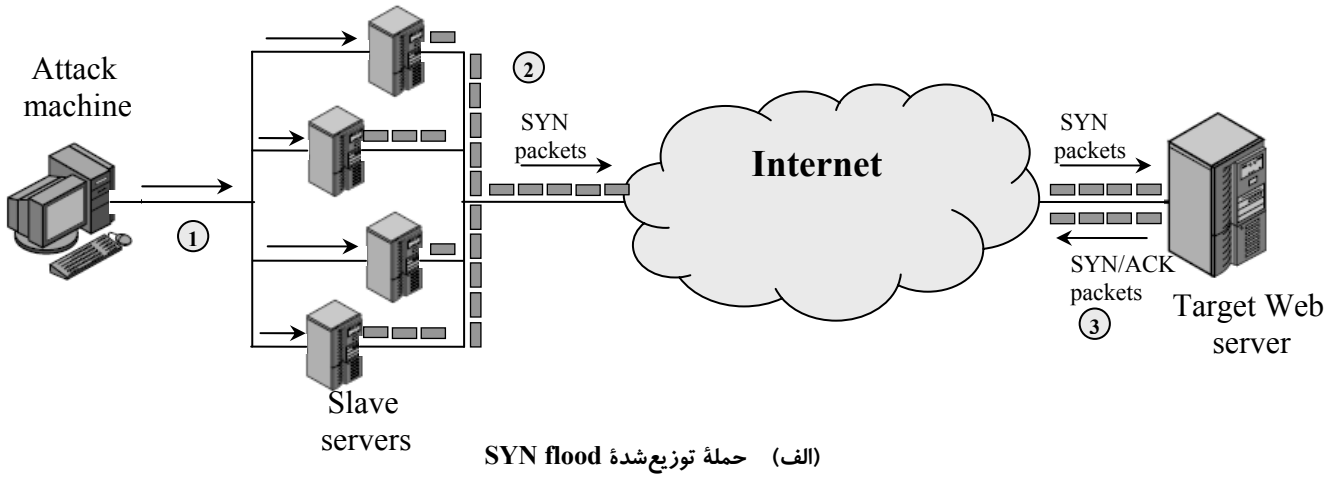
۱- حمله‌کننده کنترل تعدادی از میزبان‌های روی اینترنت را به دست گرفته و آنها را برای حمله به سِرور وِب تعلیم می‌دهد.

۲- میزبان‌های بخدمت گرفته شده شروع به ارسال بسته‌های TCP/IP SYN (synchronize/initialization) برای هدف، با اطلاعات غلط در موردآدرس‌های برگشتی IP، می‌نمایند.

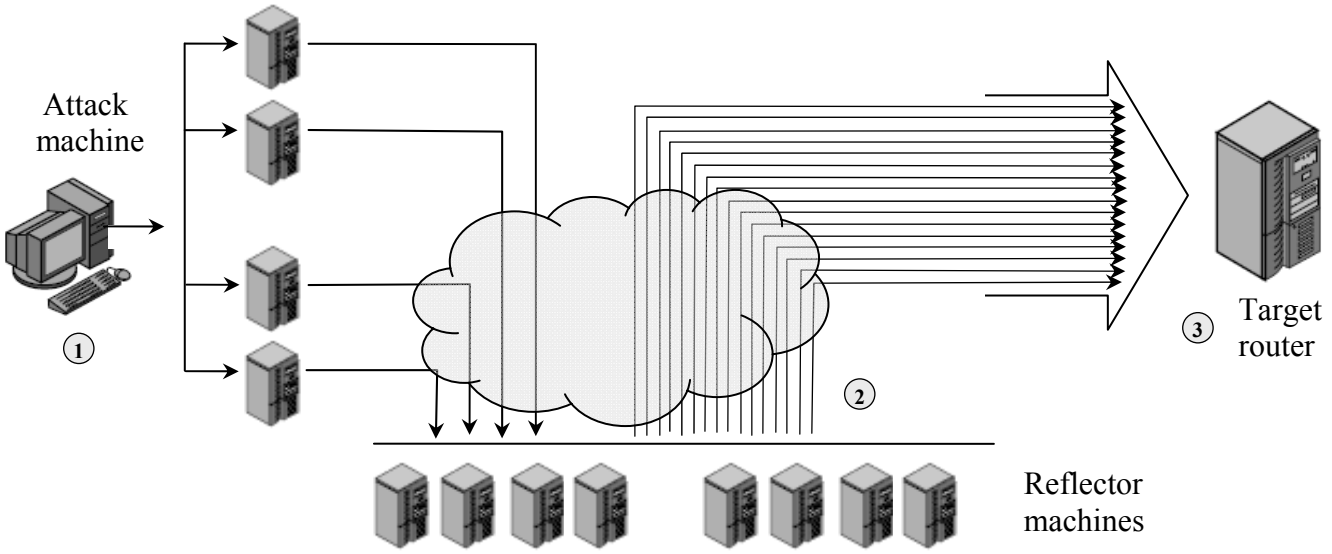
۳- هر بستهٔ SYN، یک درخواست برای گشودن یک اتصال TCP است. برای هریک از چنین بسته‌هائی، سِرور وِب با یک بستهٔ SYN/ACK (synchronize/acknowledge) جواب داده و تلاش می‌کند تا یک اتصال TCP را با واحد TCP در آدرس IP ساختگی ایجاد کند. سِرور وِب برای هر درخواست SYN که منتظر پاسخ است یک ساختار داده را نگاه داشته و همینطور که درخواست‌های بیشتری وارد می‌شود کم‌کم در باطلاق فرو می‌رود. نتیجهٔ امر این است که در حالی که ماشین قربانی شده، منتظر کامل شدن اتصالات «نیمه باز» قلابی است از پاسخ دادن به اتصالات قانونی باز می‌ماند.

حالت ساختار داده TCP، یک هدف معمول از نوع منابع داخلی بوده ولی به هیچوجه تنها منبع داخلی نیست. [CERT01] مثال‌های دیگری را ارائه می‌دهد:

۱- در بسیاری سیستم‌ها، تعداد معدودی ساختار داده وجود دارند که اطلاعات پردازشی را نگاه می‌دارند (شناسه‌های پردازش، جداول پردازش و غیره). یک مهاجم ممکن است بتواند با نوشتن یک برنامهٔ ساده و یا چند فرمان که کاری جز خلق کپی‌های مکرر از خود کاری انجام نمی‌دهند، این ساختارها را اشغال کند.



(الف) حملهٔ توزیع‌شدهٔ SYN flood



(ب) حملهٔ توزیع‌شدهٔ ICMP

شکل ۵-۱۰ مثال‌هائی از حملات سادهٔ DDoS

۲- یک مهاجم همچنین ممکن است تلاش کند تا ظرفیت دیسک را به طرق دیگر پرکند که از آن جمله‌اند:

- خلق تعداد بیشماری از پیام‌های پستی
- خلق خطاهای عمدی که بایستی ثبت شوند
- قراردادن فایل‌ها در نواحی ناشناس ftp یا نواحی ناشناس محیط‌های اشتراکی شبکه

شکل ۵-۱۰ب مثالی از یک **حمله به منابع انتقال دیتا** را نشان می‌دهد. قدم‌های زیر برداشته می‌شوند:

۱- حمله‌کننده، کنترل چندین میزبان روی اینترنت را به دست می‌گیرد و آنها را تعلیم می‌دهد تا بسته‌های ICMP ECHO با آدرس جعل شدهٔ IP هدف را به دسته‌ای از میزبان‌ها که بعنوان منعکس‌کننده عمل می‌کنند ارسال نمایند.
۲- گره‌های واقع در سایت منعکس‌کننده، درخواست‌های جعلی را دریافت کرده و در جواب بسته‌های پاسخ echo را برای سایت هدف ارسال می‌کنند.
۳- مسیریاب هدف تهاجم، با بسته‌های ارسالی از سایت منعکس‌کننده بمباران شده و دیگر ظرفیتی برای انتقال ترافیک قانونی باقی نمی‌ماند.

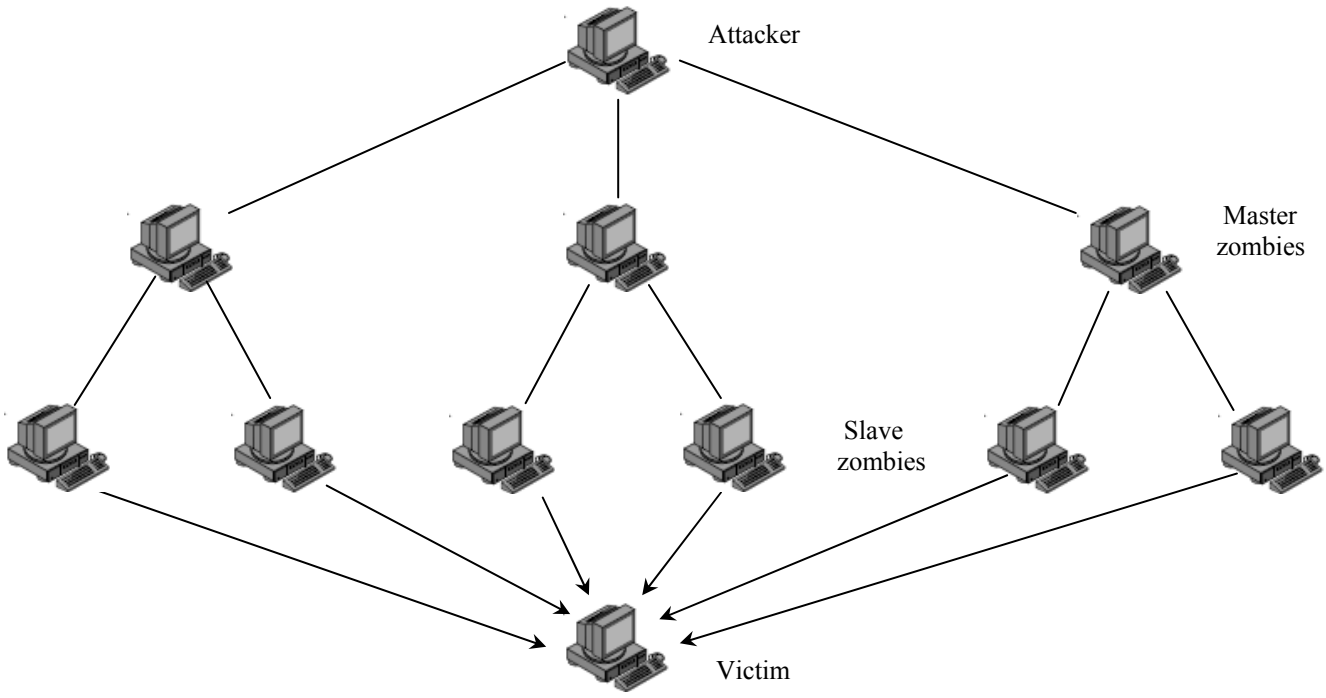
روش دیگری برای طبقه‌بندی حملات DDoS، تقسیم آنها به حملات مستقیم و حملات انعکاسی است. در یک **حملهٔ مستقیم (direct DDoS)** که در شکل ۶-۱۰الف نشان داده شده است، حمله‌کننده قادر است تا نرم‌افزارهای زامبی را در تعدادی از سایت‌هائی که در اینترنت پراکنده‌اند بنشاند. اغلب حملهٔ DDoS شامل دو سطح از ماشین‌های زامبی است: زامبی های ارباب و زامبی‌های بَرده. میزبان‌های هردو نوع زامبی با برنامهٔ بداندیش آلوده شده‌اند. حمله‌کننده، زامبی‌های ارباب را هماهنگ و به حمله وامی‌دارد که آنها هم به نوبهٔ خود زامبی‌های بَرده را هماهنگ و به حمله وامی‌دارند. استفاده از دو سطح زامبی، دنبال کردن حمله و یافتن منشاء آن را دشوارتر نموده و شبکهٔ حمله‌کنندهٔ مقاوم‌تری را بوجود می‌آورد.

یک **حملهٔ انعکاسی (reflector DDoS)**، لایهٔ دیگری از ماشین‌ها را در حمله وارد می‌کند (شکل ۶-۱۰ب). در این نوع حمله، زامبی‌های بَرده بسته‌های دیتائی را می‌سازند که نیاز به پاسخی دارد که شامل آدرس IP هدف بعنوان آدرس منبع در سرآیند بستهٔ IP است. این بسته‌ها به ماشین‌های غیرآلوده‌ای ارسال می‌شوند که منعکس‌کننده خوانده می‌شوند. ماشین‌های غیرآلوده با بسته‌هائی به مقصد ماشین هدف به این درخواست‌ها پاسخ می‌دهند. یک حملهٔ DDoS انعکاسی بآسانی می‌تواند ماشین‌های بیشتر و ترافیک بیشتری را نسبت به حملهٔ DDoS مستقیم درگیر کرده و بنابراین آسیب آن بیشتر است. علاوه برآن، یافتن منشاء حمله و فیلتر کردن بسته‌های حمله‌کننده سخت‌تر بوده زیرا حمله از تعدادی ماشین غیرآلوده که در سطح وسیع گسترده‌اند سرچشمه می‌گیرد.

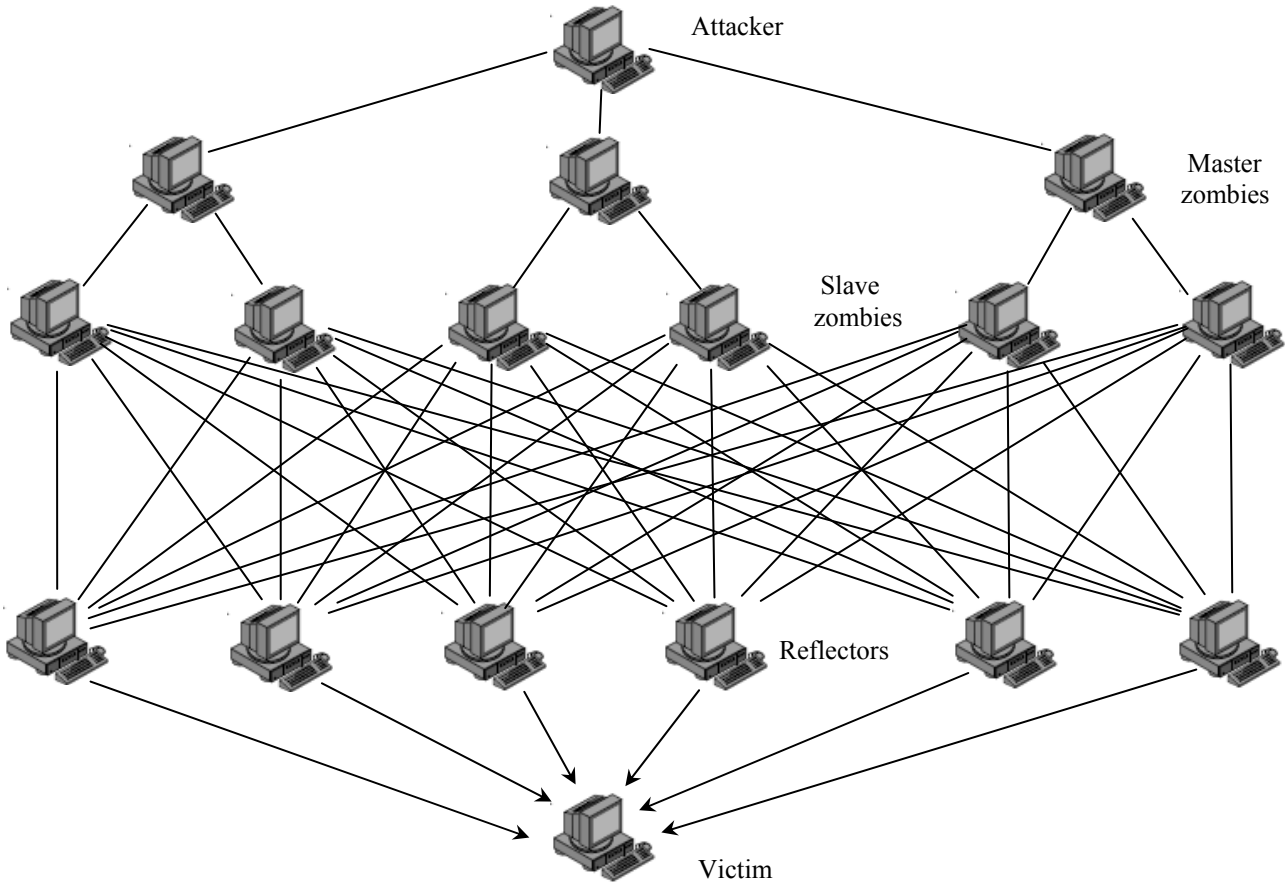
ایجاد شبکهٔ حمله‌کننده

اولین قدم در یک حملهٔ DDoS این است که حمله‌کننده تعدادی نرم‌افزار زامبی را در تعدادی ماشین بکارد تا آنها بعداً بتوانند حمله را آغاز کنند. اجزاء ضروری این مرحله از حمله عبارتند از:

۱- نرم‌افزاری که بتواند حملهٔ DDoS را انجام دهد. نرم‌افزار باید بتواند روی تعداد زیادی ماشین اجرا شده، بایستی بتواند حضور خود را پنهان کرده، باید بتواند ارتباط خود را با حمله‌کننده حفظ کرده و یا از یک مکانیسم خود انفجاری بهره‌مند بوده، و بالاخره بایستی قادر باشد تا حملهٔ برنامه‌ریزی شده را روی هدف اجرا نماید.
۲- وجود یک آسیب‌پذیری در تعداد زیادی از سیستم‌ها. حمله‌کننده بایستی از وجود یک نقطهٔ آسیب‌پذیر یا حفرهٔ امنیتی که تعداد زیادی از مدیران سیستم‌ها و کاربران منفرد از آن غافل‌اند باخبر بوده تا بتواند نرم‌افزار زامبی را در آن نقاط نصب نماید.
۳- یک استراتژی برای یافتن ماشین‌های آسیب‌پذیر، فرایندی که اِسکن کردن نامیده می‌شود.



(الف) حملهٔ DDoS مستقیم



(ب) حملهٔ DDoS انعکاسی

شکل ۶-۱۰ انواع حملات بمبارانی DDoS

در عمل اِسکن کردن، حمله‌کننده ابتدا به جستجوی تعدادی ماشین آسیب‌پذیر پرداخته و آنها را آلوده می‌سازد. آنگاه بطور معمول، نرم‌افزار زامبی که در ماشین آلوده نصب شده است همان عمل اِسکن کردن را تکرار کرده تا یک شبکهٔ بزرگ گسترده از ماشین‌های آلوده ایجاد شود. [MIRK04] از استراتژی‌های زیر برای عمل اِسکن نام می‌برد:

- **Random**: هر میزبان به دام افتاده، به آدرس‌های تصادفی در فضای آدرس‌های IP نفوذ کرده و برای هرکدام از یک seed مختلف استفاده می‌کند. این تکنیک حجم بالائی از ترافیک اینترنت را بوجود می‌آورد که ممکن است حتی قبل از آغاز حملهٔ اصلی سرویس را مختل سازد.
- **Hit-list**: حمله‌کننده ابتدا یک لیست طولانی از ماشین‌هائی که پتانسیل آسیب‌پذیری دارند را تهیه می‌کند. این امر ممکن است به کندی و در طول زمان انجام شود تا از تشخیص این که حمله‌ای در شرف وقوع است اجتناب شود. وقتی که لیست تهیه و جمع‌آوری گردید، حمله‌کننده شروع به آلوده کردن ماشین‌های موجود در لیست می‌نماید. به هر ماشین آلوده شده، بخشی از لیست برای اِسکن کردن واگذار می‌شود. این استراتژی به اِسکن سریع تعداد زیادی ماشین در مدت کوتاهی منجر شده که می‌تواند تشخیص وقوع آلودگی را با دشواری مواجه سازد.
- **Topological**: این روش از اطلاعات موجود در ماشین قربانی استفاده کرده تا میزبان‌های جدیدی را برای اِسکن کردن بیابد.
- **Local subnet**: اگر میزبانی که پشت یک دیوار آتش قرار دارد بتواند آلوده شود، آنگاه این میزبان در شبکهٔ محلی خود به دنبال آلوده کردن اهداف دیگر خواهد رفت. این میزبان از ساختار آدرسی زیرشبکه استفاده کرده تا میزبان‌های دیگری را که در غیر اینصورت تحت حفاظت دیوار آتش می‌بودند پیدا کند.

روش‌های مقابله با DDoS

در حالت کلی سه خط دفاعی در برابر حملات DDoS وجود دارد [CHAN02]:

- **جلوگیری از حمله و بازدارندگی (قبل از حمله)**: این مکانیسم‌ها قربانی را قادر می‌سازد تا بدون اینکه برای کلاینت‌های قانونی از سرویس دادن باز ماند، در برابر تلاش برای حمله مقاومت نماید. تکنیک‌های این مورد شامل اعمال سیاست‌های مناسب برای بکارگیری منابع و تدارک دیدن منابع رزرو در صورت تقاضاست. علاوه براین، مکانیسم‌های بازدارنده، سیستم‌ها و پروتکل‌های اینترنتی را طوری جرح و تعدیل می‌نماید که احتمال حملهٔ DDoS کم شود.
- **تشخیص حمله و فیلتر کردن (در طول حمله)**: این مکانیسم‌ها تلاش می‌کنند تا حمله را هرچه سریع‌تر تشخیص داده و عکس‌العمل بلادرنگ نشان دهند. این امر اثرات حمله بر هدف را به حداقل می‌رساند. تشخیص شامل جستجوی رفتارهای مشکوک است. پاسخ شامل فیلترکردن بسته‌های دیتائی است که احتمالاً بخشی از حمله هستند.
- **جستجوی منشاء حمله و شناسائی (در طول حمله و بعد از آن)**: این تلاشی برای پیدا کردن منبع حمله بعنوان قدمی در جهت جلوگیری از حملات آتی است. این روش، حتی اگر موفقیت‌آمیز باشد، معمولاً نتایج سریعی را در جهت مقابله با حملهٔ درحال انجام بدست نمی‌دهد.

چالش عمده در مبارزه با حملات DDoS روش‌های متنوع عملیاتی آنهاست. پاتک‌های DDoS بایستی به همراه تهدیدها تکامل یابند.

۱۰-۴ منابع مطالعاتی

برای یک فهم کامل از ویروس‌ها، باید کتاب [SZOR05] را خواند. منبع بسیار خوب دیگر [HARL01] است. مقالات خوب در مورد ویروس‌ها و کِرم‌ها [CASS01]، [FORR97]، [KEPH97] و [NACH97] می‌باشند. [MEIN01] اطلاعات مفیدی در مورد کِرم Code Red را در اختیار می‌گذارد.

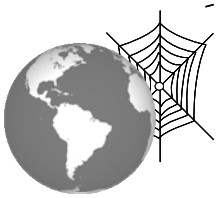
[PATR04] یک بررسی ارزنده از حملات DDoS است. [MIRK04] یک توصیف کامل از حملات DDoS و پاتک‌های آنها را ارائه می‌کند. [CHAN02] یک بررسی خوب از استراتژی‌های دفاعی DDoS است.

CASS01 Cass, S. "Anatomy of Malice." *IEEE Spectrum*, November 2001.
CHAN02 Chang, R. "Defending Against Flooding-Based Distributed Denial-of-Servise Attacks: A Tutorial." *IEEE Communications Magazine*, October 2002.

FORR97 Forrest, S.; Hofmeyr, S.; and Sommayaji, A. "Computer Immunology." *Communications of the ACM*, October 1997.
HARL01 Harley, D.; Slade, R.; and Gattiker, U. *Viruses Revealed*. New York: Osborne/McGraw-Hill, 2001.
KEPH97 Kephart, J.; Sorkin, G.; Chess,D.; and White, S."Fighting Computer Viruses." *Scientific American*, November 1997.
MEIN01 Meinel, C. "Code Red for the Web." *Scientific American*, October 2001.
MIRK04 Mirkovic, J., and Relher,P."A Taxonomy of DDoS Attack and DDoS Defence Mechnisms." *ACM SIGCOMM Computer Communications Review*, April 2004.

NACH97 Nachenberg, C. "Computer Virus-Antivirus Coevolution." *Communications of the ACM*, January 1997.
PATR04 Patrikakis, C.; Masikos, M.; and Zouraraki, O."Distributed Denial of Service Attacks." *The Internet Protocol Journal*, December 2004.
SZOR05 Szor, p., *The Art of Computer Virus Research and Defence*.Reading,MA:Addison-Wesley,2005.

وِب سایت‌های مفید



- **AntiVirus Online**: سایت IBM دربارهٔ اطلاعات مربوط به ویروس‌ها.
- **Vmyths**: مختص برملاکردن خطرات ویروس‌ها و رفع شبهات مربوط به ویروس‌های واقعی.
- **DDoS Attacks/Tools**: لیست گسترده‌ای از لینک‌ها و اسناد مربوط به این بحث.

۱۰-۵ واژه‌های کلیدی، سؤالات مرورکنندهٔ بحث و مسائل

واژه‌های کلیدی

ویروس ماکرو	macro virus	نوعی ابزار نفوذگری	auto-rooter
نرم‌افزار بداندیش	malicious software (malware)	درب مخفی	backdoor
ویروس چندچهره	polymorphic virus	سیستم مصون دیجیتال	digital immune system
حملهٔ DDoS انعکاسی	reflector DDoS attack	حملهٔ DDoS مستقیم	direct DDoS attack
ابزار دسترسی به ریشهٔ برنامه	rootkit	انکار سرویس توزیع شده	distributed denial of service
برنامهٔ ارسال هَرزنامه‌ها	spammer program	بارگذاری کننده	downloaders
ویروس پنهان شونده	stealth virus	ویروس پست الکترونیک	e-mail virus
درب مخفی	trapdoor	استثمارگرها	exploits
اسب تروا	trojan horse	حملهٔ سیلابی	flooder
ویروس	virus	ثبت کنندهٔ حرکات صفحه کلید	keylogger
کِرم	worm	جعبه ابزار	kit
زامبی	zombie	بمب لاجیک	logic bomb

سؤالات مرورکننده بحث

۱۰-۱ نقش فشرده‌سازی در عملکرد یک ویروس چیست؟

۱۰-۲ نقش رمزنگاری در عملکرد یک ویروس چیست؟

۱۰-۳ فازهای مختلف عملکرد یک ویروس یا کِرم کدام‌اند؟

۱۰-۴ یک کِرم در حالت کلی چگونه انتشار می‌یابد؟

۱۰-۵ یک سیستم مصون دیجیتال چیست؟

۱۰-۶ نرم‌افزار سدکنندهٔ رفتار چگونه کار می‌کند؟

۱۰-۷ یک DDoS چیست؟

مسائل

۱۰-۱ در برنامهٔ ویروس شکل ۱۰-۱ یک اشتباه وجود دارد. آن چیست؟

۱۰-۲ این سؤال مطرح است که آیا می‌توان برنامه‌ای نوشت که بتواند یک نرم‌افزار را تجزیه و تحلیل کرده و مشخص نماید که آیا این نرم‌افزار ویروس است یا نه؟ فرض کنید که برنامه‌ای مانند D داریم که قادر به این کار است. یعنی، برای هر برنامهٔ P اگر برنامهٔ D(P) را اجرا کنیم نتیجه یا مثبت (P ویروس است) و یا منفی (P ویروس نیست) خواهد بود. حال برنامهٔ زیر را در نظر بگیرید:

```
program CV :=
  { …
  main-program :=
          {if D(CV) then goto next :
                  else infect-executable ;
          }
next :
   }
```

در برنامهٔ بالا infect-executable یک مدول است که حافظه را بمنظور یافتن برنامه‌های اجرائی اسکن کرده و خود را در آن برنامه‌ها کپی می‌کند. تعیین کنید که آیا D می‌تواند در مورد اینکه CV یک ویروس است تصمیم صحیح بگیرد؟

فصل ۱۱

دیوارهای آتش

دیوارهای آتش می‌توانند یک وسیلهٔ مؤثر برای حفاظت یک سیستم محلی، و یا شبکه‌ای از سیستم‌ها، در مقابل تهدیدهای امنیتی مبتنی بر شبکه بوده و در عین حال دسترسی به دنیای خارج، از طریق شبکه‌های WAN و اینترنت را حفظ کنند.

این فصل را با مروری بر اصول عملکرد و نحوهٔ طراحی دیوارهای آتش آغاز می‌کنیم. سپس به مقولهٔ امنیت خود دیوار آتش پرداخته و علی‌الخصوص فرضیهٔ یک سیستم معتمد یا سیستم عامل امن را بررسی می‌نمائیم.

۱-۱۱ اصول طراحی دیوارهای آتش

سیستم‌های اطلاعات در شرکت‌ها، دوایر دولتی و سایر سازمان‌ها بطور پیوسته در حال تکامل بوده‌اند:

- سیستم متمرکز پردازش داده‌ها، با یک رایانهٔ بزرگ مرکزی، که تعدادی پایانه که مستقیماً به آن وصل‌اند را حمایت می‌نماید.
- شبکه‌های محلی (LAN) که PCها و پایانه‌ها را به هم و به رایانهٔ مرکزی متصل می‌کند.
- شبکه‌های اختصاصی که شامل تعدادی از LANها، PCهای متصل بهم، سِرورها و شاید یک یا دو رایانهٔ بزرگ مرکزی است.
- شبکه‌های وسیع تجاری، شامل تعدادی از شبکه‌های اختصاصی در مناطق جغرافیائی مختلف، که بتوسط یک WAN خصوصی با هم ارتباط دارند.
- اتصال اینترنتی که در آن شبکه‌های اختصاصی مختلف همگی به اینترنت وصل بوده و ممکن است بتوسط یک WAN خصوصی به هم متصل باشند.

برای اکثر سازمان‌ها، اتصال به اینترنت دیگر امروز یک امر تشریفاتی نیست. استفاده از اطلاعات و سرویس‌های موجود در اینترنت از ضروریات سازمانی محسوب می‌شود. علاوه برآن تک‌تک کاربران درون یک سازمان نیز تمایل و نیاز به دسترسی به اینترنت دارند و اگر این امر بتوسط شبکهٔ LAN سازمان آنها فراهم نگردد، از قابلیت‌های خط تلفن استفاده کرده و PC خود را از طریق یک فراهم‌آورندهٔ سرویس اینترنتی (ISP) به اینترنت متصل می‌سازند. از سوی دیگر در حالی‌که دسترسی به اینترنت منافعی را برای سازمان به ارمغان می‌آورد ولی دنیای خارج را نیز قادر می‌سازد تا به تجهیزات شبکه‌های محلی دسترسی یافته و با آنها تبادل اطلاعات داشته باشد. این امر تهدیدی را برای سازمان بوجود می‌آورد. اگرچه ممکن است که هر ایستگاه کاری و سِرور یک شبکه را با تجهیزات امنیتی قوی تجهیز کرد ولی چنین حفاظتی، یک روش عملی مناسب نیست. شبکه‌ای با صدها و شاید هزارها سیستم را در نظر بگیرید که از معجونی از نسخه‌های متفاوت UNIX و Windows استفاده می‌کند. وقتی یک شکاف امنیتی کشف شود، هر سیستمی که تحت تأثیر این امر قرار گرفته بایستی برای رفع مشکل

ارتقاء یابد. راه حل دیگر که بطور فزاینده‌ای مورد پذیرش قرار گرفته است، استفاده از دیوار آتش است. دیوار آتش بین شبکهٔ اختصاصی و اینترنت قرار می‌گیرد تا یک پیوند کنترل شده را ایجاد کرده و یک دیوار امنیتی خارجی را در پیرامون شبکه ایجاد نماید. هدف این دیوار پیرامونی، این است که شبکه را از حملات امنیتی اینترنتی حفاظت کرده و با فراهم آوردن تنها یک منفذ، مسئولین شبکه را قادر سازد تا از آن منفذ موارد امنیتی و ممیزی شبکه را کنترل کنند. دیوار آتش ممکن است یک کامپیوتر تنها و یا مجموعه‌ای از چند سیستم کامپیوتری باشد که با تعامل با یکدیگر وظایف دیوار آتش را انجام دهند.

در این بخش ابتدا مشخصات کلی دیوارهای آتش را بررسی می‌کنیم. سپس به انواع دیوارهای آتش که امروز مورد استفاده‌اند نظری می‌اندازیم. بالاخره تعدادی از متداول‌ترین پیکربندی‌های دیوارهای آتش را مورد بررسی قرار می‌دهیم.

مشخصه‌های دیوار آتش (Firewall)

[BELL94b] اهداف طراحی یک دیوار آتش را چنین بیان می‌کند:

۱- تمام ترافیک داخل به خارج و بالعکس بایستی از میان دیوار آتش عبور نماید. این امر با مسدود کردن فیزیکی تمام دسترسی‌ها به شبکهٔ محلی، بجز از طریق دیوار آتش حاصل می‌گردد. از پیکربندی‌های متنوعی در این مورد می‌توان استفاده کرد که بعداً مورد بحث قرار خواهد گرفت.

۲- تنها به ترافیک معتبر، برابر آنچه خط‌مشی امنیتی محلی آن را تعریف کرده است، اجازهٔ عبور داده می‌شود. انواع متنوعی از دیوارهای آتش مورد استفاده قرار می‌گیرندکه هریک انواع مختلفی از خط‌مشی‌ها را ایجاد می‌کنند. بعداً در این باره بحث خواهیم کرد.

۳- خود دیوار آتش در مقابل نفوذ بیگانه دارای امنیت است. این امر استفاده از یک سیستم مورد اعتماد یا یک سیستم عامل امن را ایجاب می‌نماید. این موضوع را نیز بعداً مورد بحث قرار خواهیم داد.

[SMIT97] چهار تکنیک عام، که دیوارهای آتش برای کنترل دست‌یابی و عملیاتی کردن خط‌مشی امنیتی سایت مورد استفاده قرار می‌دهند، را ذکر کرده است. در ابتدا دیوارهای آتش عمدتاً روی کنترل سرویس نظارت داشتند ولی تکامل آنها باعث شده است که هرچهار منظور را مورد توجه قرار دهند:

- **کنترل سرویس**: نوع سرویس‌های اینترنتی، چه در محدودهٔ شبکه و چه در خارج از محدودهٔ شبکه، که می‌توانند قابل دست‌یابی باشند را تعیین می‌کند. دیوار آتش ممکن است ترافیک را بر اساس آدرس IP و شمارهٔ پورت TCP فیلتر کند، ممکن است نرم‌افزار پروکسی(proxy)، که درخواست هرنوع سرویس قبل از عبور آن به مقصد را دریافت و تحلیل می‌نماید، فراهم سازد و یا ممکن است خود بستر نرم‌افزار سِرور همانند سرویس وِب و یا پست الکترونیک باشد.
- **کنترل جهت**: جهتی که فقط در آن جهت درخواست سرویس بخصوصی پذیرفته شده و اجازهٔ عبور از دیوار آتش دارد را تعیین می‌کند.
- **کنترل کاربر**: دست‌یابی به یک سرویس، بر اساس اینکه کدام کاربر می‌خواهد از آن استفاده کند، را کنترل می‌کند. این کنترل معمولاً به کاربران داخل محدودهٔ دیوار آتش (کاربران محلی) اعمال می‌شود. این سرویس همچنین ممکن است به ترافیک ورودی کاربران خارج از محدوده نیز اعمال شود که در این حالت نیاز به نوعی روش اعتبارسنجی همانند IPSec است.

- **کنترل رفتار**: کنترل چگونگی استفاده از سرویس را بعهده دارد. بعنوان مثال دیوار آتش ممکن است نامه‌های الکترونیک را برای جلوگیری از عبور هَرزنامه (spam) فیلتر کند و یا ممکن است دسترسی خارجی را تنها به بخشی از اطلاعات سِرور وِب ممکن سازد.

قبل از پرداختن به جزئیات انواع دیوار آتش و پیکربندی‌های مختلف آن، بهتر است آنچه را که می‌توان از یک دیوار آتش انتظار داشت خلاصه کنیم. قابلیت‌های زیر معمولاً در حوزهٔ عملکرد یک دیوار آتش قرار دارد:

۱- یک دیوار آتش یک گلوگاه منفرد را تعریف می‌کند تا کاربران غیرمعتبر را از شبکهٔ محافظت شده دور نگاه داشته، سرویس‌های بالقوه خطرآفرین را از ورود به شبکه و خروج از شبکه مانع شده، و حفاظت از انواع متنوع حملات تقلیدIP و مسیریابی را ایجاد کند. استفاده از یک گلوگاه منفرد، مدیریت امنیت را تسهیل می‌نماید زیرا قابلیت‌های امنیتی در یک سیستم تنها، و یا مجموعه‌ای از سیستم‌ها متمرکز می‌شود.

۲- یک دیوار آتش، محلی برای پائیدن پیشامدهای مرتبط با امنیت را ایجاد می‌کند. ممیزی‌ها و آلارم‌ها می‌توانند روی یک سیستم دیوار آتش بنا نهاده شوند.

۳- یک دیوار آتش یک بستر مناسب برای چندین عمل اینترنتی است که ربطی به امنیت ندارند. اینها شامل یک مترجم آدرس شبکه است که آدرس‌های محلی را به آدرس‌های اینترنتی نگاشت نموده و یا وظیفه‌ای مدیریتی است که اتصال به اینترنت را اجازه داده و یا ممیزی می‌نماید.

۴- یک دیوار آتش می‌تواند بعنوان بستر IPSec عمل نماید. با استفاده از قابلیت مُود تونل (tunnel mode) توصیف شده در فصل ۶، دیوار آتش می‌تواند برای ایجاد شبکه‌های خصوصی مجازی (VPN) بکار رود.

دیوارهای آتش محدودیت‌های مخصوص به خود را نیز داشته که شامل موارد ذیل‌اند:

۱- دیوار آتش نمی‌تواند در مقابل حملاتی که دیوار آتش را دور می‌زنند مقاومت کند. سیستم‌های داخلی شبکه ممکن است از قابلیت شماره‌گیری تلفنی برای اتصال به یک ISP استفاده کنند. یک LAN داخلی ممکن است از یک مخزن مُودم استفاده کند که قابلیت اتصال به اینترنت از طریق خط تلفن برای کارمندانی که در ماموریت خارج از سازمان هستند را فراهم نماید.

۲- دیوار آتش در برابر تهدیدهای داخلی همانند یک کارمند ناراضی و یا کارمندی که سهواً به یک نفوذگر خارجی کمک می‌کند حفاظتی ایجاد نمی‌کند.

۳- دیوار آتش نمی‌تواند در برابر انتقال برنامه‌ها یا فایل‌های ویروسی ایجاد حفاظت نماید. با توجه به تنوع سیستم‌های عامل و تنوع برنامه‌های کاربردی در درون یک محدوده، برای دیوار آتش غیرعملی و شاید غیرممکن است که تمام فایل‌های ورودی، e-mailها و پیام‌ها را بمنظور یافتن ویروس‌ها اِسکن نماید.

انواع دیوارهای آتش

شکل ۱-۱۱ سه نوع معمول دیوار آتش را نشان می‌دهد: فیلترهای بسته‌های دیتا (packet filters)، دروازه‌های سطح کاربرد (application-level gateways) و دروازه‌های سطح مدار (circuit-level gateways). هریک از این سه نوع را به نوبت بررسی می‌کنیم.

مسیریاب فیلترکنندهٔ بسته‌های دیتا (Packet- Filtering Router)

یک مسیریاب فیلترکنندهٔ بسته‌ها، یک سری قواعد را به بسته‌های IP ورودی اعمال کرده و آنگاه یا آنها را بجلو رانده و یا معدوم می‌کند. مسیریاب نوعاً طوری پیکربندی می‌شود که بسته‌های دیتا را در هر دو جهت (بسمت داخل و بسمت خارج شبکه) فیلتر نماید. قواعد فیلترینگ براساس اطلاعاتی است که در یک بستهٔ اطلاعاتی دیتا وجود دارد:

- **آدرس IP منبع**: آدرس IP سیستمی که بستهٔ IP را ارسال کرده است (مثلاً 192.168.1.1).
- **آدرس IP مقصد**: آدرس IP مقصدی که بستهٔ دیتا قصد رسیدن به آن را دارد (مثلاً 192.168.1.2).
- **آدرس سطح حمل‌ونقل منبع و مقصد**: شمارهٔ پورت سطح حمل‌ونقل (مثلTCP یا UDP)که کاربردهائی مثل SNMP یا TELNET را تعریف می‌کند.
- **میدان پروتکل IP**: نوع پروتکل حمل‌ونقل را تعریف می‌کند.
- **مدار واسط**: برای یک مسیریاب با ۳ پورت و بیشتر، اینکه از کدام مدار واسط مسیریاب، بسته خارج شده و یا به کدام مدار واسط مسیریاب، بسته وارد می‌شود.

فیلتر بسته‌ها معمولاً بصورت لیستی از قواعد، که مبتنی بر تطبیق با میدان‌های سرآیند IP یا TCP است، تنظیم می‌گردد. اگر بین قواعد وضع شده با میدان‌های بسته تطابقی یافت شود، آن قاعده برای تعیین اینکه بسته عبور کرده و یا نابود شود بکار گرفته می‌شود. اگر تطبیقی یا قاعده‌ای یافت نشود آنگاه براساس پیش‌فرض عمل خواهد شد. برای پیش‌فرض دو حالت ممکن وجود دارد:

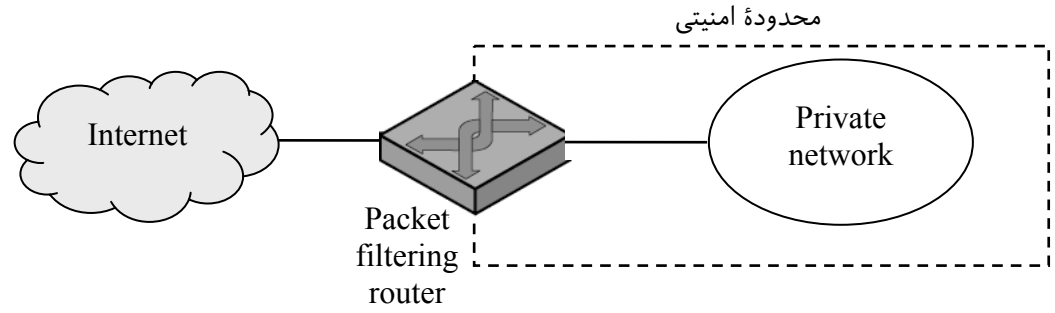
- **پیش‌فرض = *نابودی***: آنچه که مجاز تعریف نشده است ممنوع است.
- **پیش‌فرض = *عبور***: آنچه که ممنوع تعریف نشده است مجاز است.

پیش‌فرضی که نابودی را پیشنهاد می‌دهد، محافظه‌کارانه‌تر است. در ابتدا جلوی همه چیز سد می‌شود و سرویس‌ها را بایستی مورد به مورد تعریف و اضافه نمود. این روش برای کاربران ملموس‌تر بوده و احتمال اینکه آنها دیوار آتش را بعنوان یک مانع جدّی تلقی کنند بیشتر است. پیش‌فرض عبور، عملیات کاربران انتهائی را تسهیل کرده ولی امنیت کمتری را فراهم می‌سازد. در این حالت مسئول امنیت شبکه بایستی بواقع هوشیار بوده و نسبت به هر تهدید امنیتی جدید، بمحض اینکه کشف شود، واکنش نشان دهد.

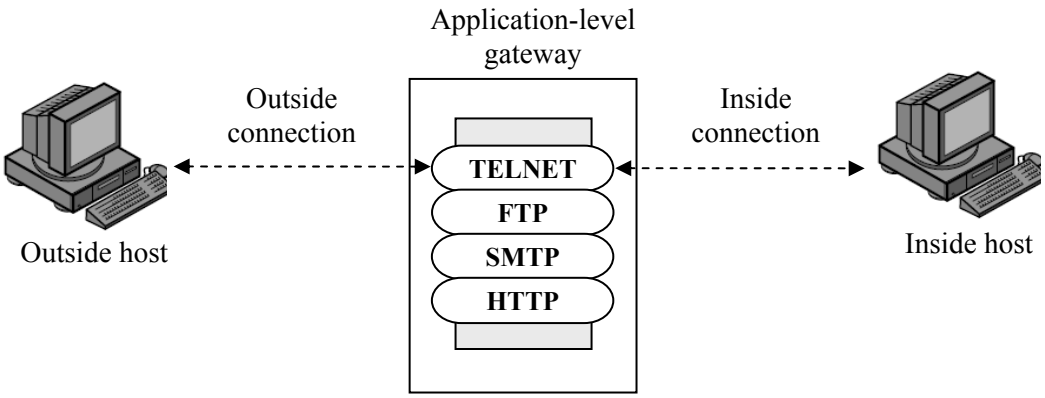
جدول ۱-۱۱ از [BELL94]، مثال‌هائی از مجموعهٔ قواعد فیلترینگ بسته‌های دیتا را نشان می‌دهد. در هر مجموعه، قوانین از بالا به پائین اعمال می‌شوند. علامت " * " در یک میدان، یک نمایشگر عام بوده که بجای آن هرچیزی می‌تواند قرار داشته باشد. فرض بر این است که پیش‌فرض = نابودی به بسته‌های دیتا اعمال می‌شود.

الف: نامه‌های وارد به محدوده (پورت ۲۵ برای SMTP ورودی است) فقط به مقصد یک دروازه میزبان مجاز است. ولی نامه‌های یک میزبان خارجی بخصوص،SPIGOT، مسدود شده زیرا این میزبان تاریخچه‌ای حاکی از ارسال فایل‌های حجیم در پیام‌های پستی خود دارد.

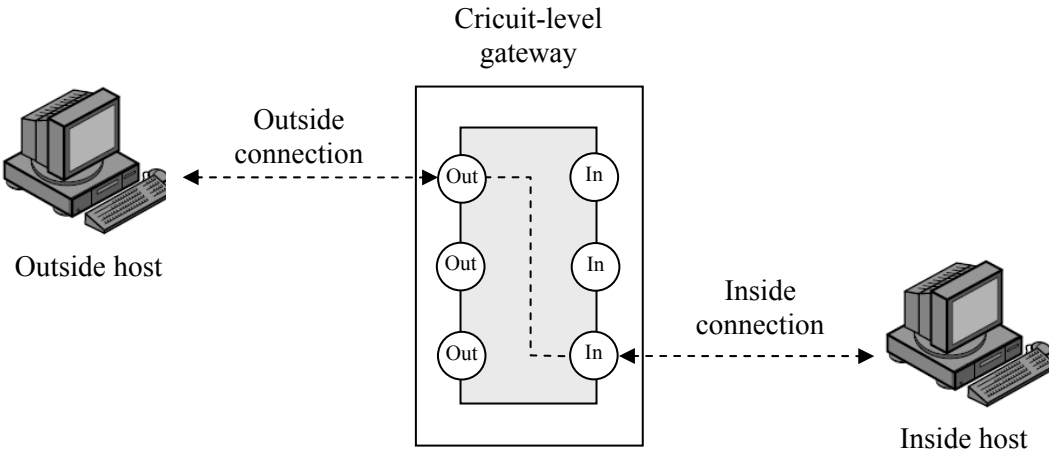
ب: این حالت بیان صریح سیاست مربوط به پیش فرض است. تمام مجموعه‌های قواعد در انتهای کار بطور ضمنی از این قانون پیروی می‌کنند.



(الف) مسیریاب فیلترکنندهٔ بسته‌های دیتا



(ب) دروازهٔ سطح کاربرد



(ج) دروازهٔ سطح مدار

شکل ۱-۱۱ انواع دیوار آتش

جدول ۱-۱۱ مثال‌هائی از فیلترینگ بسته‌های دیتا

action	ourhost	port	theirhost	port	comment
block	*	*	SPIGOT	*	we don`t trust these people
allow	OUR-GW	25	*	*	connection to our SMTP port

(الف)

action	ourhost	port	theirhost	port	comment
block	*	*	*	*	default

(ب)

action	ourhost	port	theirhost	port	comment
allow	*	*	*	25	connection to their SMTP port

(ج)

action	src	port	dest	port	flags	comment
allow	{our hosts}	*	*	25		our packets to their SMTP port
allow	*	25	*	*	ACK	their replies

(د)

action	src	port	dest	port	flags	comment
allow	{our hosts}	*	*	*		our outgoing calls
allow	*	*	*	*	ACK	replies to our calls
allow	*	*	*	>1024		traffic to nonservers

(هـ)

ج: این مجموعهٔ قواعد بیانگر این است که هر میزبان داخلی می‌تواند به خارج e-mail بزند. یک بستهٔ TCP با پورت مقصد شمارهٔ ۲۵ به سرور SMTP ماشین مقصد ارسال می‌گردد. اشکال این قاعده این است که استفاده از پورت ۲۵ برای دریافت SMTP، تنها یک پیش‌فرض است. یک دستگاه خارجی می‌تواند طوری پیکربندی شود که کاربرد دیگری برای پورت ۲۵ داشته باشد. با عمل به این قاعده یک مهاجم می‌تواند با ارسال بسته‌هائی با شمارهٔ ۲۵ در TCP، به کامپیوترهای داخل محدوده دسترسی یابد.

د: این مجموعهٔ قواعد به نتایجی منجر می‌گردد که در بند (ج) به آن دست نمی‌یافتیم. قواعد از مشخصهٔ اتصالات TCP بهره می‌گیرند. هروقت اتصالی برقرار شود، پرچم ACK یک سگمنت TCP به اهتزاز درآمده تا سگمنت‌های ارسال شده از سمت دیگر را تائید کند. بنابراین مجموعهٔ این قواعد بیان می‌دارند که بسته‌های IP که آدرس‌های IP مبداء آنها یکی از میزبان‌های داخلی مشخص بوده و شمارهٔ پورت TCP مقصد آنان ۲۵ است پذیرفته می‌گردند. همچنین بسته‌های ورودی که شمارهٔ پورت مبداء آنها ۲۵ بوده و پرچم ACK را در سگمنت TCP در اهتزاز دارند، مجاز به عبور می‌باشند. توجه شود که ما سیستم‌های منبع و مقصدکه بایستی این قواعد را بطور صریح تعریف کنند، با روشنی کامل مشخص نموده‌ایم.

هـ: این مجموعهٔ قواعد، یکی از روش‌های مدیریت اتصالات FTP است. درFTP از دو اتصال TCP استفاده می‌شود: یک اتصال کنترلی برای برقراری مقدمات انتقال فایل و یک اتصال دیتا برای انتقال واقعی خود فایل. اتصال دیتا از

یک شمارهٔ پورت متفاوت که بطور پویا برای این امر اختصاص می‌یابد، استفاده می‌کند. بیشتر سِرورها روی شمارهٔ پورت‌های پائین کارکرده که در نتیجه بیشتر نیز هدف حملات قرار می‌گیرند. بیشتر مکالمات خارجی تمایل به استفاده از پورت‌های با شماره های بالاتر، نوعاً بالای ۱,۰۲۳، دارند. بنابراین این قاعده به موارد زیر اجازه می‌دهد:

- بسته‌هائی که از داخل شبکه سرچشمه گرفته‌اند
- بسته‌های پاسخ برای یک اتصال که از یک ماشین داخلی سرچشمه گرفته‌اند
- بسته‌هائی که برای یک پورت شمارهٔ بالا در یک ماشین داخلی ارسال می‌شوند

این روش نیازمند این است که سیستم‌ها طوری پیکربندی شوند که تنها پورت‌های مناسب بکار گرفته شوند.

مجموعهٔ قواعد (هـ) مشکلاتی که در راه برخورد با کاربرها در سطح فیلترکردن بسته‌ها وجود دارد را خاطرنشان می‌سازد. روش دیگری برای برخورد با FTP و کاربردهای مشابه، استفاده از یک دروازه سطح کاربرد است که بعداً در این بخش به آن خواهیم پرداخت.

یکی از محاسن مسیریاب فیلترکنندهٔ بسته‌ها، سادگی آن است. همچنین فیلترهای بسته‌های دیتا نوعاً از نظر کاربران شفاف بوده و خیلی سریع هستند.[WACK02] نقاط ضعف زیر را برای دیوارهای آتش از این نوع برمی‌شمارد:

- چون دیوارهای آتش فیلترکنندهٔ بسته‌ها، داده‌های لایهٔ بالاتر را بررسی نمی‌کنند، آنها نمی‌توانند از حملاتی که عملیات و یا نقاط آسیب‌پذیر مختص به کاربرد را هدف قرار می‌دهند، جلوگیری کنند. بعنوان مثال یک دیوار آتش فیلترکنندهٔ بسته‌ها نمی‌تواند فرامین کاربردی مشخصی را بلوکه کند و بنابراین تمام عملیات موجود در آن کاربرد مجاز شناخته می‌شود.
- بعلت اطلاعات محدود دیوار آتش فیلترکنندهٔ بسته‌ها، عملیات اتصال به شبکه در این فیلتر محدود است. عملیات اتصال به سیستم معمولاً شامل همان اطلاعاتی است که از آنها برای تصمیم‌گیری در مورد کنترل دست‌یابی استفاده می‌شود (آدرس منبع، آدرس مقصد و نوع ترافیک).
- بیشتر دیوارهای آتش فیلترکنندهٔ بسته‌ها روش‌های پیشرفتهٔ تصدیق هویت کاربران را به خدمت نمی‌گیرند. بازهم این محدودیت عمدتاً بعلت فقدان کارآئی سیستم دیوار آتش در لایهٔ بالاتر است.
- آنها معمولاً در مقابل حملات و استثماری که از مشکلات درونی TCP/IP و پشتهٔ پروتکلی آن ناشی می‌شود، همانند *network layer address spoofing*، آسیب‌پذیرند. بسیاری از دیوارهای آتش فیلترکنندهٔ بسته‌ها قادر به تشخیص یک بستهٔ لایهٔ شبکه، که اطلاعات آدرسی لایهٔ سوم OSI آن تغییر داده شده است، نیستند. مهاجمین معمولاً از تقلید آدرس برای عبور از کنترل‌های امنیتی یک دیوار آتش استفاده می‌کنند.
- بالاخره بعلت تعداد کم پارامترهای دخیل در تصمیم‌گیری نسبت به کنترل دست‌یابی، دیوارهای آتش فیلترکنندهٔ بسته‌ها در معرض رخنه‌های امنیتی ناشی از پیکربندی نامناسب قرار دارند. بعبارت دیگر بصورت خیلی ساده و تصادفی، یک دیوار آتش فیلترکنندهٔ بسته‌ها ممکن است طوری پیکربندی شود که به ترافیک، نوع منابع و نوع مقاصدی که قاعدتاً بایستی بر اساس خط‌مشی امنیت اطلاعات سازمان از عبور آنها جلوگیری شود، اجازهٔ عبور دهد.

برخی از انواع حملات و روش‌های معقول دفاع در برابر آنها، که ممکن است بر علیه دیوارهای آتش فیلترکنندهٔ بسته‌ها انجام شود، بقرار زیراند:

- **تقلیدآدرسIP (IP spoofing)**: مهاجم، بسته‌هائی را از خارج ارسال می‌دارد که در محل آدرس IP منبع آنها، آدرس یک میزبان داخلی جا داده شده است. حمله‌کننده امیدوار است که استفاده از یک آدرس تقلیدی باعث شود که او بتواند در سیستم‌هائی که بسادگی فقط آدرس منبع را کنترل نموده و در آن بسته‌های مشخص میزبان‌های داخلی مورد اعتماد پذیرش می‌گردند، نفوذ یابد. ضدحملهٔ این روش این است که بسته‌هائی که از یک مدار واسط خارجی وارد شده ولی دارای آدرس یک منبع داخلی هستند را نابود کرد.
- **حملات مسیریابی منبع**: ایستگاه منبع، مسیر عبور یک بسته در اینترنت را تعیین کرده و امیدوار است که این روش، موانع امنیتی که اطلاعات مسیریابی را کنترل نمی‌کنند دور بزند. ضدحملهٔ این روش این است که تمام بسته‌هائی که از این ابزار استفاده می‌کنند را نابود کرد.
- **حملات فِرَگمنت‌های کوچک**: مهاجم از ابزار IP fragmentation استفاده کرده تا فِرگمنت‌های فوق‌العاده کوچکی را خلق نموده و اطلاعات سرآیند TCP را مجبور سازد تا در یک فِرگمنت مجزا از دیتا قرار گیرد. این حمله برای فریب دادن قواعد فیلترینگ که مبتنی بر اطلاعات سرآیند TCP هستند طراحی شده است. حمله کننده امیدوار است که تنها اولین فِرگمنت بتوسط مسیریاب فیلترکننده بررسی شده و بقیه فِرگمنت‌ها عبور نمایند. این حمله را می‌توان با معدوم کردن تمام بسته‌هائی که نوع پروتکل آنها TCP بوده و IP fragment offset آنها برابر ۱ است، خنثی کرد.

دیوارهای آتش تفتیش‌کننده حالت (Stateful Inspection Firewall)

یک فیلتر معمولی بسته‌های دیتا، تصمیمات فیلترینگ را بر مبنای تک‌تک بسته‌ها انجام داده و مضامین هیچ لایه بالاتر را بکار نمی‌گیرد. برای فهم این مطلب که مضامین لایهٔ بالاتر چه بوده و چرا یک فیلتر سنّتی بسته‌های دیتا نسبت به این امر دارای محدودیت است، لازم است که این موضوع کمی شکافته شود. اکثر کاربردهای استاندارد که روی لایهٔ TCP قرار دارند از یک مدل کِلاینت/ سِرور پیروی می‌کنند. بعنوان مثال در پروتکل سادهٔ انتقال نامه‌های الکترونیک(SMTP)، نامه از یک سیستم کِلاینت به یک سیستم سِرور منتقل می‌شود. سیستم کِلاینت پیام‌های e-mail را معمولاً بتوسط کاربر تولید می‌کند. سیستم سِرور پیام‌های پست الکترونیک ورودی را پذیرفته و آنها را در صندوق پستی کاربر قرار می‌دهد. SMTP یک اتصال TCP بین کِلاینت و سِرور برقرار می‌کند که در آن شمارهٔ پورت TCP سِرور که تعیین‌کنندهٔ کاربرد SMTP سِرور است، ۲۵ می‌باشد. شمارهٔ پورت TCP برای SMTP کِلاینت، عددی بین ۱,۰۲۴ و ۱۶,۳۸۳ است که بتوسط پروتکل SMPT کِلاینت تعیین می‌گردد.

در حالت کلی وقتی کاربردی که از TCP استفاده می‌کند تماسی با میزبان دوردست می‌گیرد، یک اتصال TCP ایجاد نموده که در آن شمارهٔ پورت TCP کاربرد دوردست (سِرور)، عددی کمتر از ۱,۰۲۴ بوده و شمارهٔ پورت TCP کاربرد محلی (کِلاینت)، عددی بین ۱,۰۲۴ و ۱۶,۳۸۳ است. اعداد کمتر از ۱,۰۲۴ شمارهٔ پورت‌های «شناخته شده» بوده و بطور دائمی به کاربردهای مشخصی تخصیص می‌یابند (مثلاً ۲۵ برای SMTP). اعداد بین ۱,۰۲۴ و ۱۶,۳۸۳ بطور پویا تولید شده و تنها برای زمان حیات یک اتصال TCP دارای اهمیت‌اند.

یک دیوار آتش فیلترکنندهٔ بسته‌های دیتا بایستی به ترافیک داخل محدوده که دارای شمارهٔ پورت‌های بالا بوده و بر مبنای TCP می‌باشند اجازهٔ عبور دهد. این امر یک نقطهٔ آسیب‌پذیر را ایجاد می‌نماید که ممکن است بتوسط کاربران غیرمجاز مورد سوءاستفاده قرار گیرد.

جدول ۲-۱۱ مثالی از جدول وضعیت اتصالات دیوار آتش از نوع تفتیش‌کنندهٔ حالت [WACK02]

Source Address	Source Port	Destination Address	Destination Port	Connection State
192.168.1.100	1030	210.9.88.29	80	Established
192.168.1.102	1031	216.32.42.123	80	Established
192.168.1.101	1033	173.66.32.122	25	Established
192.168.1.106	1035	177.231.32.12	79	Established
223.43.21.231	1990	192.168.1.6	80	Established
219.22.123.32	2112	192.168.1.6	80	Established
210.99.212.18	3321	192.168.1.6	80	Established
24.102.32.23	1025	192.168.1.6	80	Established
223.212.212	1046	192.168.1.6	80	Established

یک فیلتر بسته‌های دیتا از نوع تفتیش‌کننده حالت، ترافیک TCP را با ایجاد یک فهرست از اتصالات TCP خارج از محدودهٔ شبکه همانند جدول ۲-۱۱، با محدودیت‌های بیشتری مواجه می‌سازد. برای هر اتصال برقرار شده، یک قلم به جدول اضافه می‌گردد. فیلتر بسته‌های دیتا اکنون به ترافیک ورودی که بمقصد پورت‌های شمارهٔ بالا ارسال شده‌اند در صورتی اجازهٔ عبور می‌دهد که بسته‌ها دارای پروفایل یکی از اقلام موجود در جدول باشند.

دروازهٔ سطح کاربرد (Application - Level Gateway)

یک دروازهٔ سطح کاربرد که یک سرور پروکسی (proxy) نیز خوانده می‌شود برای ترافیک سطح کاربرد بصورت یک رله عمل می‌کند (شکل ۱-۱۱ب). کاربر با یک برنامهٔ کاربردی TCP/IP همانند Telnet یا FTP با دروازه تماس می‌گیرد و دروازه از کاربر نام میزبان دوری که کاربر تمایل به اتصال با او دارد را سؤال می‌کند. وقتی کاربر پاسخ داده و یک IP معتبر و اطلاعات اعتبارسنجی لازم را ارائه نمود، دروازه با برنامهٔ کاربردی میزبان دور تماس گرفته و سگمنت‌های TCP شامل داده‌های کاربر را بین دو نقطهٔ انتهائی رله می‌کند. اگر دروازه کُد پروکسی خاصی را فراهم ننماید، سرویس عملیاتی نشده و داده‌ها نمی‌توانند از عرض دیوار آتش عبور کنند. علاوه براین دروازه را می‌توان طوری پیکربندی کرد که تنها از حالت خاصی از کاربردها که مسئول شبکه آنها را قابل پذیرش می‌داند حمایت کرده و برای حالات دیگر از دادن سرویس خودداری نماید.

دروازه‌های سطح کاربرد نسبت به فیلترهای بسته‌های دیتا امن‌ترند. بجای تلاش برای مقابله با ترکیبات فوق‌العاده زیاد ممکن که بایستی در سطح TCP و IP مجاز یا ممنوع شناخته شوند، دروازهٔ سطح کاربرد تنها کافی است نسبت به تعداد محدودی از کاربردها دقت نماید. علاوه بر این ثبت و ممیزی تمام ترافیک ورودی در سطح کاربرد، کاری آسان است.

عیب اصلی این نوع دروازه، سربارهٔ پردازش بیشتر در هربار اتصال است. در واقع اتصال بین دو کاربر انتهائی در بین راه شکسته شده و دروازه با قرار گرفتن در این نقطهٔ مَفصلی، بایستی همهٔ ترافیک دو جهت را کنترل نماید.

دروازهٔ سطح مدار (Circuit - Level Gateway)

نوع سوم دیوار آتش، دروازهٔ سطح مدار است (شکل ۱-۱۱ج). این دیوار آتش می‌تواند یک سیستم منفرد بوده و یا می‌تواند عمل خاصی باشد که بتوسط دروازهٔ سطح کاربرد برای کاربردهای معینی انجام شود. یک دروازهٔ سطح کاربرد اجازهٔ یک اتصالTCP سر- به- سر را نمی‌دهد بلکه دروازه‌ای بین دو اتصال TCP ایجاد می‌کندکه یک اتصال بین خودش و استفاده‌کننده از TCP در میزبان داخلی، و اتصال دیگر بین خودش و استفاده‌کننده از TCP در یک میزبان خارجی واقع شده است. زمانی که دو اتصال برقرار گردید دروازه معمولاً سِگمنت‌های TCP را، بدون اینکه محتویات آنها را بررسی کند، از یک اتصال به اتصال دیگر رله می‌کند. تدبیر امنیتی بکارگرفته شده در این مورد فقط تعیین آن است که برقراری چه اتصالاتی مجاز هستند.

دروازهٔ سطح مدار معمولاً وقتی مورد استفاده قرار می‌گیرد که مسئول سیستم به کاربران داخل سیستم اعتماد داشته باشد. دروازه را می‌توان طوری پیکربندی کرد که سرویس سطح کاربرد یا سرویس پروکسی در اتصالات داخل محدوده، و عملیات سطح مدار در اتصالات خارج محدوده، را حمایت نماید. در این نوع پیکربندی، دروازه می‌تواند سربارهٔ بررسی داده‌های کاربردی ورودی برای عملیات ممنوع را تحمل کرده ولی درعوض نیازی به بررسی سربارهٔ داده‌های خروجی برای این منظور نداشته باشد.

مثالی از اجرای یک دروازهٔ سطح مدار، بستهٔ نرم‌افزاری SOCKS [KOBL92] است. نسخهٔ پنجم SOCKS در RFC 1928 تعریف شده است. این تعریف چنین است:

> پروتکل توصیف شده در اینجا بدین منظور طراحی شده است تا بستری برای کاربردهای کلاینت/ سرور در هردو بُعد TCP و UDP فراهم کرده تا بطور سهل و امن از سرویس‌های یک دیوار آتش شبکه استفاده نمایند. پروتکل اصولاً یک "shim-layer" بین لایه‌های کاربرد و حمل‌ونقل است و در چنین حالتی سرویس‌های دروازه‌ای سطح شبکه مثل جلوراندن پیام‌های ICMP را فراهم نمی‌سازد.
>
> SOCKS شامل مؤلفه‌های زیر است:
>
> - سِرور SOCKS که روی یک دیوار آتش بر پایهٔ UNIX کار می‌کند.
> - کتابخانهٔ کِلاینت SOCKS که روی میزبان‌های داخلی حفاظت شده بتوسط دیوار آتش کار می‌کند.
> - نسخه‌های SOCKS تهیه شده چندین برنامهٔ استاندارد کِلاینت همانند FTP و TELNET. پیاده‌سازی پروتکل SOCKS معمولاً شامل دوباره کامپایل کردن و یا تغییر مسیردادن کاربردهای مبتنی بر TCP کِلاینت برای استفاده از کپسولی کردن مناسب روتین‌های کتابخانهٔ SOCKS می‌باشد.

وقتی یک کِلاینت مبتنی بر TCP بخواهد با شیئی اتصال برقرار کند که تنها از طریق دیوار آتش قابل دست‌یابی است (چنین امری به پیاده‌سازی مربوط می‌شود)، بایستی یک اتصال TCP به پورت مناسب SOCKS بر روی سِرور SOCKS برقرار نماید. سرویس SOCKS روی پورت ۱,۰۸۰ TCP قرار دارد. اگر درخواست اتصال پذیرفته شود، کِلاینت وارد یک مذاکره برای تعیین روش اعتبارسنجی مورد استفاده شده، از متد انتخاب شده استفاده کرده و آنگاه تقاضای رلهٔ داده‌ها را می‌نماید. سرور SOCKS تقاضا را ارزیابی نموده و اتصال مناسب را یا برقرار و یا به آن پاسخ رد می‌دهد. مبادلات UDP به روش مشابه‌ای بررسی و اجرا می‌گردند. فی‌الواقع یک اتصال TCP باز می‌شود تا اعتبار یک کاربر را چه برای دریافت و چه برای ارسال سِگمنت‌های UDP بسنجد و این سِگمنت‌ها تا زمانی که اتصال TCP باز است به جلو رانده می‌شوند.

میزبان دِژدار (Bastion Host)

یک میزبان دژدار سیستمی است که بتوسط مدیریت دیوار آتش بعنوان یک استحکامات اساسی در امنیت شبکه شناخته شده است. نوعاً میزبان دژدار بعنوان بستری برای دروازه‌های سطح کاربرد و یا دروازه‌های سطح مدار عمل می‌کند. مشخصات عمومی یک میزبان دژدار چنین است:

- بستر سخت‌افزاری میزبان دژدار یک نسخهٔ امن از سیستم عامل خود را اجرا کرده که در نتیجه آن را به یک سیستم معتمد تبدیل می‌کند.
- تنها سرویس‌هائی روی میزبان دژدار نصب می‌شوند که مدیریت شبکه آنها را ضروری می‌داند. اینها شامل کاربردهای پروکسی مانند SMTP, FTP, DNS , Telnet و اعتبارسنجی کاربرند.
- میزبان دژدار ممکن است قبل از اینکه یک کاربر اجازهٔ دست‌یابی به سرویس‌های پروکسی را داشته باشد، نیازمند اعتبارسنجی‌های بیشتری باشد. علاوه برآن هر سرویس پروکسی ممکن است قبل از اینکه به کاربر اجازهٔ استفاده از خود را بدهد، نوعی تصدیق هویت خود را طلب نماید.
- هر پروکسی طوری پیکربندی می‌شود که تنها زیرمجموعه‌ای از مجموعهٔ فرامین کاربردی استاندارد را حمایت نماید.
- هر پروکسی طوری پیکربندی می‌شود که اجازهٔ دست‌یابی به سیستم‌های میزبان خاصی را بدهد. این بدین معنی است که مجموعهٔ فرامین/ قابلیت‌ها تنها به یک زیرمجموعه از سیستم‌های شبکهٔ حفاظت شده محدود گردد.
- هر پروکسی با ثبت کردن همهٔ رخدادهای ترافیکی، اطلاعات کاملی از هر اتصال و زمان هر اتصال را کسب و نگهداری می‌کند. ممیزی اتصالات یک ابزار ضروری برای کشف و خنثی کردن حملات مهاجمین است.
- هر مدول پروکسی یک بستهٔ نرم‌افزاری کوچک است که فقط برای مقاصد امنیت شبکه طراحی شده است. بعلت سادگی نسبی آن، کنترل کردن چنین مدول‌هائی برای کشف موارد نقض امنیت ساده‌تر است. بعنوان مثال، یک کاربرد معمول پستی در UNIX ممکن است شامل بیش از ۲۰,۰۰۰ خط برنامه باشد در حالی‌که یک پروکسی پستی ممکن است کمتر از ۱,۰۰۰ خط برنامه داشته باشد.
- هر پروکسی مستقل از سایر پروکسی‌های میزبان دژدار است. اگر مشکلی در عملکرد هر پروکسی پیش آید، و یا نقطهٔ آسیب‌پذیری در آن کشف گردد، می‌توان بدون اینکه عملیات کاربردی سایر پروکسی‌ها مختل شوند آن را از روی سیستم برداشت. همچنین اگر جمع کاربران برای سرویس جدیدی نیاز به حمایت داشته باشند، مدیر شبکه می‌تواند بسهولت پروکسی مورد نیاز را روی میزبان دژدار نصب نماید.
- یک پروکسی بجز در ابتدا، و برای خواندن فایل پیکربندی خود، معمولاً نیازی به دسترسی به دیسک ندارد. این امر کار را برای یک مهاجم که بخواهد sniffer اسب تروا و یا سایر فایل‌های خطرناک را روی میزبان دژدار نصب کند دشوار می‌سازد.
- هر پروکسی بصورت یک کاربر فاقد امتیاز، در یک شاخهٔ خصوصی و امن روی میزبان دژدار اجرا می‌شود.

پیکربندی‌های دیوار آتش

علاوه بر استفاده از یک پیکربندی ساده شامل یک سیستم منفرد، همانند مسیریاب فیلترکنندهٔ بسته‌ها و یا یک دروازهٔ منفرد (شکل ۱-۱۱)، پیکربندی‌های پیچیده‌تری نیز ممکن بوده و در واقع معمول‌ترند. شکل ۲-۱۱ سه نوع پیکربندی متداول دیوار آتش را نشان می‌دهد. هریک از آنها را بنوبت بررسی می‌کنیم.

در پیکربندی **دیوارآتش حفاظت‌کنندهٔ میزبان با یک دژ** (screened host firewall, single-homed bastion) (شکل ۲-۱۱الف)، دیوار آتش شامل دو سیستم است که یک مسیریاب فیلترکنندهٔ بسته‌ها و یک میزبان دژدار می‌باشند. معمولاً مسیریاب طوری پیکربندی می‌شودکه:

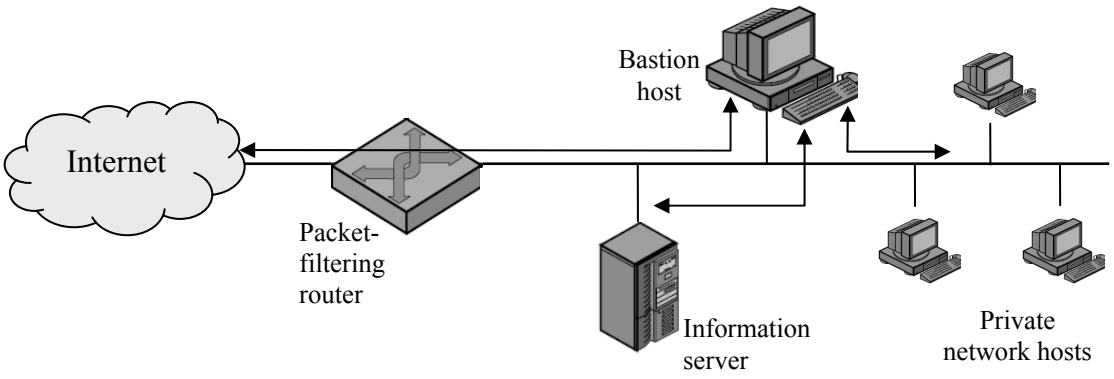
۱- برای ترافیکی که از سمت اینترنت وارد می‌شود، تنها بسته‌های IP که مقصد آنها میزبان دژدار است اجازهٔ ورود دارند.
۲- برای ترافیک خروجی از شبکهٔ داخلی، تنها بسته‌های IP که از میزبان دژدار صادر می‌شوند اجازهٔ خروج دارند.

میزبان دژدار عملیات احراز هویت و پروکسی را انجام می‌دهد. این پیکربندی دارای امنیت بیشتری از یک مسیریاب فیلترکنندهٔ بسته‌ها و یا یک دروازهٔ سطح کاربرد بوده و علت این امر دو چیز است. اول اینکه این پیکربندی هم فیلترینگ سطح بسته و هم فیلترینگ سطح کاربرد را انجام داده و قابلیت انعطاف قابل ملاحظه‌ای در تعریف خط‌مشی‌های امنیتی بوجود می‌آورد. دوم اینکه یک مهاجم قبل از اینکه بتواند امنیت شبکهٔ داخلی را به مخاطره اندازد، قاعدتاً بایستی از دو سیستم مجزا عبور کند.

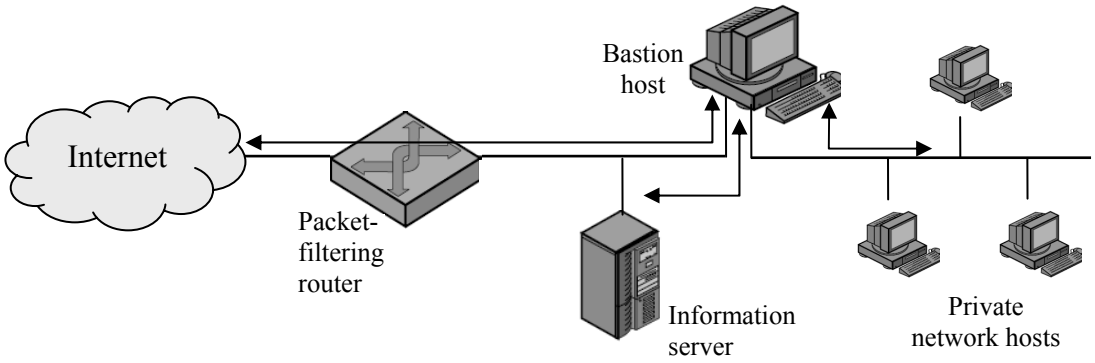
این پیکربندی همچنین انعطافی را در فراهم آوردن دسترسی مستقیم به اینترنت بوجود می‌آورد. بعنوان مثال، شبکهٔ داخلی ممکن است شامل یک سِرور اطلاعاتی عمومی همانند یک سِرور وِب بوده باشد که برای آن نیازی به اِعمال محدودیت‌های امنیتی سطح بالا نمی‌باشد. در چنین موردی مسیریاب را میتوان طوری پیکربندی کرد که ترافیک بین سِرور اطلاعاتی و اینترنت را مستقیماً عبور دهد.

در پیکربندی دیوار آتش با یک دژ که در بالا ذکر گردید، اگر مسیریاب فیلترکنندهٔ بسته‌ها کاملاً در اختیار مهاجم قرار گیرد ترافیک خواهد توانست مستقیماً از درون مسیریاب، بین اینترنت و سایر میزبان‌های متصل به شبکهٔ خصوصی برقرار گردد. پیکربندی **دیوارآتش حفاظت‌کنندهٔ میزبان با دژ دوگانه** (screened host firewall, dual-homed bastion) بطور فیزیکی از چنین عمل ضدامنیتی جلوگیری می‌کند (شکل ۲-۱۱ب). محاسن لایهٔ دوگانهٔ امنیتی که در پیکربندی قبل وجود داشت در اینجا نیز بجای خود باقی است. بازهم سِرور اطلاعاتی و یا سایر میزبان‌ها را میتوان مجاز به ارتباط مستقیم با مسیریاب نمود، البته اگر این امر در راستای خط‌مشی‌های امنیتی سیستم باشد.

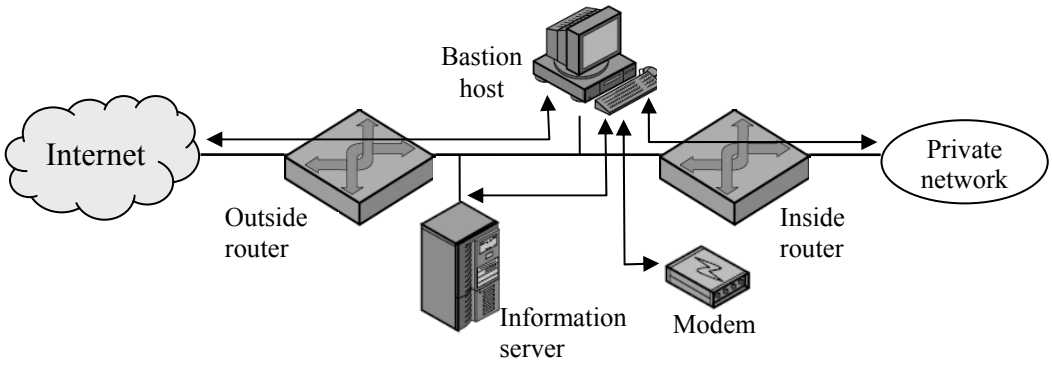
پیکربندی **دیوارآتش حفاظت‌کنندهٔ زیرشبکه** (screened subnet firewall) امن‌ترین نوع پیکربندی است (شکل ۲-۱۱ج). در این پیکربندی، از دو مسیریاب فیلترکنندهٔ بسته‌ها استفاده می‌شود که یکی بین میزبان دژدار و اینترنت و دیگری بین میزبان دژدار و شبکهٔ داخلی قرار دارد. این پیکربندی یک زیرشبکهٔ ایزوله را ایجاد می‌کند که ممکن است به سادگی شامل یک میزبان دژدار بوده ولی می‌تواند شامل یک یا چند سِرور اطلاعاتی و مُودم برای تماس تلفنی با اینترنت نیز باشد. معمولاً هم اینترنت و هم شبکهٔ داخلی به میزبان‌های روی زیرشبکهٔ حفاظت‌شده دسترسی داشته ولی ترافیک درعرض شبکهٔ حفاظت‌شده مسدود می‌شود. این پیکربندی محاسن متعددی دارد:



(الف) سیستم دیوار آتش Screened host (single-homed bastion host)



(ب) سیستم دیوار آتش Screened host (dual-homed bastion host)



(ج) سیستم دیوار آتش Screened-subnet

شکل ۲-۱۱ پیکربندی‌های دیوار آتش

- در اینجا از سه سدّ دفاعی در مقابل مهاجمین استفاده می‌شود.
- مسیریاب خارجی تنها حضور زیرشبکهٔ حفاظت‌شده را به اینترنت اعلام می‌دارد و بنابراین شبکهٔ داخلی برای اینترنت مرئی نمی‌باشد.
- بطریق مشابه، مسیریاب داخلی تنها حضور زیرشبکهٔ حفاظت‌شده را به شبکهٔ داخلی اطلاع می‌دهد و بنابراین سیستم‌های داخل شبکه نمی‌توانند ارتباط مستقیمی با اینترنت داشته باشند.

۱۱-۲ سیستم‌های معتمد (TRUSTED SYSTEMS)

یک روش برای بالابردن قابلیت دفاعی یک سیستم در مقابل مهاجمین و برنامه‌های بداندیش، پیاده‌سازی تکنولوژی سیستم معتمد است. در این بخش نگاه مختصری به این مقوله می‌اندازیم. مطالعه را با نگاهی به برخی مفاهیم اساسی کنترل دست‌یابی به داده‌ها آغاز می‌کنیم.

کنترل دست‌یابی به داده‌ها

پس از یک ورود موفق به سیستم، عملاً به کاربر اجازه داده شده است تا به یک یا چند میزبان و تعدادی برنامه‌های کاربردی دست یابد. این امر معمولاً برای سیستمی که شامل داده‌های گران‌قیمت در پایگاه داده خود می‌باشد، کافی نیست. از طریق روش‌های کنترل دست‌یابی کاربر، یک کاربر می‌تواند به سیستم معرفی گردد. در رابطه با هر کاربر، میتوان یک پروفایل از عملیات مجاز و دسترسی‌های ممکن به فایل‌ها را تعریف کرد. سیستم عامل آنگاه خواهد توانست بر اساس پروفایل کاربر، قوانینی را به نحوهٔ عملیات او اِعمال نماید. با وجود این، سیستم مدیریت پایگاه داده بایستی دست‌یابی به رکوردهای بخصوص و حتی بخشی از رکوردها را کنترل کند. مثلاً ممکن است برای همهٔ افراد یک سازمان دست‌یابی به فایلی که نام پرسنل آن سازمان در آن ضبط شده است ممکن بوده ولی تنها افراد بخصوصی بتوانند به اطلاعات حقوق و دستمزد آنها دست یابند. این مقدار، بیش از یک سطح از جزئیات را دربر می‌گیرد. در حالی که سیستم عامل ممکن است به یک کاربر اجازهٔ دست‌یابی به یک فایل و یا استفاده از یک برنامه را اعطا نماید، که قاعدتاً پس از آن کنترل‌های امنیتی دیگری اعمال نخواهد شد، سیستم مدیریت پایگاه داده بایستی در مورد هر بار تلاش برای دست‌یابی فرد تصمیمی اتخاذ نماید. تصمیم این مدیریت نه تنها به مشخصات کاربر وابسته بوده بلکه بستگی به این دارد که او به کدام بخش دیتا علاقه داشته و یا حتی اینکه او قبلاً به کدام بخش دست یافته است.

یک مدل عمومی کنترل دست‌یابی که بتوسط سیستم مدیریت پایگاه داده و یا مدیریت فایل به اجرا گذاشته می‌شود، **ماتریس دست‌یابی (access matrix)** (شکل ۳-۱۱الف) است. مؤلفه‌های اساسی مدل بقرار زیراند:

- **سوژه (subject)**: واحدی که قادر به دست‌یابی به موضوعات است. معمولاً مفهوم سوژه معادل یک پردازش است. هر کاربر یا برنامهٔ کاربردی در واقع با کمک یک سری عملیات که نمایشگر آن کاربر یا برنامه کاربردی هستند، به موضوع دست می‌یابد.
- **موضوع (object)**: هر چیزی که دست‌یابی به آن باید کنترل شود. مثال‌های این مورد شامل فایل‌ها یا بخش‌هائی از یک فایل، برنامه‌ها و قسمت‌هائی از حافظه می‌باشند.
- **حق دست‌یابی**: روشی است که بتوسط آن یک سوژه به یک موضوع دسترسی پیدا می‌کند. خواندن، نوشتن و اجراکردن یک برنامه مثال‌هائی از این دست می‌باشند.

یک محور ماتریس شامل سوژه‌های شناخته شده‌ای است که ممکن است برای دست‌یابی به موضوعی تلاش نمایند. نوعاً این لیست شامل کاربران منفرد و یا گروه کاربران است ولی دست‌یابی را میتوان برای ترمینال‌ها، میزبان‌ها و یا برنامه‌ها نیز بجای کاربران کنترل نمود. محور دیگر موضوعاتی که می‌توان به آنها دست یافت را مشخص می‌سازد. در جزئی‌ترین سطح دست‌یابی، موضوعات می‌توانند میدان‌های منفرد دیتا باشند. گروه‌های متشکل‌تر مانند رکوردها، فایل‌ها و یا حتی کل پایگاه داده نیز می‌تواند موضوع قابل دست‌یابی را تشکیل دهند. در محل تلاقی هر سطر و هر ستون ماتریس یا جدول، حق دست‌یابی مجاز از طرف آن کاربر به آن موضوع تعیین شده است.

در عمل یک ماتریس دست‌یابی دارای خانه‌های خالی زیادی بوده و بنابراین برای اجرای عملیاتی آن به یکی از دو روش زیر عمل می‌گردد. ماتریس را می‌توان به مجموعهٔ ستون‌های آن تجزیه نمود که در نتیجه **لیست‌های کنترل دست‌یابی (access control lists)** (شکل ۳-۱۱ب) ایجاد می‌شود. بنابراین یک لیست کنترل دست‌یابی، برای هر موضوع، کاربران را لیست نموده و حقوق دست‌یابی آنان را مشخص می‌سازد. لیست کنترل دست‌یابی ممکن است شامل یک پیش‌فرض برای عموم باشد. این امر به کاربرانی که نام آنها در لیست ذکر نشده است اجازه می‌دهد تا به مجموعه‌ای از حقوق از پیش تعیین شده برسند. لیست ممکن است شامل اسامی کاربران منفرد و یا گروه‌های کاربران باشد.

تجزیهٔ لیست به ردیف‌ها، **بلیت‌های توانائی (capability tickets)** را ایجاد می‌کند (شکل۳-۱۱ج). یک بلیت از این نوع، موضوعات مطرح و عملیات مجاز هر کاربر روی آنها را تعیین می‌کند. هر کاربر تعدادی بلیت دارد و ممکن است مجاز باشد تا آنها را به دیگران نیز قرض داده و یا اهدا کند. چون این بلیت‌ها را می‌توان در جاهای مختلف سیستم خرج کرد، آنها می‌توانند مشکلات امنیتی جدی‌تری را نسبت به لیست‌های کنترل دست‌یابی ایجاد نمایند. علی‌الخصوص، بلیت بایستی غیرقابل جعل باشد. یکی از راه‌های اجرائی این است که از سیستم عاملی استفاده نمود که تمام بلیت‌ها را، بجای نزد کاربر، نزد خود نگاه دارد. این بلیت‌ها بایستی در ناحیه‌ای از حافظه نگهداری شوند که بتوسط کاربران قابل دسترس نباشد.

مفهوم سیستم‌های معتمد

بیشتر آنچه تا بحال مورد بحث قرار گرفته است، مربوط به حفاظت یک پیام و یا مقوله‌ای نظیر آن در برابر حملهٔ فعال و یا غیرفعال یک کاربر مشخص بوده است. نیاز متفاوت دیگری که بطور گسترده مورد استفاده است، حفاظت داده‌ها یا منابع بر اساس سطوح متفاوت امنیتی است. این امر را معمولاً در ارتش، که در آنجا اطلاعات به انواع طبقه‌بندی نشده (U)، محرمانه (C)، سری (S)، فوق سری (TS) و بالاتر از آن دسته‌بندی شده‌اند، می‌توان مشاهده کرد. این مفهوم در سایر زمینه‌ها نیز می‌تواند بکار گرفته شده و اطلاعات را به دسته‌های متفاوت تقسیم کرد. برای هر کاربر نیز می‌توان دست‌یابی مجاز به دسته‌ای از اطلاعات و ممنوعیت دست‌یابی به بخش‌های دیگر را تعیین نمود. بعنوان مثال، بالاترین سطح امنیت ممکن است متعلق به اسناد و داده‌های مربوط به سیاست‌های استراتژیک یک سازمان باشدکه فقط بایستی در دسترس هیئت مدیرهٔ سازمان و عوامل او قرار گیرند. در ردهٔ بعدی ممکن است اسناد حساس مالی و اطلاعات مربوط به پرسنل قرار داشته باشدکه فقط بایستی در دسترس پرسنل مدیریت امور اداری و مالی و عوامل آنها قرار گیرند

وقتی دسته‌ها و یا سطوح متفاوت داده‌ها مطرح می‌شوند، نیاز امنیتی آنها را **امنیت چند سطحه (multilevel security)** خوانند. بیان عمومی مساله امنیت چندسطحه بدین ترتیب است که یک کاربر سطح بالاتر ممکن نیست اطلاعات را به یک کاربر سطح پائین‌تر و یا سطح غیرقابل مقایسه با خود بدهد مگر اینکه این امر با دقت کامل از ارادهٔ یک کاربر معتبر ناشی شده باشد. برای اجرای این هدف، این نیاز به دو بخش تقسیم شده و بسادگی بیان گردیده است. یک سیستم امن چند سطحه بایستی دو قانون زیر را رعایت نماید:

	Program1	. . .	Segment A	Segment B
Process 1	Read Execute		Read Write	
Process 2				Read
.				
.				
.				

(الف) ماتریس دست‌یابی

Access control list for Program 1: Process1 (Read,Execute)
Access control list for segment A: Process1 (Read,Write)
Access control list for Segment B: Process2 (Read)

(ب) لیست کنترل دست‌یابی

Capability list for Process1: Program1 (Read,Execute) Segment A (Read,Write)
Capability list for Process2: Segment B (Read)

(ج) لیست توانائی‌ها

شکل ۳-۱۱ ساختار کنترل دست‌یابی

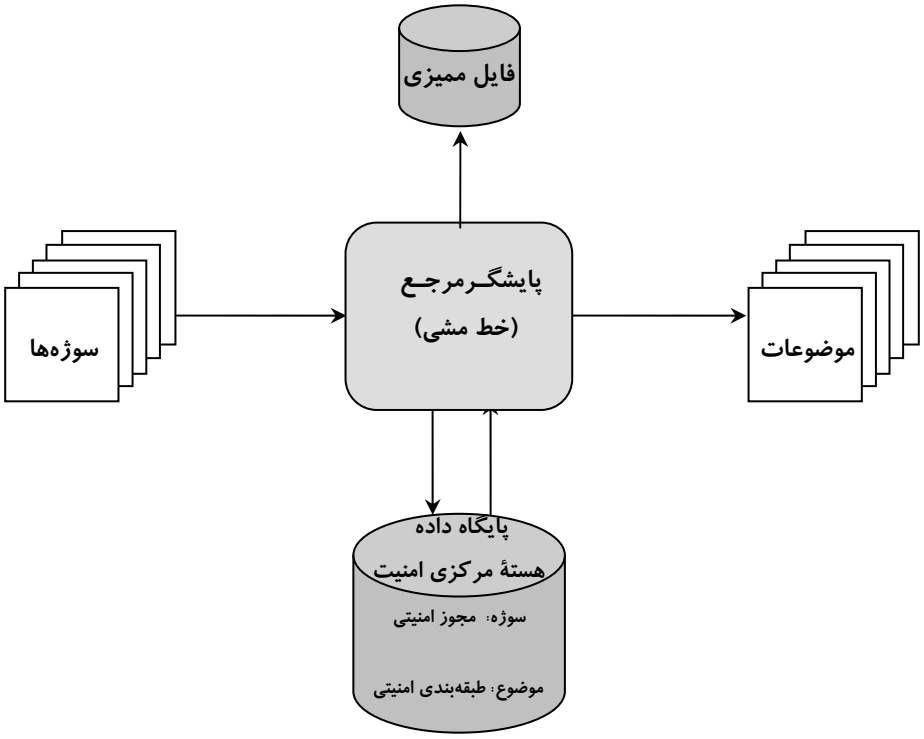
- **نخواندن سطح بالاتر**: یک سوژه تنها می‌تواند موضوعی را بخواند که در سطح او و یا پائین‌تر از سطح او قرار داشته باشد. این امر را معمولاً **خاصیت امنیتی ساده (Simple Security Property)** گویند.
- **ننوشتن در سطح پائین‌تر**: یک سوژه تنها می‌تواند در موضوعی دخل تصرف نماید که در سطح او و یا بالاتر از سطح او قرار داشته باشد. در متون امنیتی این امر را ***-Property** (star property) گویند.

اگر این دو قاعده بطور صحیح بکار گرفته شوند، امنیت چند سطحه ایجاد خواهد شد. برای یک سیستم پردازش داده‌ها، روشی که بکار گرفته شده و موضوع بسیاری از تحقیقات بوده است، استفاده از *پایشگر مرجع* (*reference monitor*) است. این روش برخورد با مسأله در شکل ۴-۱۱ نشان داده شده است. پایشگر مرجع یک واحد کنترل‌کنندهٔ در سخت‌افزار و سیستم عامل یک کامپیوتر بوده و نحوهٔ دست‌یابی کاربران به موضوعات را بر مبنای پارامترهای امنیتی سوژه و موضوع قانون‌مند می‌نماید. پایشگر مرجع به فایلی به نام *پایگاه داده هستهٔ مرکزی امنیت* (*security kernel database*) دسترسی داشته که در آن امتیازات دست‌یابی هر سوژه و سطوح حفاظتی هر موضوع لیست شده است. پایشگر مرجع قواعد امنیتی (نخواندن سطح بالاتر و ننوشتن در سطح پائین‌تر) را به اجرا گذاشته و دارای خصوصیات زیر است:

- **وساطت کامل**: قواعد امنیتی در هربار دست‌یابی، و نه مثلاً فقط زمان بازشدن یک فایل، به اجرا گذاشته می‌شوند.
- **ایزولاسیون**: پایشگر مرجع و پایگاه داده در مقابل دستکاری‌های غیرمسئولانه حفاظت می‌شوند.
- **قابلیت تائید**: صحت عمل پایشگر مرجع بایستی قابل اثبات باشد. یعنی بایستی بتوان از نظر ریاضی ثابت نمود که پایشگر مرجع قواعد امنیتی را به اجرا گذاشته و دخالت کامل و ایزولاسیون را بوجود می‌آورد.

شرایط بالا بسیار محکم‌اند. نیاز به وساطت کامل بدین معنی است که هر دست‌یابی به دیتا در حافظهٔ اصلی و روی دیسک و نوار، بایستی با واسطه باشد. اگر قرار باشد که این امر صرفاً از طریق نرم‌افزار انجام شود، پنالتی عملکرد قابل تحمل نخواهد بود و بنابراین حداقل بخشی از راه حل مساله را بایستی در سخت‌افزار اجرا نمود. نیاز به ایزولاسیون بدین معنی است که یک مهاجم هرچقدر زرنگ باشد نبایستی بتواند منطق پایشگر مرجع و یا محتویات پایگاه داده هستهٔ مرکزی امنیت را تغییر دهد. بالاخره نیاز به اثبات ریاضی مطلب برای سیستمی پیچیده همانند یک کامپیوتر با قابلیت‌های عام، کاری فوق تصور است. سیستمی که بتواند چنین قابلیت‌هائی را فراهم آورد یک **سیستم معتمد (trusted system)** خوانده می‌شود.

یک عنصر نهائی که در شکل ۴-۱۱ نشان داده شده است، یک فایل ممیزی (audit) است. پیشامدهای مهم امنیتی همانند نقض مقررات امنیتی و همچنین تغییرات مجاز در پایگاه داده هستهٔ مرکزی امنیت، در فایل ممیزی ذخیره می‌شود.



شکل ۴-۱۱ مفهوم پایشگر مرجع

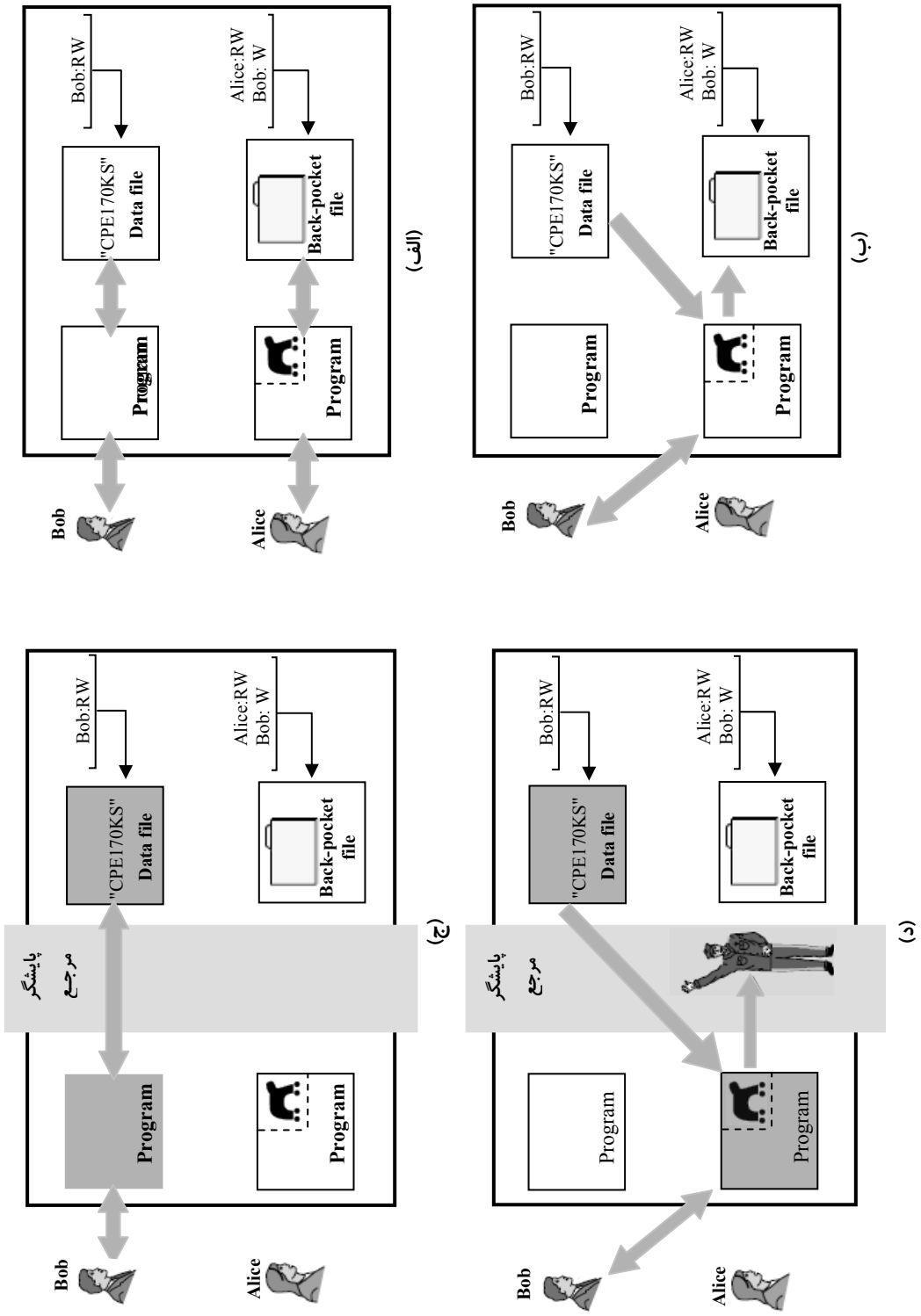
در تلاش برای تأمین نیازهای خود و همچنین برای ایجاد یک روش عـام، وزارت دفـاع امریکـا در سـال ۱۹۸۱ مرکـز امنیت کامپیوتر(Computer Security Center) در آژانس امنیت ملـی (NSA) را بـا هـدف تـشویق بـه ایجـاد گـستردهٔ سیستم‌های کامپیوتری معتمد تأسیس نمود. این هدف، از طریق خلق یک برنامهٔ ارزیابی محصولات تجاری دنبال گردید. بطـور خلاصه، این مرکز تلاش میکند تا محصولات تجاری موجود را از نظر برآوردن لازمه‌های امنیتی فوق‌الذکر ارزیابی نماید. مرکز، محصولات ارزیابی شده را برحسب نوع مشخصه‌های امنیتی فراهم آمده طبقه‌بندی می‌نماید. این ارزیابی‌ها در راسـتای اهـداف وزارت دفاع امریکا بوده ولی برای عموم هم انتشار یافته و در دسترس می‌باشند. بنـابراین ایـن اسـناد مـی‌تواننـد بـرای خریـد محصولات تجاری موجود در بازار مورد استفادهٔ مشتریان تجاری نیز قرار گیرند.

دفاع اسب تروا (Trojan Horse)

یکی از راه‌های ایجاد امنیت در مقابل حملات اسب تروا، استفاده از یک سیستم عامل امن و معتمد است. شکل ۵-۱۱ مثالی از این دست را نشان می‌دهد. در این مورد از یک اسب تروا برای دور زدن مکانیسم امنیتی استانداردی که بتوسط بیشتر مدیریت‌های فایل و سیستم‌های عامل بکار گرفته می‌شود، یعنی لیست کنترل دست‌یابی، استفاده شده است. در این مثال کاربری بنام Bob از طریق یک برنامه، به یک فایل داده که شامل دنباله‌های بسیار مهم و حساس "CPE170KS" است دسترسی دارد. Bob فایل را چنان خلق کرده است که اجازهٔ نوشتن/خواندن را تنها به برنامه‌هائی که بتوسط خودش اجرا می‌شود می‌دهد، یعنی تنها پردازش‌هائی که متعلق به Bob هستند ممکن است به فایل دست‌یابند.

حملهٔ اسب تروا زمانی آغاز می‌شود که یک کاربر مهاجم بنام Alice با دست‌یابی مجاز به سیستم راه یافته و یک برنامه اسب تروا به همراه یک فایل خصوصی که قرار است در حمله بعنوان جیب مخفی بکار رود را در سیستم نصب می‌کند. Alice برای این فایل بخودش اجـازه نوشتن/خواندن را داده، در حالیکه برای Bob تنها اجـازهٔ نوشتن را می‌دهد (شکل ۵-۱۱الف). حال Alice، Bob را وسوسه کرده تا برنامهٔ اسب تروا را بکار گیرد و شاید این کار را با تبلیغ اینکه این برنامه یک ابزار کاربردی خوب است انجام دهد. وقتی برنامه تشخیص می‌دهد که دارد بتوسط Bob اجرا می‌شود، دنبالهٔ کاراکترهای سرّی را از فایل Bob خوانده و آن را در فایل جیب مخفی Alice کپی می‌کند (شکل ۵-۱۱ب). هردو عملیات خواندن و نوشتن از محدودیت‌هائی که بتوسط لیست دست‌یابی تعیین شده است تبعیت می‌کنند. تنها کاری که برای Alice باقی مانده است این است که در زمان دیگری به فایل Bob دست یافته و دنباله را بخواند.

حال به استفاده از یک سیستم عامل امن در این سناریو توجه کنید (شکل ۵-۱۱ج). سطوح امنیتی در هنگـام اتـصال بـه سیستم به سوژه‌ها تخصیص می‌یابند و مبنای عمل، مواردی همچون نوع ترمینالی که بکار گرفته می‌شود و مشخصات کاربر بـر اساس ID و کلمه عبور او است. در این مثال دو سطح امنیتی حساس و عمومی وجود دارد که بنحوی سطح‌بندی شـده‌انـد کـه سطح حساس بالاتر از سطح عمومی قرار دارد. به پردازش‌های مربوط به Bob و فایل دیتای او سطح امنیتی حساس تخـصیص داده شده است. فایل Alice و پردازش‌های مربوط به او در سطح عمومی که پـائین‌تـر اسـت قـرار دارنـد. اگـر Bob برنامـهٔ بداندیش اسب تروا را بکار گیرد (شکل ۵-۱۱د)، آن برنامه سطح امنیتی Bob را استعلام می‌کند و بنابراین او قادر خواهد بـود بر اساس خاصیت امنیتی ساده، دنبالهٔ کاراکترهای مهم را مشاهده نماید. وقتی برنامه بـرای ذخیـرهٔ ایـن دنبالـه در یـک فایـل عمومی تلاش کند (فایل جیب مخفی)، Property-* نقض گردیده و پایشگر مرجع جلوی این کار را می‌گیـرد. بنـابراین تـلاش برای نوشتن دنباله در فایل جیب مخفی، حتی اگر لیست کنترل دست‌یابی آن را مجاز بشمارد، بی‌نتیجه مـی‌مانـد. حاصـل ایـن است که خط‌مشی امنیتی بر مکانیسم کنترل دست‌یابی، توفق می‌یابد.



شکل ۵-۱۱ اسب تروا و سیستم عامل امن

۳-۱۱ معیارهای مشترک برای ارزیابی امنیت تکنولوژی اطلاعات

کارهای انجام شده بتوسط آژانس امنیت ملّی و سایر سازمان‌های دولتی امریکا برای تعیین لازمه‌ها و معیارهای ارزیابی سیستم‌های معتمد، همگام با کارهای مشابه در سایر کشورها بوده است. معیارهای مشترک Common Criteria(CC) برای تکنولوژی اطلاعات و ارزیابی امنیت، یک ابتکار بین‌المللی از سوی بخش‌های استاندارد کشورهای مختلف برای تعیین لازمه‌های امنیتی و تعریف معیارهای ارزیابی بوده است.

لازمه‌ها

CC یک مجموعه مشترک از لازمه‌های امنیتی برای استفاده در ارزیابی‌ها را تعریف می‌کند. اصطلاح **هدف ارزیابی** target of evaluation (TOE) به بخشی از محصول و یا سیستم که تحت ارزیابی قرار می‌گیرد اشاره می‌کند. لازمه‌ها در دو طبقه قرار دارند:

- **لازمه‌های عملیاتی**: اینها رفتارهای مطلوب امنیتی را تعریف می‌کنند. اسناد CC مجموعه‌ای از مؤلفه‌های عملیاتی امنیتی را ایجاد می‌نمایندکه روش استانداردی برای بیان لازمه‌های عملیاتی امنیت در یک TOE است.
- **لازمه‌های اطمینان‌بخشی**: اینها مبانی کسب اطمینان از این است که معیارهای امنیت مؤثر بوده و بطرز صحیحی پیاده‌سازی شده است. اسناد CC مجموعه‌ای از مؤلفه‌های اطمینان‌بخش را ایجاد می‌نمایند که روش استانداردی برای بیان لازمه‌های اطمینان‌بخشی در یک TOE است.

هم لازمه‌های عملیاتی و هم لازمه‌های اطمینان‌بخشی در طبقات مختلف سازمان‌دهی می‌شوند: یک **طبقه** عبارت از مجموعه‌ای از لازمه‌هاست که دارای تمرکز و یا منظور خاصی است. جداول ۳-۱۱ و ۴-۱۱ بطور مختصر طبقات مربوط به لازمه‌های عملیاتی و لازمه‌های اطمینان‌بخشی را تعریف می‌کنند. هریک از این طبقات شامل تعدادی خانواده‌اند. لازمه‌های درون هر **خانواده** اهداف امنیتی خاصی داشته ولی از جهت تأکید یا قوت با هم متفاوت‌اند. بعنوان مثال طبقه ممیزی شامل شش خانواده است که مربوط به جنبه‌های مختلف ممیزی است (مثل تولید اطلاعات ممیزی، تحلیل ممیزی و ذخیره کردن پیشامدهای مرتبط با ممیزی). هر خانواده بنوبهٔ خود شامل یک یا دو مؤلفه است. یک **مؤلفه**، مجموعه‌ای مشخص از لازمه‌های امنیتی را توصیف کرده و کوچک‌ترین مجموعهٔ قابل انتخاب از لازمه‌های امنیتی برای شمول در ساختار تعریف شده در CC است.

برای مثال، طبقهٔ پشتیبانی رمزنگاری از لازمه‌های عملیاتی، شامل دو خانواده است: مدیریت کلید رمزنگاری و تابع رمزنگاری. در زیر خانوادهٔ مدیریت کلید رمزنگاری، چهار مؤلفه قرار دارد که برای تعیین الگوریتم تولید کلید و اندازهٔ کلید، روش توزیع کلید، روش دست‌یابی به کلید و روش نابودی کلید بکار می‌رود. برای هر مؤلفه، ممکن است یک استاندارد تعیین نمود. در تحت خانوادهٔ تابع رمزنگاری تنها یک مؤلفهٔ منفرد قرار دارد که یک الگوریتم و اندازهٔ کلید آن را برحسب یک استاندارد مشخص می‌سازد.

مجموعه‌های مؤلفه‌های عملیاتی و اطمینان‌بخشی ممکن است با هم گرد آمده تا بسته‌های قابل استفاده مجدد جهت برآورده نمودن اهداف مشخصی را تشکیل دهند. مثالی از چنین بسته‌هائی می‌تواند مجموعه‌ای از مؤلفه‌های عملیاتی لازم برای کنترل‌های دست‌یابی منصفانه (Discretionary Access Control) باشد.

جدول ۳-۱۱ لازمه‌های عملیاتی امنیت CC

طبقه	توصیف
ممیزی	شامل شناسائی، ثبت، ذخیره‌سازی و تحلیل اطلاعات مرتبط با فعالیت‌های امنیتی است. سوابق ممیزی بتوسط این عملیات تولیدشده و می‌توانند از نظر ارتباط با مسائل امنیت مورد مطالعه قرار گیرند.
پشتیبانی رمزنگاری	زمانی که TOE توابع رمزنگاری را پیاده‌سازی می‌کند، مورد استفاده قرار می‌گیرند. اینها برای مثال ممکن است برای حمایت از ارتباطات، شناسائی و تصدیق هویت، یا جداسازی داده‌ها بکار روند.
ارتباطات	دو خانواده که مرتبط با عدم انکار بتوسط خلق‌کنندهٔ دیتا و دریافت‌کنندهٔ دیتا است را ایجاد می‌کند.
حفاظت داده‌های کاربر	لازمه‌های مرتبط با حفاظت از داده‌های کاربر در TOE درخلال ورود، خروج و ذخیره‌سازی را برآورده نموده و علاوه بر این مشخصه‌های امنیتی مرتبط با دیتای کاربر را تعیین می‌کند.
شناسائی و احراز هویت	شناسائی بدون ابهام کاربران معتبر و ارتباط صحیح مشخصه‌های امنیتی با کاربران و سوژه‌ها را اطمینان می‌دهد.
مدیریت امنیت	مدیریت مشخصه‌های امنیتی، داده‌ها و توابع را تعیین می‌کند.
سرّی بودن	یک کاربر را با حفاظت در برابر کشف و سوءاستفاده از هویت او از سوی دیگر کاربران تجهیز می‌کند.
حفاظت از عملیات امنیتی TOE	بر حفاظت از داده‌های TSF (عملیات امنیتی TOE)، و نه داده‌های کاربر، تمرکز دارد. این طبقه مرتبط با صحت و مدیریت مکانیسم‌ها و داده‌های TOE است.
بهره گیری از منابع	در دسترس بودن منابع لازم همچون توانائی پردازش و ظرفیت ذخیره‌سازی را پشتیبانی می‌کند. شامل لازمه‌های مربوط به تحمل خطا، اولویت سرویس و تخصیص منابع است.
دست‌یابی به TOE	لازمه‌های عملیاتی، علاوه بر آنهائی که برای شناسائی و احراز هویت تعیین شده‌اند، برای کنترل و برقراری یک اجلاس کاربر را تأمین می‌نماید. شامل لازمه‌های دست‌یابی TOE به اموری همچون محدودکردن تعداد و کیفیت اجلاس‌های کاربر، نمایش تاریخچهٔ دست‌یابی و دستکاری پارامترهای دست‌یابی است.
مسیرها/کانال‌های معتمد	در رابطه با مسیرهای ارتباطی مورد اعتماد بین کاربرها و TSFها، و بین خود TSFها عمل می‌کند.

پروفایل‌ها و هدف‌ها

CC همچنین دو نوع از اسناد که می‌توانند بر حسب لازمه‌های تعریف شده در CC تولید شوند، را تعریف می‌کند.

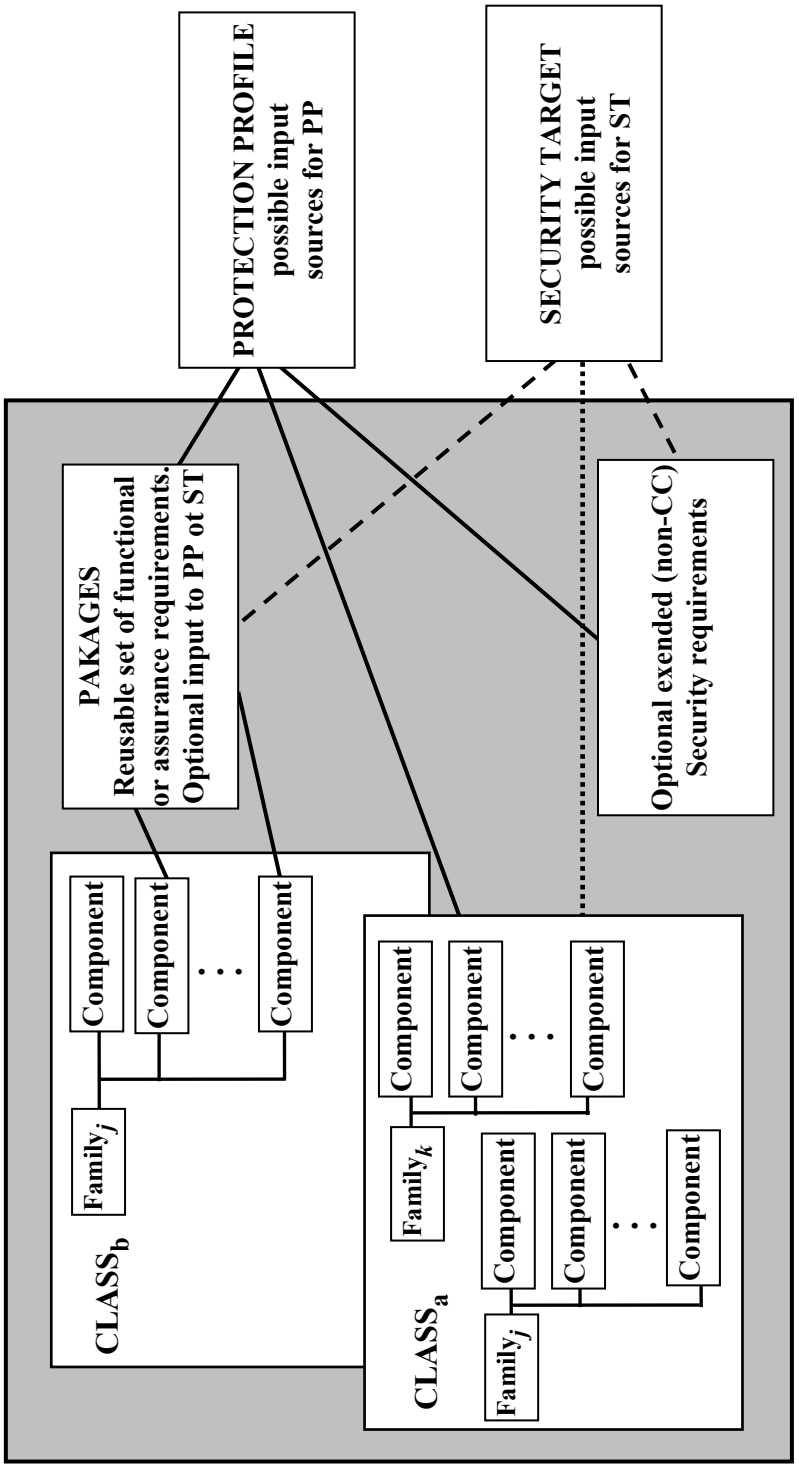
- **پروفایل‌های حفاظتی(PP)**: یک مجموعه از لازمه‌ها و اهداف امنیتی مستقل از پیاده‌سازی برای یک گروه از محصولات و یا سیستم‌ها که نیازهای مشابهی در زمینهٔ امنیت IT دارند را تعریف می‌کند. هدف از یک PP این است که قابل استفادهٔ مجدد بوده و لازمه‌هائی که از نظر برآوردن اهداف مشخصی مفید و مؤثر بنظر می‌رسند را تعریف کند. هدف از PP، پشتیبانی از تعریف استانداردهای عملیاتی و کمک به فرموله کردن مشخصات بوده است. PP منعکس‌کنندهٔ نیازهای امنیتی کاربر است.

جدول ۴-۱۱ لازمه‌های اطمینان بخشی امنیت CC

طبقه	توصیف
مدیریت پیکربندی	نیازمند به این است که سلامت TOE بطرز مناسبی حفظ گردد. علی‌الخصوص مدیریت پیکربندی این اطمینان را ایجاد می‌کند که TOE و اسناد استفاده شده برای ارزیابی همان‌هائی هستند که برای توزیع آماده شده‌اند.
تحویل و عملیات	مرتبط با معیارها، رَویه‌ها و استانداردها برای تحویل امن، نصب و استفادهٔ عملیاتی از TOE برای اطمینان‌بخشی از این مطلب است که حفاظت امنیتی ایجادشده بتوسط TOE در خلال این پیشامدها نقض نگردد.
توسعه	مرتبط با پالایش فرم TSF از فرم تعریف شده در ST تا مرحلهٔ پیاده‌سازی بوده و یک نگاشت از لازمه‌های امنیتی به پائین‌ترین سطح ارائه است.
اسناد راهبردی	مرتبط با استفادهٔ عملیاتی امن از TOE بتوسط کاربران و مدیران است.
پشتیبانی از دورهٔ حیات	مرتبط با دورهٔ حیات TOE بوده و شامل تعریف دورهٔ حیات، ابزارها و تکنیک‌ها، امنیت محیط پیاده‌سازی و رفع اشکالاتی است که بتوسط کاربران در TOE یافت می‌شود.
آزمایشات	مرتبط با نمایش این امر است که TOE لازمه‌های عملیاتی خود را اجرا می‌کند. خانواده‌ها در این کلاس مرتبط با اندازه‌گیری پوشش و عمق تست‌های اجرائی و تست‌های مستقل دیگرند.
ارزیابی آسیب‌پذیری	لازمه‌هائی را تعریف می‌کند که هدف آن شناسائی آسیب‌پذیری‌های قابل سوءاستفاده است که می‌تواند ناشی از ساخت، عملکرد و یا پیکربندی ناصحیح TOE باشد. خانواده‌هائی که در اینجا تعریف می‌شوند نگران شناسائی‌های آسیب‌پذیری‌ها در تحلیل یک کانال پنهان، تحلیل پیکربندی TOE، آزمایش توانائی مکانیسم‌های امنیتی، و شناسائی خطاهائی است که در خلال ایجاد TOE بوجود آمده‌اند. خانوادهٔ دوم، طبقه‌بندی امنیتی مؤلفه‌های TOE را پوشش می‌دهد. سومی و چهارمی تحلیل تغییرات در اثر یک ضربهٔ امنیتی و ایجاد شواهد برای رعایت رَویه‌ها است. این کلاس فراهم‌آورندهٔ بلوک‌های سازندهٔ استقرار روش‌های حفظ اطمینان‌بخشی است.
حفظ اطمینان‌بخشی	لازمه‌هائی که بایستی پس از تأئید یک TOE در برابر یک CC اعمال شوند را فراهم می‌آورد. این لازمه‌ها این هدف را دنبال می‌کنند که اطمینان دهند که TOE در صورت اعمال تغییراتی به آن و یا محیطش، عملیاتی باقی خواهد ماند.

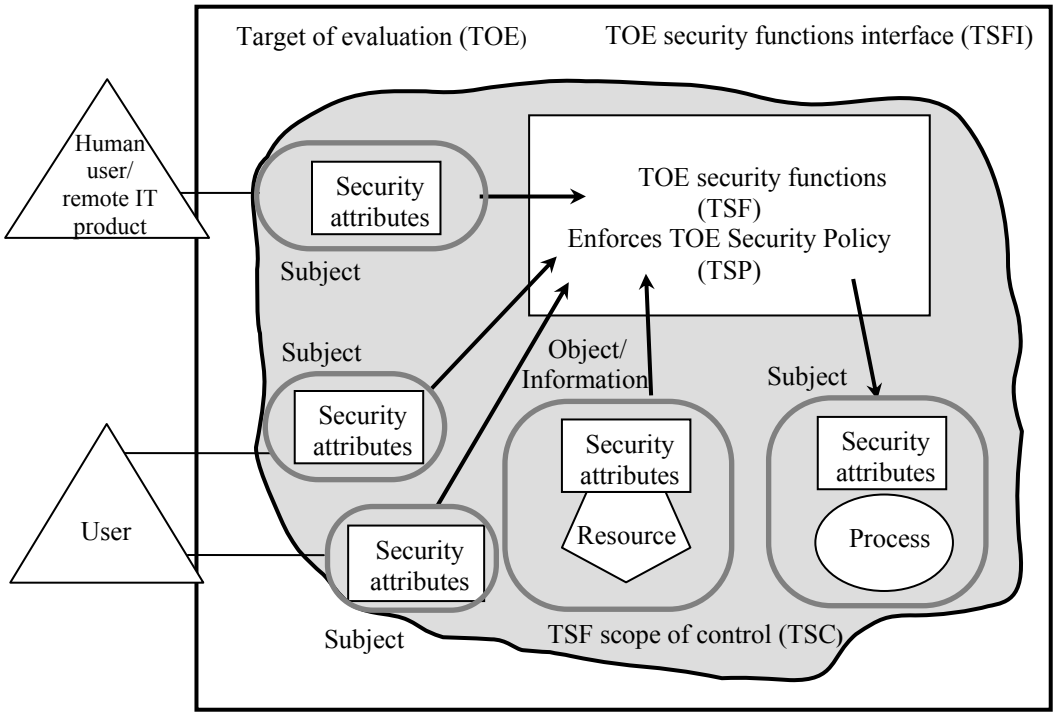
- **اهداف امنیتی (ST)**: شامل اهداف امنیتی IT و لازمه‌های یک TOE مشخص شناسائی شده بوده و معیارهای عملیاتی و اطمینان‌بخش پیشنهادشده بتوسط آن TOE برای دست‌یابی به لازمه‌های یادشده را تعریف می‌کند. ST ممکن است با یک یا چند PP مطابقت داشته و مبنای یک ارزیابی را تشکیل دهد. ST بتوسط یک فروشنده یا تولیدکننده فراهم می‌شود.

شکل ۶-۱۱ رابطهٔ بین لازمه‌ها از یک سو و پروفایل‌ها و هدف‌ها از سوی دیگر را نشان می‌دهد. برای یک PP، یک کاربر می‌تواند تعدادی از مؤلفه‌ها را انتخاب کرده تا لازمه‌های محصول مطلوب خود را تعریف نماید. کاربر همچنین می‌تواند به بسته‌های از قبل تعریف شده‌ای مراجعه کندکه تعدادی از لازمه‌های معمول یادشده در اسناد یک محصول را گردهم آورده است. بطریق مشابه، یک تولیدکننده یا طراح می‌تواند تعدادی از مؤلفه‌ها یا بسته‌ها را انتخاب کرده و یک ST را تعریف کند.



شکل ۶-۱۱ سازمان و ساختار لازمه‌های معیارهای مشترک

شکل ۷-۱۱ آنچه که در اسناد CC، پارادایم لازمه‌های عملیاتی امنیت خوانده می‌شود، را نشان می‌دهد. در واقع این شکل بر اساس مفهوم پایشگر مرجع بنا شده، اما از اصطلاحات و فلسفهٔ طراحی CC استفاده می‌کند.



شکل ۷-۱۱ پارادایم لازمه‌های عملیاتی امنیت

۱۱-۴ منابع مطالعاتی

یک منبع کلاسیک برای مطالعهٔ دیوارهای آتش [CHAP00] است. منبع کلاسیک دیگری که اخیراً بروزرسانی شده است [CHES03] است. [OPPL97]، [LODI98] و [BELL94b] مقالات خوبی برای مرور مطلب هستند. [WACK02] یک مرور عالی بر تکنولوژی دیوار آتش و سیاست‌های مرتبط با آن است. [AUDI04] و [WILS05] مباحث خوبی در مورد دیوارهای آتش دارند.

[GASS88] یک بررسی تفصیلی از سیستم‌های کامپیوتری معتمد را ارائه می‌دهد. [PFLE03] و [GOLL99] نیز این مقوله را پوشش می‌دهند. [FELT03] و [OPPL05] بحث‌های مفیدی در بارهٔ محاسبات مورد اعتماد دارند.

AUDI04 Audin, G. "Next-Gen Firewalls: What to Expect." *Business Communications Review*, June 2004.

BELL94b Bellovin, S., and Cheswick, W. "Network Firewalls." *IEEE Communications Magazine*, September 1994.

CHAP00 Chapman, D., and Zwicky, E. *Building Internet Firewalls*. Sebastopol, CA: O'Reilly,2000.

CHES03 Cheswick, W., and Bellovin, S. *Firewalls and Internet Security: Repelling the Wily Hacker*. Reading, MA: Addison-Wesley, 2003.

FELT03 Felten, E. "Understanding Trusted Computing: Will Its Benfits Outweigh Its Drawbacks?" *IEEE Society and Privacy*, May/June 2003.
GASS88 Gasser, M. *Building a Secure Computer System*. New York: Van Nostrand Reinhold, 1988.
GOLL99 Gollmann, D. *Computer Security*. New York: Wiley, 1999.
LODI98 Lodin, S., and Schuba, C. "Firewalls Fend Off Invasions from the Net." *IEEE Spectrum*, February 1998.
OPPL97 Oppliger, R. "Internet Security: Firewalls and Beyond." *Communications of the ACM*, May 1997.
OPPL05 Oppliger, R., and Rytz, R. "Does Trusted Computing Remedy Computer Security Problems?" *IEEE Security and Privacy*, March/April 2005.
PFLE03 Pfleeger, C. *Security in Computing*. Upper Saddle River, NJ: Prentice Hall, 2003.
WACK02 Wack, J.; Cutler, K.; and Pole, J. *Guidelines on Firewalls and Firewall Policy*. NIST Special Publication SP 800-41, January 2002.
WILS05 Wilson, J. "The Future of the Firewall."*Business Communications Review*, May 2005.

وِب سایت‌های مفید

- **Firewall.com**: لینک‌های بسیار به مراجع دیوارهای آتش و منابع نرم‌افزاری.
- **Trusted Computing Group**: گروه سازندگانی که درگیر ساخت و توسعهٔ استانداردهای کامپیوتری هستند. سایت شامل مقالات، مشخصه‌هاو لینک‌هائی به سازندگان است.
- **Common Criteria Portal**: وِب‌سایت رسمی پروژهٔ معیارهای مشترک.

۱۱-۵ واژه‌های کلیدی، سؤالات مرورکنندهٔ بحث و مسائل

واژه‌های کلیدی

دیوار آتش	firewall
امنیت چند سطحه	multilevel security
موضوع	object
مسیریاب فیلترکنندهٔ بسته‌ها	packet- filtering router
پایشگر مرجع	reference monitor
دیوارآتش تفتیش‌کننده حالت	stateful inspection firewall
سوژه	subject
سیستم معتمد	trusted system
لیست کنترل دست‌یابی	access control list (ACL)
ماتریس دست‌یابی	access matrix
حق دست‌یابی	access right
دروازهٔ سطح کاربرد	application- level gateway
میزبان دژدار	bastion host
بلیت توانائی	capability ticket
دروازهٔ سطح مدار	circuit- level gateway
معیارهای مشترک	common criteria(CC)

سؤالات مرورکننده بحث

۱۱-۱ سه هدف اصلی طراحی یک دیوار آتش را نام ببرید.

۱۱-۲ چهار تکنیک مختلف که بتوسط دیوارهای آتش برای کنترل دست‌یابی و استقرار نظام امنیتی استفاده می‌شود، را نام ببرید.

۱۱-۳ چه اطلاعاتی از سوی یک مسیریاب فیلترکنندهٔ بسته‌ها مورد استفاده قرار می‌گیرد؟

۱۱-۴ برخی از نقاط ضعف یک مسیریاب فیلترکنندهٔ بسته‌ها کدام است؟

۱۱-۵ فرق بین یک مسیریاب فیلترکنندهٔ بسته‌ها و یک دیوار آتش تفتیش‌کنندهٔ حالت چیست؟

۱۱-۶ یک دروازهٔ سطح مدار چیست؟

۱۱-۷ یک دروازهٔ سطح کاربرد چیست؟

۱۱-۸ تفاوت‌های سه پیکربندی شکل ۲-۱۱ کدام است؟

۱۱-۹ در مقولهٔ کنترل دست‌یابی، فرق بین یک سوژه و یک موضوع چیست؟

۱۱-۱۰ فرق بین یک لیست کنترل دست‌یابی و یک بلیت توانائی چیست؟

۱۱-۱۱ دو قانونی که یک پایشگر مرجع اعمال می‌کند، کدام‌اند؟

۱۱-۱۲ یک پایشگر مرجع چه خواصی باید داشته باشد؟

۱۱-۱۳ معیارهای مشترک چیستند؟

مسائل

۱۱-۱ همانطور که دربخش ۱-۱۱ ذکر گردید، یکی از روش‌های شکست دادن حملهٔ فِرگمنت‌های کوچک این است که یک طول حداقل از سرآیند حمل‌ونقل را مجبور سازیم تا در اولین فِرگمنت بستهٔ IP قرار گیرد. اگر اولین فِرگمنت پذیرفته نشود، همهٔ فِرگمنت‌های دیگر نیز می‌توانند پذیرفته نشوند. از سوی دیگر، ماهیت IP چنان است که فِرگمنت‌ها می‌توانند خارج از نظم دریافت شوند. بنابراین یک فِرگمنت میانی ممکن است قبل از اینکه فِرگمنت اولیه برگشت داده شود از فیلتر عبور کند. چگونه می‌توان این مشکل را رفع کرد؟

۱۱-۲ در یک بستهٔ IPv4، اندازهٔ محموله در اولین فِرگمنت برحسب اُکتت مساوی $\text{Total Length} - (4 \times \text{IHL})$ است. اگر این اندازه کمتر از حداقل لازم (۸ اُکتت برای TCP) باشد، آنگاه این فِرگمنت و تمام بسته پذیرفته نمی‌شوند. روش دیگری برای برآورده نمودن این منظور پیشنهاد کنید که فقط از میدان Fragment Offset استفاده کند.

۱۱-۳ RFC 791 که مشخص‌کنندهٔ پروتکل IPv4 است، یک الگوریتم بازسازی مجدد(reassembly) را توصیف می‌کند که منجر به تولید یک فِرگمنت جدید می‌گردد که جای قسمت‌های هم‌پوشان فِرگمنت‌های قبل را پر می‌کند. با داشتن چنین پیاده‌سازی، یک حمله‌کننده می‌تواند یک سری بسته‌هائی را بسازد که در آنها پائین‌ترین فِرگمنت (zero-offset) شامل داده‌های بی‌آزار بوده (بنابراین بتوسط مدیریت فیلتر بسته‌ها عبور کند) و برخی بسته‌های بعد که دارای offset غیرصفربوده و روی اطلاعات سرآیند TCP (مثلاً پورت مقصد) افتاده و باعث تغییر آنها شود. بستهٔ دوم از بیشتر پیاده‌سازی‌ها عبور نموده زیرا دارای یک fragment offset صفر نیست. روشی را پیشنهاد کنید که بتواند از سوی یک فیلتر بسته‌ها بکار گرفته شده و با این حمله مقابله نماید.

۱۱-۴ ضرورت قانون « بالاتر را نخواند» برای یک سیستم امن چند سطحه کاملاً روشن است. اهمیت قانون « پائین‌تر را ننویسد» چیست؟

۱۱-۵ در شکل ۱۱-۵ یک لینک اسب تروا در زنجیرهٔ copy-and-observe-later پاره شده است. دو زاویهٔ ممکن دیگر برای حمله بتوسط Alice وجود دارد: Alice وارد سیستم شده و تلاش کند تا دنباله را مستقیماً بخواند و یا اینکه Alice یک سطح امنیتی حساس را به فایل جیب مخفی اعمال نماید. آیا پایشگر مرجع از این حملات جلوگیری می‌کند؟

پیوست (الف)

جنبه‌هائی از تئوری از اعداد

الف - ۱ اعداد اول و اول نسبی

مقسوم‌علیه‌ها
اعداد اول
اعداد اول نسبی

الف- ۲ حساب پیمانه‌ای

الف-۱ اعداد اول و اول نسبی

در این بخش تنها با اعداد صحیح غیرمنفی سروکار خواهیم داشت. استفاده از اعداد صحیح منفی تفاوت زیادی را در بحث بوجود نمی‌آورد.

مقسوم علیه‌ها

می‌گوئیم که $a, b \neq 0$ را می‌شمارد اگر برای مقداری از m، $a = mb$ باشد که درآن a ، b و m اعداد صحیح می‌باشند. یعنی b، a را می‌شمارد اگر تقسیم این دو برهم باقیمانده نداشته باشد. علامت $b \mid a$ اغلب به مفهوم این‌که b، a را می‌شمارد بکار می‌رود. همچنین $b \mid a$ به این مفهوم است که b *مقسوم‌علیه* a است. برای مثال، مقسوم‌علیه‌های عدد ۲۴، اعداد ۱، ۲، ۳، ۴، ۶، ۸، ۱۲ و ۲۴ هستند.

روابط زیر برقرارند:

- اگر $a \mid 1$ ، آنگاه $a = \pm 1$
- اگر $a \mid b$ و $b \mid a$ آنگاه $a = \pm b$
- هر $b \neq 0$ ، 0 را می‌شمارد
- اگر $b \mid g$ و $b \mid h$، آنگاه برای هر عدد صحیح اختیاری m و n، $b \mid (mg + nh)$

برای ملاحظهٔ آخرین نکته ذکرشده توجه کنید که

اگر $b \mid g$ ، آنگاه برای یک عدد صحیح g_1 ، $g = b \times g_1$ خواهد بود

اگر $b \mid h$ ، آنگاه برای یک عدد صحیح g_2 ، $h = b \times g_2$ خواهد بود

بنابراین

$$mg + nh = mbg_1 + nbh_1 = b \times (mg_1 + nh_1)$$

و در نتیجه b مقسوم علیه $mg + nh$ خواهد بود.

اعداد اول

هر عدد صحیح $p > 1$ یک عدد اول است اگر تنها مقسوم‌علیه‌های آن ± 1 و $\pm p$ باشند. اعداد اول نقش مهمی در تئوری اعداد و در تکنیک‌های مورد بحث در فصل ۳ این کتاب دارند.

هر عدد صحیح $a > 1$ را می‌توان به فرم یکتائی به صورت فاکتور زیر درآورد:

$$a = p_1{}^{a_1} p_2{}^{a_2} \dots p_t{}^{a_t}$$

که درآن $p_1 < p_2 < \dots < p_t$ اعداد اول بوده و هر a_i یک عدد صحیح مثبت است. برای مثال، ۱۳ × ۷ = ۹۱ و ۱۳ × ۱۱^{۲} × ۷ = ۱۱۰۱۱ می‌باشد.

بیان مسأله به نحو دیگر نیز مفید است. اگر P مجموعهٔ همهٔ اعداد اول باشد، آنگاه هر عدد صحیح مثبت را می‌توان به فرم یکتائی به شکل زیر نوشت:

$$a = \prod_{p \in \mathrm{P}} p^{a_p}$$ که در آن هر $a_p \geq 0$ است

طرف سمت راست عبارت بالا، حاصلضرب تمام اعداد اول p ممکن است. برای هر مقدار a، بیشتر توان‌های a_p صفر هستند.

اندازهٔ هر عدد مثبت داده شده را می‌توان با لیست نمودن تمام توان‌های غیرصفر فرمول قبل تعیین کرد. بنابراین، عدد صحیح ۱۲ بصورت $\{ a_2 = ۲ \text{ و } a_3 = ۱ \}$ و عدد صحیح ۱۸ بصورت $\{ a_2 = ۱ \text{ و } a_3 = ۲ \}$ نشان داده می‌شود. حاصلضرب دو عدد معادل جمع کردن توان‌های متناظر با هم است.

$$k = mn \quad \rightarrow \quad k_p = m_p + n_p \qquad \text{برای تمام مقادیر } p$$

ببینیم برحسب فاکتورهای اول ذکرشده $a \mid b$ چه معنی می‌دهد؟ هر عدد صحیح به فرم p^k را تنها می‌توان به یک عدد صحیح با توان کوچک‌تر یا مساوی عدد اول p^j $(j \leq k)$ تقسیم کرد. بنابراین می‌توان گفت

$$a \mid b \quad \rightarrow \quad a_p \leq b_p \qquad \text{برای تمام } p \text{ ها}$$

اعداد اول نسبی

نماد $\gcd(a, b)$ را برای نمایش **بزرگترین مقسوم‌علیه مشترک** a و b بکار می‌بریم. عدد صحیح مثبت c را بزرگترین مقسوم‌علیه مشترک a و b می‌نامیم اگر

۱- c یک مقسوم‌علیه a و b باشد.

۲- هر مقسوم‌علیه a و b یک مقسوم‌علیه c باشد.

بیان دیگر مطلب چنین است:

$$\gcd(a, b) = \text{ماکزیمم } k \text{ با شرط اینکه } k \mid a \text{ و } k \mid b$$

چون می‌خواهیم که بزرگترین مقسوم‌علیه مشترک مثبت باشد، gcd(*a*,*b*)=gcd(-*a*,*b*)=gcd(*a*,-*b*)=gcd(-*a*,-*b*). در حالت کلی، $\gcd(a, b) = \gcd(|a|, |b|)$. برای مثال gcd (60,24) = gcd (60,-24) = 12. همچنین چون تمام اعداد صحیح 0 را می‌شمارند، $\gcd(a, 0) = |a|$.

تعیین بزرگترین مقسوم‌علیه مشترک دو عدد صحیح مثبت ساده است در صورتی که بتوان آن دو عدد را بصورت مضربی از اعداد اول نوشت. برای مثال

$$300 = 2^2 \times 3^1 \times 5^2$$

$$18 = 2^1 \times 3^2$$

$$\gcd(18,300) = 2^1 \times 3^1 \times 5^0 = 6$$

در حالت کلی

$$k = \gcd(a, b) \quad \rightarrow \quad k_p = \min(a_p, b_p) \qquad \text{برای تمام مقادیر } p$$

تعیین فاکتورهای اول یک عدد بزرگ کار آسانی نیست و بنابراین رابطهٔ قبل مستقیماً به محاسبهٔ بزرگترین مقسوم‌علیه مشترک منجر نخواهد شد.

اعداد صحیح a و b نسبت به هم اول‌اند اگر هیچ فاکتور اول مشترکی نداشته باشند، یعنی اگر تنها فاکتور مشترک آنها عدد ۱ باشد. این بیان معادل این است که بگوئیم a و b نسبت به هم اول‌اند اگر $\gcd(a, b) = 1$ باشد. برای مثال، ۸ و ۱۵ نسبت به هم اول‌اند زیرا مقسوم‌علیه‌های ۸ مساوی ۱ و ۲ و ۴ و ۸ بوده و مقسوم‌علیه‌های ۱۵، اعداد ۱ و ۳ و ۵ و ۱۵ هستند و بنابراین ۱ تنها عدد مشترک این دو لیست است.

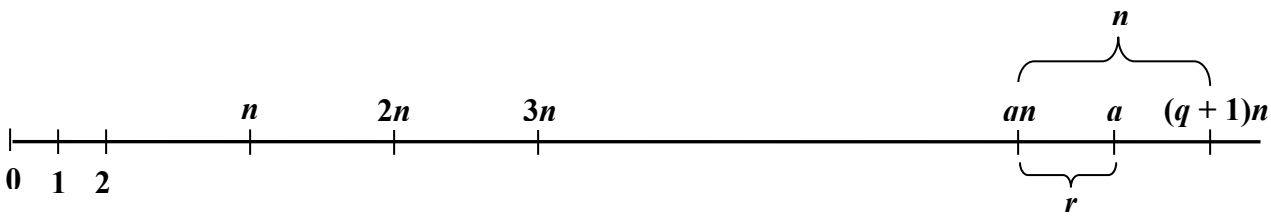
الف -۲ حساب پیمانه‌ای

برای هر عدد صحیح مثبت n و هر عدد صحیح غیرمنفی a، اگر a را بر n تقسیم کنیم، یک خارج قسمت صحیح q و یک باقیماندهٔ صحیح r بدست می‌آوریم که از رابطهٔ زیر تبعیت می‌کنند:

$$a = qn + r \qquad 0 \le r < n \; ; \; q = \lfloor a / n \rfloor$$

که در آن $\lfloor x \rfloor$ بزرگترین عدد صحیح کوچکتر یا مساوی x است.

شکل الف-۱ نشان می‌دهد که با داشتن a و n مثبت، همیشه می‌توان q و r را طوری پیدا کرد که شرط قبل را ارضاء کند. با نمایش اعداد صحیح روی یک محور اعداد، a در جائی از خط قرار خواهد گرفت (مقدار مثبت a نشان داده شده است، می‌توان عدد منفی a را نیز روی محور نشان داد). با شروع از 0 بسمت n، $2n$، تا qn طوری پیش می‌رویم که $qn \le a$ و $(q + 1)n > 1$ باشد. فاصلهٔ بین qn تا a برابر r بوده و ما اندازه‌های یکتای مقادیر q و r را پیدا کرده‌ایم. باقیماندهٔ r را اغلب **residue** گویند.



شکل الف-۱ رابطهٔ $a = qn + r \;;\; 0 \le r < n$

اگر a یک عدد صحیح و n یک عدد صحیح مثبت باشد، باقیماندهٔ تقسیم a بر n را بصورت $a \bmod n$ تعریف می‌کنیم. بنابراین، برای هر عدد صحیح n، همیشه می‌توان نوشت:

$$a = \lfloor a / n \rfloor \times n + (a \bmod n)$$

دو عدد صحیح a و b را **هم‌نهشت به پیمانهٔ *n*** گویند هرگاه $(a \bmod n) = (b \bmod n)$ باشد. این خاصیت را بصورت $a \equiv b \bmod n$ می‌نویسند. بعنوان مثال $73 \equiv 4 \bmod 23$ و $21 \equiv -9 \bmod 10$. توجه کنید که اگر $a \equiv 0 \bmod n$ باشد، آنگاه $n \mid a$.

عملگر پیمانه دارای خواص زیر است:

۱- اگر $n \mid (a-b)$ آنگاه $a \equiv b \bmod n$ است.

۲- $(a \bmod n) = (b \bmod n)$ به مفهوم $a \equiv b \bmod n$ است.

۳- اگر $a \equiv b \bmod n$ باشد، $b \equiv a \bmod n$ است.

۴- اگر $a \equiv b \bmod n$ و $b \equiv c \bmod n$ باشد، $a \equiv c \bmod n$ است.

برای اثبات خاصیت اول، اگر $n \mid (a-b)$، آنگاه برای مقداری از k، $kn = (a - b)$. بنابراین می‌توان نوشت $a = b + kn$. در نتیجه $(a \bmod n)$ = (باقیمانده تقسیم $b + kn$ بر n) = (باقیماندهٔ تقسیم b بر n) = $(b \bmod n)$

بقیه خواص نیز به آسانی اثبات می‌شوند.

عملگر $(\bmod n)$ تمام اعداد صحیح را به مجموعهٔ اعداد صحیح $\{ 0 , 1 , 2 , \ldots , (n-1) \}$ نگاشت می‌کند. در اینجا این سؤال مطرح می‌شود: آیا می‌توان عملیات حساب در این مجموعهٔ محدود را انجام داد؟ جواب سؤال مثبت بوده و تکنیک شناخته شده برای انجام این عمل **حساب پیمانه‌ای** (modular) است.

حساب پیمانه‌ای دارای خواص زیر است:

۱- $[(a \bmod n) + (b \bmod n)] \bmod n = (a + b) \bmod n$

۲- $[(a \bmod n) - (b \bmod n)] \bmod n = (a - b) \bmod n$

۳- $[(a \bmod n) \times (b \bmod n)] \bmod n = (a \times b) \bmod n$

خاصیت اول را ثابت می‌کنیم. اگر $(a \bmod n) = r_a$ و $(b \bmod n) = r_b$ تعریف شود، آنگاه برای یک عدد صحیح j، $a = ra + jn$ بوده و برای یک عدد صحیح k، $b = rb + kn$ خواهد بود. آنگاه

$$\begin{aligned}(a + b) \bmod n &= (ra + jn + rb + kn) \bmod n \\ &= (ra + rb + (k + j)n) \bmod n \\ &= (ra + rb) \bmod n \\ &= [(a \bmod n) + (b \bmod n)] \bmod n\end{aligned}$$

بقیهٔ خواص نیز به آسانی اثبات می‌شوند.

پیوست (ب)

واژه‌های امنیت شبکه

گرچه این واژه‌ها در طول فصول کتاب بطور مفصل تعریف شده و مورد بحث قرار گرفته‌اند ولی برای کسی که تنها بدنبال آشنائی با تعریف مختصر آنهاست، این واژه‌نامه می‌تواند مفید واقع شود.

asymmetric encription (رمزنگاری غیرمتقارن): نوعی سیستم رمزنگاری که در آن رمزنگاری و رمزگشائی با استفاده از دو کلید مختلف انجام می‌شود. نام یکی از کلیدها، کلیدعمومی و نام دیگری کلیدخصوصی است. این نوع رمزنگاری را رمزنگاری کلید- عمومی نیز می‌خوانند.

authentication (اعتبارسنجی): عمل تأئید هویت یک موجودیت سیستم است. در این مقوله اعتبارسنجی را می‌توان نوعی احراز هویت دانست.

authenticator (اعتبارسنج): نوعی اطلاعات اضافی است که به پیام متصل می‌شود تا گیرنده را قادر سازد تا معتبربودن پیام را آزمایش کند. اعتبارسنج ممکن است مستقل از محتوای پیام بوده (مثل یک nonce یا یک شناسهٔ منبع) و یا ممکن است تابعی از محتوای پیام باشد(مثل یک اندازهٔ hash یا یک parity).

avalanche effect (اثر بهمنی): یک خاصیت مثبت در یک الگوریتم رمزنگاری که بواسطهٔ آن یک تغییر کوچک در متن ساده و یا کلید، موجب تغییر بزرگی در متن رمزشده می‌گردد. برای یک کُد درهم‌ساز، اثر بهمنی باعث می‌گردد تا تغییر کوچکی در پیام موجب تغییر بزرگی در چکیدهٔ پیام شود.

bacteria (باکتری): برنامه‌ای که برای تولید کپی‌هائی از خود، منابع سیستم را به کار بیهوده مشغول می‌سازد.

birthday attack (حملهٔ روز تولد): این حمله که نوعی تلاش برای شکستن رمز است، سعی می‌کند تا دو مقدار مختلف در دامنهٔ یک تابع را که به یک مقدار یکسان در بُرد تابع نگاشت می‌شوند، پیدا کند.

block chaining (زنجیره‌کردن قالب‌ها): روشی در خلال رمزنگاری قالبی متقارن است که یک بلوک خروجی را نه تنها وابسته به متن سادهٔ جاری و کلید کرده، بلکه آن را وابسته به ورودی و/ یا خروجی مرحلهٔ قبل نیز می‌نماید. اثر زنجیره‌کردن قالب‌ها این است که دو بلوک متن سادهٔ مشابه، دو بلوک متن رمزشدهٔ متفاوت تولید کرده و در نتیجه شکستن رمز سخت‌تر خواهد شد.

block cipher (رمز قالبی): یک الگوریتم رمزنگاری متقارن است که در آن یک بلوک از بیت‌های متن ساده (معمولاً ۶۴ یا ۱۲۸ بیت)، کلاً به صورت یک بلوک از متن رمزشده با همان طول تبدیل می‌شود.

byte (بایت): یک ردیف ۸- تائی از بیت‌ها را گویند. به نام *اُکتت* نیز خوانده می‌شود.

cipher (رمز): یک الگوریتم برای رمزنگاری و رمزگشائی است. یک رمز، یک بخش از اطلاعات (یک عنصر از متن ساده) را با هدف پنهان کردن محتوای آن با عنصر دیگری عوض می‌کند. نوعاً نحوهٔ این تعویض از طریق یک کلید سرّی هدایت می‌شود.

ciphertext (متن رمزشده): خروجی یک الگوریتم رمزنگاری است که فرم رمزشدهٔ داده‌ها یا پیام می‌باشد.

code (کُد): یک قاعدهٔ تغییرناپذیر برای جایگزین کردن یک بخش از اطلاعات (مثل حرف، کلمه، جمله) با عنصر دیگری است که لزوماً از جنس اطلاعات اولیه نیست. معمولاً هدف این عمل پنهان کردن اطلاعات نیست. مثال‌هائی در این زمینه شامل کُد حرفی ASCII (هر علامت بتوسط یک ردیف ۷- بیتی نشان داده می‌شود) و کُد فرکانسی FSK (هر مقدار باینری با یک فرکانس خاص نشان داده می‌شود)، می‌باشند.

computaionally secure (ازنظر محاسباتی امن): از اینجهت امن است که زمان و/ یا هزینهٔ شکستن امنیت بقدری بالاست که انجام آن معقول نمی‌باشد.

confusion (گیج کردن): یک تکنیک رمزنگاری است که در جستجوی هرچه پیچیده‌تر کردن رابطهٔ بین خواص آماری متن رمزشده با کلید رمزنگاری است.این امر با استفاده از الگوریتم‌های مخلوط‌کنندهٔ پیچیده که وابسته به کلید و متن ورودی‌اند صورت می‌پذیرد.

conventional encryption (رمزنگاری رسمی): همان رمزنگاری متقارن است.

covert channel (کانال پنهان): یک کانال ارتباطی است که انتقال اطلاعات را بنحوی ممکن می‌سازد که خارج از اهداف طراحی آن تسهیلات بوده است.

cryptanalysis (شکستن رمز): شاخه‌ای از علم رمزنگاری است که مرتبط با کشف رمز برای استخراج اطلاعات، و یا جعل اطلاعات رمزشده بنحوی است که معتبر تلقی گردد.

cryptographic checksum (جمع کنترلی رمزی): یک اعتبارسنج است که تابعی هم از دیتائی که باید اعتبار آن سنجیده شود و هم از یک کلید سرّی می‌باشد. به آن کُد اعتبارسنجی پیام (MAC) نیز گفته می‌شود.

cryptography (رمزنگاری): شاخه‌ای از علم رمزشناسی است که مرتبط با طراحی الگوریتم‌های رمزنگاری و رمزگشائی بوده که هدف آنها سرّی نگاه داشتن و/ یا معتبرنگاه داشتن پیام است.

cryptology (رمزشناسی): بررسی ارتباطات امن است که هم شاخهٔ رمزنگاری و هم شاخهٔ کشف رمز را شامل می‌گردد.

decryption (رمزگشائی): تبدیل متن و یا دیتای رمزشده (که متن رمزشده نامیده می‌شود) به متن و یا دیتای اولیه (که متن ساده نامیده می‌شود) است.

differential cryptanalysis (شکستن تفاضلی رمز): روشی که در آن متون سادهٔ انتخاب شده با اُلگوهای تفاضلی XOR شده بخصوص رمزنگاری می‌شوند. اُلگوهای تفاضلی متن رمزشدهٔ منتجه، اطلاعاتی را به دست می‌دهند که می‌تواند برای کشف کلید رمز مفید باشد.

diffusion (پخش کردن): یک تکنیک رمزنگاری است که در پی ایجاد ابهام در ساختار آماری پیام، با پخش کردن اثر هر عنصر متن ساده به بسیاری از عناصر متن رمزشده، است.

digital signature (امضای دیجیتال): یک تکنیک رمزنگاری است که خلق‌کنندهٔ پیام را قادر می‌سازد تا کُدی که خاصیت یک امضاء را دارد به پیام متصل سازد. امضاء با محاسبهٔ hash پیام و رمزنگاری پیام با کلید خصوصی خلق‌کنندهٔ پیام انجام می‌شود. امضاء، منبع صدور و صحت پیام را تضمین می‌کند.

digram (دو- حرفی): یک ردیف دو- حرفی است. در زبان انگلیسی و سایر زبان‌ها، وقوع نسبی دو- حرفی‌ها در متن ساده می‌تواند در شکستن بعضی رمزها مورد استفاده قرار گیرد.

discretionary access control (کنترل دست‌یابی منصفانه): یک سرویس کنترل دست‌یابی که یک سیاست امنیتی، مبتنی بر هویت موجودیت‌های سیستم و حقوق دست‌یابی آنها به منابع سیستم، را پیاده می‌نماید. این سرویس از این‌جهت «منصفانه» نام دارد که یک موجودیت ممکن است دارای آنچنان حق دست‌یابی باشد که بتواند با صلاحدید خود به موجودیت دیگری حق دست‌یابی به بعضی منابع را تفویض نماید.

divisor (مقسوم علیه): یک عدد صحیح را مقسوم علیه عدد صحیح دیگر گویند در صورتی که باقیماندهٔ تقسیم این بر آن صفر باشد.

encryption (رمزنگاری): تبدیل یک متن ساده و یا دیتا به یک فرم نامفهوم را گویند که با استفاده از یک نگاشت برگشت‌پذیر مبتنی بر یک جدول تبدیل یا الگوریتم صورت می‌پذیرد.

firewall (دیوار آتش): یک کامپیوترکه بعنوان واسط ارتباط با کامپیوترهای خارج از یک شبکه تخصیص یافته و عوامل حفاظتی بخصوصی در آن تعبیه شده است تا فایل‌های حساس و یا کامپیوترهای درون شبکه را محافظت نماید. این دستگاه به شبکهٔ خارجی، بخصوص اینترنت، اتصالات و خطوط تلفنی ورودی سرویس می‌دهد.

greatest common divisor (بزرگترین مقسوم علیه مشترک): بزرگترین مقسوم علیه مشترک دو عدد صحیح a و b، بزرگترین عدد صحیح مثبتی است که هم a و هم b را می‌شمارد. گویند یک عدد صحیح عدد صحیح دیگر را می‌شمارد در صورتی که تقسیم آن دو به هم باقیمانده نداشته باشد.

hash function (تابع درهم‌ساز): تابعی را گویندکه یک بلوک دیتا یا پیام با طول متغیر را به یک اندازهٔ ثابت، که کُد hash خوانده می‌شود، نگاشت می‌کند. این تابع طوری طراحی می‌شود که که وقتی مورد محافظت واقع گردد یک اعتبارسنج برای دیتا و یا پیام باشد. به آن چکیدهٔ پیام هم می‌گویند.

honeypot (طعمه) : یک سیستم دام است که برای فریب دادن حمله کننده‌ها و دور نگاه‌داشتن آنها از سیستم‌های اصلی بکار می‌رود. نوعی تشخیص تهاجم است.

initialization vector (بردار آغازگر): یک بلوک تصادفی از داده‌ها است که برای آغاز رمزنگاری بلوک‌های متعدد دیتا در هنگام استفاده از تکنیک رمزنگاری زنجیره‌ای قالبی از آن استفاده می‌شود. از IV برای خنثی نمودن حملات متن سادهٔ معلوم استفاده می‌شود.

intruder (مهاجم): فردی که بصورت غیرمجاز به یک سیستم کامپیوتری دست یافته و یا برای این منظور تلاش می‌کند. او می‌تواند یک کاربر معتبر بوده که برای کسب امتیازاتی بیشتر از حقوق خود تلاش می‌نماید.

intrusion detection system (سیستم تشخیص تهاجم): مجموعه‌ای از ابزارهای خودکار که برای تشخیص دست‌یابی‌های غیرمجاز به یک سیستم میزبان بکار گرفته می‌شود.

Kerberos: نامی است که به برنامهٔ سرویس اعتبارسنجی پروژهٔ Athena داده شده است.

key distribution center (مرکز توزیع کلید): یک سیستم معتبر برای ارسال کلیدهای موقت اجلاس به رؤسای ارتباط است. هر کلید اجلاس با استفاده از یک کلید اصلی، که بین مرکز توزیع کلید و رئیس موردنظر به اشتراک گذاشته شده است، رمزنگاری می‌شود.

logic bomb (بمب لاجیک): منطقی که در یک برنامهٔ کامپیوتری طوری جاسازی شده است که وقوع شرایط خاصی در سیستم را کنترل کند. وقتی این شرایط حاصل شوند، عملی اجرا خواهد شد که منجر به فعالیت‌های غیرمجاز می‌گردد.

mandatory access control (کنترل دست‌یابی اجباری): وسیله‌ای برای محدودکردن دست‌یابی به موضوعات است که مبتنی بر تخصیص صفاتی به یک کاربر، یک فایل و سایر موضوعات است. این کنترل‌ها به این مفهوم اجباری هستند که نمی‌توانند بتوسط کاربر و یا برنامه‌های او دستکاری شوند.

man-in-the-middle attack (حملهٔ واسطه‌گرانه): نوعی حملهٔ استراق سمع فعال است که در آن حمله کننده در جریان مخابرهٔ داده‌ها وارد شده و بطور انتخابی دیتای انتقال داده شده را طوری دستکاری نماید که خود را بجای یک یا چند طرف درگیر ارتباط جا بزند.

master key (کلید اصلی): یک کلید پردوام که بین یک مرکز توزیع کلید و یک رئیس ارتباط به اشتراک گذاشته می‌شود تا از آن برای رمزکردن انتقال کلیدهای اجلاس استفاده شود. . معمولاً کلیدهای اصلی به فرمی خارج از قواعد رمزنگاری توزیع می‌شوند. با آن کلید رمزنگاری- کلید هم می‌گویند.

meet-in-the-middle attack : این یک حملهٔ شکستن رمز است که تلاش می‌کند تا اندازه‌ای در برد و دامنه ترکیب دو پیام را بنحوی پیدا نماید که نگاشت مستقیم تابع اول برابر تصویر معکوس تابع دوم باشد. بعبارت دیگر ملاقات دو مقدار در وسط دو تابع ترکیبی انجام شود.

message authentication (اعتبارسنجی پیام): عملی که برای تأئید صحت پیام از آن استفاده می‌شود.

Message authentication code (MAC): سرجمع مرتبط با رمزنگاری

message digest (چکیدهٔ پیام): تابع درهم ساز (hash)

modular arithmetic (حساب پیمانه‌ای): نوعی از حساب اعداد صحیح که برای عددی مانند n ، همهٔ اعداد صحیح را به یک مجموعهٔ متناهی $[0, 1,\ldots,n\text{-}1]$ کاهش می‌دهد. هر عدد صحیح خارج از این محدوده با انتخاب باقیماندهٔ تقسیم آن بر n به یک عدد داخل این محدوده تبدیل می‌شود.

mode of operation (مُود عملیاتی): تکنیکی برای ارتقاء اثر یک الگوریتم رمزنگاری و یا وفق دادن یک الگوریتم به کاربردی بخصوص، همچون اعمال یک رمز قالبی به ردیفی از بلوک‌های دیتا و یا یک جریان داده‌هاست.

multilevel security (امنیت چند لایه): قابلیتی است که کنترل دست‌یابی را به سطوح چندگانهٔ طبقه‌بندی داده‌ها اعمال می‌کند.

multiple encryption (رمزنگاری چندگانه): استفادهٔ مکرر از یک تابع رمزنگاری، با کلیدهای مختلف، برای تولید یک نگاشت پیچیده‌تر از متن ساده به متن رمزشده است.

nibble: ردیفی از چهار بیت را گویند

nonce: یک شناسه و یا یک عدد که تنها یک‌بار بکار می‌رود.

one-way function (تابع یک- طرفه): تابعی که محاسبهٔ آن آسان بوده ولی محاسبهٔ معکوس آن غیرممکن است.

password (کلمهٔ عبور): یک اندازهٔ سرّی، معمولاً ردیفی از کاراکترها، که از آن بعنوان یک اطلاعات اعتبارسنجی استفاده می‌شود. یک کلمهٔ عبور معمولاً با یک شناسهٔ کاربر جفت بوده که این شناسهٔ در عمل اعتبارسنجی بصورت واضح ارائه می‌شود. در بعضی موارد این شناسه ممکن است ضمنی باشد.

plaintext (متن ساده): ورودی یک تابع رمزنگاری و یا خروجی یک تابع رمزگشائی است.

primitive root (ریشهٔ اولیه): اگر r و n نسبت به هم اول باشند، $n > 0$، و اگر $\Phi(n)$ کوچک‌ترین توان مثبت m بنحوی باشد که $r^m \equiv 1 \bmod n$ باشد، آنگاه r را ریشهٔ اولیهٔ با پیمانهٔ n خوانند.

private key (کلید خصوصی): یکی از دو کلیدی که در سیستم رمزنگاری نامتقارن از آن استفاده می‌شود. بمنظور ارتباطات امن، کلید خصوصی بایستی تنها برای خلق‌کنندهٔ آن معلوم باشد.

pseudorandom number generator (تولیدکنندهٔ اعداد شبه تصادفی): تابعی که بصورت یقینی ردیفی از اعداد را درست می‌کند که ظاهراً از نظر آماری تصادفی هستند.

public key (کلید عمومی): یکی از دو کلیدی است که در سیستم رمزنگاری نامتقارن از آن استفاده می‌شود. کلید عمومی، در معرض استفادهٔ عموم قرار می‌گیرد تا بهمراه کلید خصوصی جفت آن مورد استفاده قرار گیرد.

public-key certificate (گواهی‌نامهٔ کلید- عمومی): شامل یک کلید عمومی باضافه User ID صاحب آن است، که کل آن بتوسط یک شخص ثالث معتبر امضاء شده است. شخص ثالث معمولاً یک مقام مسئول صدورگواهی‌نامه (CA) است که همچون یک سازمان دولتی و یا یک مؤسسهٔ اعتباری مورد اعتماد یک جمعیت از کاربران است.

public-key encryption (رمزنگاری کلید- عمومی): رمزنگاری نامتقارن.

public-key infrastructure(PKI) (زیرساخت کلید- عمومی): مجموعهٔ سخت‌افزارها، نرم‌افزارها، مردم، سیاست‌ها و رَویه‌های لازم برای خلق، مدیریت، ذخیره سازی، توزیع و ابطال گواهی‌نامه‌های دیجیتال مبتنی بر رمزنگاری نامتقارن می‌باشد.

relatively prime (اول نسبی): دو عدد در صورتی نسبت به هم اول هستند که هیچ فاکتور اول مشترکی با هم نداشته باشند، یعنی مقسوم علیه مشترک آنها یک باشد.

replay attack (حملات بازخوانی): حمله‌ای که در آن یک سرویس که قبلاً اعتبارسنجی و کامل شده است با جعل تقاضای مجددی برای کسب فرامین معتبر تلاش می‌کند.

residue: وقتی عدد صحیح a بر عدد صحیح n تقسیم می‌شود، باقیماندهٔ r را residue گویند. بطور معادل $r = a \bmod n$

residue class: تمام اعداد صحیحی که وقتی بر n تقسیم شوند دارای باقیماندهٔ یکسانی باشند یک residue class به پیمانه n را تشکیل می‌دهند. بنابراین، برای یک باقیماندهٔ داده شده r، اعداد صحیح r، $r \pm n$، $r \pm 2n$ و ... متعلق به residue class (mod n) هستند.

RSA algorithm: یک الگوریتم رمزنگاری کلید- عمومی است که بر مبنای بتوان رساندن یک عدد در حساب پیمانه‌ای قرار دارد. این الگوریتم تنها الگوریتم پذیرفته شده عمومی برای یک رمزنگاری کلید- عمومی امن و عملی است.

secret key (کلید سرّی): کلیدی است که از آن در یک سیستم رمزنگاری متقارن استفاده می‌شود.

security attack (حملهٔ امنیتی): یک ضربه بر امنیت سیستم که از یک تهدید هوشمندانه ناشی می‌شود. یعنی یک عمل هوشمندانه برای یک تلاش هوشمندانه در جهت شکست سرویس‌های امنیتی و نقض سیاست امنیتی یک سیستم است.

security mechanism (مکانیسم امنیتی): یک پردازش (یا دستگاهی که این پردازش را فراهم می‌کند) که برای تشخیص، جلوگیری و یا بازیابی یک حملهٔ امنیتی طراحی شده است.

security service (سرویس امنیتی): یک پردازش و یا یک سرویس ارتباطی است که امنیت سیستم‌های پردازش دیتا و انتقال اطلاعات یک سازمان را ارتقاء می‌دهد. سرویس‌ها به منظور مقابله با حملات امنیتی طراحی شده و از یک یا چند مکانیسم امنیتی برای ایجاد سرویس استفاده می‌کنند.

security threat (تهدید امنیتی): خطر بالقوه‌ای برای نقض امنیت است که وقتی وجود دارد که شرایط، قابلیت، عمل و یا رویدادی بتواند امنیت را نقض کرده و ایجاد اخلال نماید. بعبارت دیگر یک تهدید یک خطر بالقوه است که ممکن است از یک نقطهٔ ضعف استفاده کند.

session key (کلید اجلاس): یک کلید رمزنگاری موقت است که بین دو رئیس ارتباط از آن برای رمزنگاری استفاده می‌شود.

steganography (پنهان نگاری): روش‌های پنهان کردن وجود یک پیام و یا داده‌های دیگر است. این مقوله با رمزنگاری که در آن مفهوم پیام، و نه وجود پیام، پنهان می‌شود متفاوت است.

stream cipher (رمز دنباله‌ای): یک الگوریتم رمزنگاری متقارن است که در آن متن رمز شدهٔ خروجی، بیت- به- بیت و یا بایت- به- بایت، از روی دنبالهٔ متن سادهٔ ورودی تولید می‌شود.

symmetric encryption (رمزنگاری متقارن): یک روش رمزنگاری که در آن رمزنگاری و رمزگشائی با استفاده از یک کلید صورت می‌پذیرد. به آن رمزنگاری رسمی و یا سنّتی نیز گویند.

trapdoor (درب مخفی): نقطهٔ ورودی مخفی و غیرمستندی که دست‌یابی به برنامه یا سیستم، بدون عبور از مراحل معمول کنترلی، را امکان‌پذیر می‌نماید.

trapdoor one-way function (تابع یک- طرفه با درب مخفی): تابعی که محاسبهٔ آن ساده بوده و محاسبهٔ معکوس آن مقدور نیست مگر اینکه اطلاعات ویژه‌ای در دست باشد.

Trojan horse (اسب تروا): یک برنامهٔ کامپیوتری که در ظاهر مفید و قابل استفاده بوده ولی شامل یک تابع بالقوه بداندیش نیز هست که مکانیسم‌های امنیتی را شکست می‌دهد. این کار، گاهی با استثمار اعتبارهای قانونی موجودیتی که برنامه را بکار گرفته است، صورت می‌پذیرد.

trusted system (سیستم مورد اعتماد): یک کامپیوتر و سیستم عامل که برای اجرای یک سیاست امنیتی مورد تأئید است.

unconditionally secure (بطور غیرمشروط امن): در برابر دشمنی که زمان نامحدود و منابع محاسباتی نامحدود دارد، امن است

virtual private network(VPN) (شبکهٔ خصوصی مجازی): شامل مجموعه‌ای از کامپیوترهاست که بتوسط یک شبکهٔ نسبتاً ناامن بهم متصل شده و از رمزنگاری و پروتکل‌های مخصوص برای ایجاد امنیت استفاده می‌کند.

virus (ویروس): کُد برنامه‌ای که در درون یک برنامهٔ دیگر جاسازی شده و باعث می‌شود تا یک کپی از ویروس در برنامه یا برنامه‌های دیگر وارد شود. علاوه بر انتشار، ویروس معمولاً عملیات ناخواسته‌ای را انجام می‌دهد.

worm (کِرم): برنامه‌ای است که می‌تواند خود را تکثیر کرده و کپی‌های خود را در عرض شبکه از یک کامپیوتر به کامپیوتر دیگر ارسال دارد. پس از ورود، یک کِرم ممکن است فعال شده و مجدداً تکثیر و انتشار یابد. علاوه بر انتشار، کِرم معمولاً کارهای ناخواسته‌ای را نیز انجام می‌دهد.

zombie (زامبی): برنامه‌ای است که بطور مخفیانه کنترل کامپیوتر دیگری،که به اینترنت متصل است، را بدست گرفته و از آن کامپیوتر مبادرت به حملاتی می‌کند که دنبال کردن آن برای خلق‌کنندهٔ زامبی نیز مشکل است.

پیوست (ج)

مراجع

علائم اختصاری

ACM Association for Computing Machinery
IEEE Institute of Electrical and Electronics Engineers
NIST National Institute of Standards and Technology

ALVA90 Alvare,A. "How Crackers Crack Passwords or What Passwords to Avoid." *Proceedings, UNIX Security Workshop II*, August 1990.

ANDE80 Anderson, J. *Computer Security Threat Monitoring and Surveillance*. Fort Washington, PA: James P. Anderson Co., April 1980.

AUDI04 Audin, G. "Next-Gen Firewalls: What to expect." *Business Communications Review*, June 2004.

AXEL00 Axelsson, S. "The Base-Rate Fallacy and the Difficulty of Intrusion Detection." *ACM Transactions and Information and system Security*, August 2000.

BACE00 Bace, R. *Intrusion Detection*. Indianapolis, IN: Macmillan Technical Publishing, 2000.

BACE01 Bace, R., and Mell, P. *Intrusion Detection Systems*. NIST Special Publication SP 800-31 , November 2000.

BARR03 Barreto, P., and Rijmen, V. "The Whirlpool Hashing Function." *Submitted to NESSIE*, September 2000, revised May 2003.

BAUE88 Bauer, D., and Koblentz, M."NIDX- An Expert System for Real-Time Network Intrusion Detection." *Proceedings, Computer Networking Symposium*, April 1988.

BELL90 Bellovin, S. and Merritt, M. " Limitations of the Kerberos Authentication System." *Computer Communications Review*, October 1990.

BELL92 Bellovin, S., "There Be Dragons." *Proceedings, UNIX Security Symposium III*, September 1992.

BELL93 Bellovin, S. "Packets Found on an Internet." *Computer Communications Review*, July 1993.

BELL94 Bellovin, S., and Cheswick, W. "Network Firewalls." *IEEE Communications Magazine*, September 1994.

BELL96a Bellare, M.; Canetti, R.; and Krawczyk, H. "Keying Hash Functions for Message Authentication." *Proceedings*, *CRYPTO '96*, August 1996; published by Springer-Verlag. An expanded version is available at http://www-cse.ucsd.edu/users/mihir.

BELL96b Bellare, M.; Canetti, R.; and Krawczyk, H. "The HMAC Construction." *CryptoBytes*, Spring 1996.

BERS92 Berson, T."Differential Cryptoanalysis Mod 2^{32} with Applications to MD5." *Proceedings*, *EUROCRYPT '92*, May 1992; published by Springer-Verlag.

BISH03 Bishop, M. *Computer Security: Art and Science*. Boston: Addison-Wesley,2003.

BISH05 Bishop, M. *Introduction to Computer Security*. Boston: Addison-Wesley,2005.

BLOO70 Bloom, B. "Space/time Trade-offs in Hash Coding with Allowable Errors." *Communications of the ACM*, July 1970.

BLUM97a Bluementhal, U.; Hien, N.; and Wijnen, B. "Key Derivation for Network Management Applications." *IEEE Network*, May/June, 1997.

BLUM97b Blumenthal, U, and Wijnen, B. "Security Features for SNMPv3." *The Simple Times*, December 1997.

BOER93 Boer, B., and Bosselaers, A. "Collisions for the Compression Function of MD5." *Proceedings, EUROCRYPT '93*, 1993; published by Springer-Verlag.

BRYA88 Bryant, W. *Designing an Authentication System: A Dialog in Four Scenes*. Project Athena document, February 1988. Available at http://web.mit.edu/kerberos/www/dialogue.html.

CASS01 Cass, S."Anatomy of Malice." *IEEE Spectrum*, November 2001.

CERT01 CERT Coordination Center. "Denial of Service Attacks." June 2001. http://www.cert.org/tech_tips/denial_of_service.html

CERT02 CERT Coordination Center. "Multiple vulnerabilities in Many Implementations of the Simple Network Management Protocol." CERT Advisory CA-2002-03, 25 June 2002. www.cert.org/advisories/CA-2002-03.html

CHAN02 Chang, R. "Defending Against Flooding-Based Distributed Denial - of – Service Attacks: A Tutorial." *IEEE Communications Magazine*, October 2002.

CHAP00 Chapman, D., and Zwicky, E. *Building Internet Firewalls*. Sebastopol, CA: O'Reilly, 2000.

CHEN98 Cheng, P., et al. "A Security Architecture for the Internet Protocol." *IBM Systems Journal*, Number 1, 1998.

CHES97 Chess, D. "The Future of Viruses on the Internet." *Proceedings, Virus Bulletin International Conference*, October 1997.

CHES03 Cheswick, W., and Bellovin, S. *Firewalls and Internet Security:* Repelling the wily Hacker. Reading, MA: Addison-Wesley, 2003.

COHE94 Cohen, F. *A Short Course on Computer Viruses*. New York: Wiley, 1994.

CORM01 Cormen, T.; Leiserson, C.; Rivest. R.; and Stein. C. *Introduction to Algorithms*. Cambridge, MA: Addison-Wesley, 2003.

DAVI89 Davies, D., and Price, W. *Security for Computer Networks*. New York: Wiley, 1994.

DAMG89 Damgard, I. "A Design Principle for Hash Functions." *Proceedings, CRYPTO '89*, 1989; published by Springer-Verlag.

DAVI93 Davies, C., and Ganesan, R. "BApassed: A New Proactive Password Checker." *Proceedings, 16th National Computer Security Conference*, September 1993.

DAWS96 Dawson, E., and Nielsen, L. "Automated Cryptoanalysis of XOR Plaintext Strings." *Cryptologia,* April 1996.

DENN87 Denning, D. "An Intrusion-Detection Model." *IEEE Transaction on Software Engineering*. February 1987.

DIFF76 Diffie, W., and Hellman, M. "Multiuser Cryptographic Techniques." *IEEE Transactions on Information Theory*, November 1976.

DIFF88 Diffie, W., "The First Ten Years of Public-Key Cryptography." *Proceedings of the IEEE*, May 1988. Reprinted in [SIMM92]

DOBB96 Dobbertin, H. "The Status of MD5 After a Recent Attack." *CryptoBytes*, Summer 1996.

DORA03 Doraswamy, N., and Harkins, D. *IPSec*, Upper Saddle River, NJ: Prentice Hall, 2003.

DREW99 Drew, G. *Using SET for Secure Electronic Commerce*. Upper Saddle River, NJ: Prentice Hall, 1999.

EFF98 Electronic Frontier Foundation. *Cracking DES: Secrets of Encryption Research, Wire-tap Politics, and Chip Design*. Sebastopol, CA: O'Reilly, 1998.

ENGE80 Enger, N., and Howerton, P. *Computer Security*. New York: Amacom, 1980.

FEIS73 Feistel, H. "Cryptography and Computer Privacy." *Scientific American*, May 1973.

FELT03 Felten, E. "Understanding Trusted Computing: Will Its Benefits Outweigh its Drawbacks?" *IEEE Security and Privacy*, May/June 2003.

FLUH00 Fluhrer, S., and McGrew, D. "Statistical Analysis of the Alleged RC4 Key Stream Generator." Proceedings, *Fast Software Encryption 2000*, 2000.

FLUH01 Fluhrer, S.; Mantin, I.; and Shamir, A. "Weakness in the Key Scheduling Algorithm of RC4." *Proceedings, Workshop in Selected Areas of Cryptography*, 2001.

FORD95 Ford, W. "Advances in Public-Key Certificate Standards." *ACM SIGSAC Review*, July 1995.

FORR97 Forrest, S.; Hofmeyr, S.; and Somayaji, A. "Computer Immunology." *Communications of the ACM*, October 1997.

FRAN01 Frankel, S. *Demystifying the IPSec Puzzle*. Boston: Artech House, 2001.

GARD77 Gardner, M. "A New Kind of Cipher That Would Take Millions of Years to Break." *Scientific American*, August 1977.

GARF97 Garfinkel, S., and Spafford, G. *Web Security & Commerce*. Cambridge, MA: O'Reilly and Associates, 1997.

GASS88 Gasser, M. *Building a Secure Computer System*. New York: Van Nostrand Reinhold, 1988.

GAUD00 Gaudin, S. "The Omega Files." *Network World*, June 26, 2000.

GOLL99 Gollmann, D. *Computer Security*. New York: Wiley, 1999.

GUTM02 Gutmann, P. "PKI: It's Not Dead, Just Resting." *Computer*, August 2002.

HARL01 Harley, D.; Slade, R.; and Gattiker, U. *Viruses Revealed*. New York: Osborne/McGraw Hill, 2001.

HEBE92 Heberlein, L.; Mukherjee, B.; and Levitt, K. "Internetwork Security Monitor: An Intrusion Detection System for Large-Scale Networks." *Proceedings, 15th National Computer Security Conference*, October 1992.

HELD96 Held, G. *Data and Image Compression: Tools and Techniques*. New York: Wiley, 1996.

HONE01 The Honeynet Project. *Know Your Enemy: Revealing the Security Tools, Tactics, and Motives of the Blackhat Community*. Reading, MA: Addison-Wesley, 2001.

HUIT98 Huitema, C. *IPv6: The New Internet Protocol*. Upper Saddle River, NJ: Prentice Hall, 1998.

IANS90 I'Anson, C., and Mitchell, C. "Security Defects in CCITT Recommendation X.509 – The Directory Authentication Framework." *Computer Communications Review*, April 1990.

ILGU93 Ilgun, K. "USTAT: A Real-Time Intrusion Detection System for UNIX." *Proceedings, 1993 IEEE Computer Society Symposium on Research in Security and Privacy*, May 1993.

JAVI91 Javitz, H., and Valdes, A. "The SRI IDES Statistical Anomaly Detector." *Proceedings, 1991 IEEE Computer Society Symposium on Research in Security and Privacy*, May 1991.

JIAN02 Jiang, G. "Multiple Vulnerabilities in SNMP." *Security and Privacy Supplement to Computer Magazine*, 2002.

JUEN85 Jueneman, R.; Matyas, S.; and Meyer, C. "Message Authentication." *IEEE Communications Magazine*, September 1988.

KENT00 Kent, S. "On the Trail of Intrusions into Information Systems." *IEEE Spectrum*, December 2000.

KEPH97a Kephart, J.; Sorkin, G.; Chess, D.; and White, S. "Fighting Computer Viruses." *Scientific American*, November 1997.

KEPH97b Kephart, J.; Sorkin, G,; Swimmer, B.; and White, S. "Blueprint for a Computer Immune System." *Proceedings, Virus Bulletin International Conference*, October 1997.

KLEI90 Klein, D. "Foiling the Cracker: A Survey of, and Improvements to, Password Security." *Proceedings, UNIX Security Workshop II*, August 1990.

KNUD98 Knudsen, L., et al. "Analysis Method for Alleged RC4." *Proceedings, ASIACRYPT '98*, 1998.

KOBL92 Koblas, D., and Koblas, M. "SOCKS." *Proceedings, UNIX Security Symposium III*, September 1992.

KOHL89 Kohl, J. "The Use of Encryption in Kerberos for Network Authentication." *Proceedings, Crypto '89*, 1989; published by Springer-Verlag.

KOHL94 Kohl, J.; Neuman, B.; and Ts'o, T. "The Evolution of the Kerberos Authentication Service." In Brazier, F., and Johansen, D. *Distributed Open Systems*. Los Alamitos, CA: IEEE Computer Society Press, 1994. Available at http://web.mit.edu/kerberos/www/papers.html.

KUMA97 Kumar, I. *Cryptology*. Laguana Hills, CA: Aegean Park Press, 1997.

LEUT94 Leutwyler, K. "Superhack." *Scientific American*, July 1994.

LODI98 Lodin, S., and Schuba, C. "Firewalls Fend Off Invasions from the Net." *IEEE Spectrum*, February 1998.

LUNT88 Lunt, T., and Jagannathan, R. " A Prototype Real-Time Intrusion-Detection Expert System." *Proceedings, 1988 IEEE Computer Society Symposium on Research in Security and Privacy*, April 1988.

MACG97 Macgregor, R.; Ezvan, C.; Liguori, L.; and Han, J. *Secure Electronic Transactions: Credit Card Payment on the Web in Theory and Practice*. IBM RedBook SG24-4978-00, 1997. Available at www.redbooks.ibm.com.

MADS93 Madsen, J. "World Record in Password Checking." *Usenet, comp.security.misc news-group*, August 18, 1993.

MANT01 Mantin, I., Shamir, A. "A Practical Attack on Broadcast RC4." *Proceedings, Fast Software Encryption*, 2001.

MARK97 Markham, T. "Internet Security Protocol." *Dr. Dobb's Journal*, June 1997.

MCHU00 McHugh, J.; Christie, A.; and Allen, J. "The Role of Intrusion Detection Systems." *IEEE Software*, September/October 2000.

MEIN01 Meinel, C. "Code Red for the Web." *Scientific American*, October 2001.

MENE97 Menezes, A.; van Oorschot, P.; and Vanstone, S. *Handbook of Applied Cryptography*. Boca Raton, FL: CRC Press, 1997.

MERK97 Merkle, R. Secrecy**,** *Authentication, and Public Key Systems*. PHD Thesis, Stanford University, June 1979.

MERK89 Merkle, R. "One Way Hash Functions and DES." *Proceedings, CRYPTO '89, 1989*;published by Springer-Verlag.

MEYE82 Meyer, C., and Matyas, S. *Cryptography: A New Dimension in Computer Data Security*. New York: Wiley, 1982.

MILL88 Miller, S.; Neuman, B.; Schiller, J.; and Saltzer, J. "Kerberos Authentication and Authorization System." *Section E.2.1, Project Athena Technical Plan*, M.I.T. Project Athena, Cambridge, MA. 27 October 1988.

MIRK04 Mirkovic, J., and Relher, P. "A Taxonomy of DDoS Attack and DDoS Defense Mechanisms." ACM S*IGCOMM Computer Communications Review*, April 2004.

MIST98 Mister, S., and Tavares, S. "Cryptoanalysis of RC4-Like Ciphers." *Proceedings, Workshop in Selected Areas of Cryptography, SAC' 98*. 1998.

MITC90 Mitchell, C.; Walker, M.; and Rush, D. "CCITT/ISO Standards for Secure Message Handling." *IEEE Journal on Selected Areas in Communications*, May 1989.

MOOR01 Moore, M. "Inferring Internet Denial-of-Service Activity." *Proceedings of the 10th USENIX Security Symposium*, 2001.

NACH97 Nachenberg, C. "Computer Virus-Antivirus Coevolution." *Communications of the ACM*, January 1997.

NEED78 Needham, R., and Schroeder, M. "Using Encryption for Authentication in Large Networks of Computers." *Communications of the ACM*, December 1978.

NING04 Ning, P., et al. "Techniques and Tools for Analyzing Intrusion Alerts." *ACM Transactions on Information and System Security*, May 2004.

OPPL97 Oppliger, R. "Internet Security: Firewalls and Beyond." *Communications of the ACM*, May 1997.

OPPL05 Oppliger, R., and Rytz, R. "Does Trusted Computing Remedy Computer Security Problems?" *IEEE Security and Privacy*, March/April 2005.

PATR04 Patrikakis, C.; Masikos, M.; and Zouraraki, O. "Distributed Denial of Service Attacks." *The Internet Protocol Journal*, December 2004.

PAUL03 Paul, S., and Preneel, B. "Analysis of Non-fortuitous Predictive States of the RC4 Keystream Generator." *Proceedings, INDOCRYPT '03*, 2003.

PAUL04 Paul, S., and Preneel, B. "A New Weakness in the RC4 Keystream Generator and an Approach to Improve the Secutity of the Cipher." *Proceedings, Fast Software Encryption*, 2004.

PERL99 Perlman, R. "An Overview of PKI Trust Models." *IEEE Network*, November/December 1999.

PFLE03 Pfleeger, C. *Security in Computing*. Upper Saddle River, NJ: Prentice Hall, 2003.

PIAT91 Piattelli-Palmarini, M. "Probability: Neither Rational nor Capricious." *Bostonia*, March 1991.

PIEP03 Pieprzyk, J.; Hardjono, T.; and Seberry, J. *Fundamentals of Computer Security*. New York: Springer-Verlag, 2003.

PORR92 Porras, P. *STAT: A State Transition Analysis Tool for Intrusion Detection*. Master's Thesis, University of California at Santa Barbara, July 1992.

PREN02 Preenel, B. "New European Schemes for Signature, Integrity and Encryption (NESSIE): A Status Report." *Proceedings of the 5th International Workshop on Practice and Theory in Public Key Cryptosystems: Public Key Cryptography*. 2002.

PROC01 Proctor, P., *The Practical Intrusion Detection Handbook*. Upper Saddle River, NJ: Prentice Hall, 2001.

PUDO02 Pudovkina, M. "Statistical Weaknesses in the Alleged RC4 Keystream Generator." *Proceedings, 4th International Workshop on Computer Science and Information Technologies*, 2002.

RESC01 Rescorla, E. SSL and TLS: *Designing and Building Secure Systems*. Reading, MA: Addison-Wesley, 2001.

RIVE78 Rivest, R.; Shamir, A.; and Adleman, L. "A Method for Obtaining Digital Signatures and Public Key Cryptosystems." *Communications of the ACM*, February 1978.

ROBS95a Robshaw, M. *Stream Ciphers*. RSA Laboratories Technical Report TR-701 , July 1995. http://www.rsasecurity.com/rsalabs

ROBS95b Robshaw, M. *Block Ciphers*. RSA Laboratories Technical Report TR-601, August 1995. http://www.rsasecurity.com/rsalabs

RODR02 Rodriguez, A.,A., et al. *TCP/IP Tutorial and Technical Overview*. Upper Saddle River: NJ: Prentice Hall, 2002.

SAFF93 Safford, D.; Schales, D.; and Hess. D. "The TAMU Security Package: An Ongoing Response to Internet Intruders in an Academic Environment." *Proceedings, UNIX Security Symposium IV*, October 1993.

SCHN00 Schneier, B. *Secrets and Lies: Digital Security in a Networked World*. New York: Wiley 2000.

SCHN96 Schneier, B. *Applied Cryptography*. New York: Wiley, 1996.

SIMM92 Simmons, G., ed. *Contemporary Cryptology: The Science of Information Integrity*. Piscataway, NJ; IEEE Press 1992.

SING99 Singh, S. The Code Book: *The Science of Secrecy from Ancient Egypt to Quantum Cryptography*. New York: Anchor Books, 1999.

SMIT97 Smith, R. *Internet Cryptography*. Reading, MA: Addison-Wesley, 1997.

SNAP91 Snapp, S., et al. "A System for Distributed Intrusion Detection." *Proceedings, COMPCON Spring '91*, 1991.

SPAF92a Spafford, E. "Observing Reusable Password Choices." *Proceedings, UNIX Security Symposium III*, September 1992.

SPAF92b Spafford, E. "OPUS: Preventing Weak Password Choices." *Computers and Security*, No. 3, 1992.

STAL99 Stallings, W. *SNMP, SNMPv2, SNMPv3, and RMON 1 and 2*. Reading, MA: Addison-Wesley, 1999.

STAL04 Stallings, W. *Computer Networking with Internet Protocols and Technology*. Upper Saddle River, NJ: Prentice Hall, 2004.

STAL06a Stallings, W. *Cryptography and Network Security: Principles and Practice, Fourth Edition.* Upper Saddle River, NJ: Prentice Hall, 2006.

STAL06b Stallings, W. "The Whirlpool Secure Hash Function." *Cryptologia*, January 2006.

STEI88 Steiner, J.; Neuman, C.; and Schiller, J. "Kerberos: An Authentication Service for Open Networked Systems." *Proceedings of the Winter 1988 USENIX Conference*, February 1988.

STEP93 Stephenson, P. "Preventive Medicine." *LAN Magazine*, November 1993.

STER92 Sterling, B. *The Hacker Crackdown: Law and Disorder on the Electronic Frontier.* New York: Bantam, 1992.

STIN06 Stinson, D. *Cryptography: Theory and Practice*. Boca Raton, FL: CRC Press, 2006.

STOL88 Stoll, C. "Stalking the Wiley Hacker." *Communications of the ACM*, May 1988.

STOL89 Stoll, C. *The Cuckoo's Egg*. New York: Doubleday, 1989.

SZOR05 Szor, P., *The Art of Computer Virus Research and Defense*. Reading, MA: Addison-Wesley, 2005.

THOM84 Thompson, K. "Reflections on Trusting Trust (Deliberate Software Bugs)." *Communications of the ACM*, August 1984.

TIME90 Time, Inc. *Computer Security, Understanding Computers Series*. Alexandria, VA: Time-Life Books, 1990.

TSUD92 Tsudik, G. "Message Authentication with One-Way Hash Functions." *Proceedings, INFOCOM '92*, May 1992.

TUNG99 Tung, B. Kerberos: *A Network Authentication System*. Reading, MA:Addison-Wesley, 1999.

VACC89 Vaccaro, H., and Liepins, G. "Detection of Anomalous Computer Session Activity." *Proceedings of the IEEE Symposium on Research in Security and Privacy*, May 1989.

VIJA02 Vijayan, J. "Denial-of-Service Attacks Still a Threat." *Computer World*, April 8, 2002.

WACK02 Wack, J.; Cutler, K.; and Pole, J. *Guidelines on Firewalls and Firewall Policy*. NIST Special Publications SP 800-41, January 2002.

WAGN00 Wagner, D., and Goldberg, I. "Proofs of Security for the UNIX Password Hashing Algorithm." *Proceedings, ASIACRYPT '00*, 2000.

WANG05 Wang, X.; Yin, Y.;and Yu, H. "Finding Collisions in the Full SHA-1. *Proceedings, Crypto'05*, 2005; published by Springer-Verlag.

WILS05 Wilson, J. "The Future of the Firewall." *Business Communications Review*, May 2005.

YUVA79 Yuval, G. "How to Swindle Rabin." *Cryptologia*, July 1979.

ZIV77 Ziv, J., and Lempel, A. "A Universal Algorithm for Sequential Data Compression." *IEEE Transactions on Information Theory*, May 1977.

واژه‌نامه انگلیسی– فارسی

abort	نادیده گرفتن
access control	کنترل دست یابی
acquirer	مباشر- فراهم‌کننده
active attack	حملهٔ فعال
affiliation	پیوستگی
agent	عامل
aggregate	تراکم
alarm	هشدار
algorithm	الگوریتم
anomaly	ناهنجاری
append	وصل کردن
arbiter	داور
association	اتحاد
asymmetric	نامتقارن
audit	بازرسی، ممیزی
authentic	معتبر
authentication	اعتبارسنجی
authentication header	سرآیند اعتبارسنجی
authenticator	اعتبارسنج
autoexecute	خوداجرا
availibility service	قابلیت دسترسی
backdoor	درب مخفی
bait	طعمه
bastion	دژ
batch	دسته
benign	بی خطر
block	بلوک
block cipher	رمز قالبی
body	بدنه
bog	باتلاق
bogus	ساختگی
boot-sector virus	ویروس بخش راه اندازی
bottleneck	گلوگاه
browser	مرورگر
brute-force attack	حملهٔ همه جانبه
bug	اشکال
byte	بایت
canonical	قانونی

CAST-128	نام یک الگوریتم رمزنگاری
certificate	گواهی‌نامه
chaining	زنجیر کردن
checksum	جمع کنترلی
cipher	رمز
ciphertext	متن رمزشده
ciphertext-only	فقط- متن رمزشده
clandestine	خفیه
client	کلاینت
clogging	انسداد
codebook	کتاب کُد
community	جامعه
compatibility	سازگاری
compiler	کامپایلر
component	مؤلفه
compression	فشرده سازی
compromise	لو رفتن
computationally secure	از نظر محاسباتی امن
computer	رایانه
computer security	امنیت رایانه
concatenation	جمع رشته ای
concievable	قابل تصور
confidentiality	محرمانگی
conformance	مطابقت
confusion	سردرگُمی
connection	اتصال
connection oriented	اتصال گرا
connectionless	بدون اتصال
connection-request	درخواست ارتباط
consensus	مطابقت
context	مقوله
conventional	قراردادی
cookie	کوکی
counter	شمارنده
covert	پنهان
cracker	شکننده
cryptoanalysis	شکستن رمز
cryptoanalyst	شکنندهٔ رمز
cryptography	علم رمزنگاری
cryptology	علم رمزشناسی
daemon	دیو
data	داده
decode	کُدگشائی
decoy	دام
decryption	رمزگشائی
detached	جداشده- مجزا

diffusion	انتشار
digest	چکیده
digital signature	امضای دیجیتال
digram	دو- حرفی
directory	فهرست راهنما
discretionary access cotrol	کنترل دست‌یابی منصفانه
dispatcher	حمل‌کننده
disruptive	مخل
distributed enviroments	محیط های توزیع شده
domain	دامنه- قلمرو
dual	دوگانه
eavesdropping	استراق سمع- شنود
editor	ویرایش‌گر
elliptic curve	خَم بیضوی
email	پست الکترونیک
emoticon	احساس‌نما
encryption	رمزنگاری
encryption devices	تجهیزات رمزنگاری
end system	سیستم انتهائی
end-to-end	سر- به - سر
ephermal	یکبارمصرف
error-detection	تشخیص خطا
expert system	سیستم خبره
exploit	بهره برداری
extension	الحاقیه
external	خارجی
extranet	اکسترانت
fallacy	خطا
fax-back	فاکس برگردان
feasible	مقدور
feedback	بازخورد
Feistel	نام یک ساختار رمزنگاری متقارن
field	میدان
fingerprint	اثر انگشت
firewall	دیوارآتش
flaw	خطا
flooding	بمباران
foil	خنثی کردن
forgery	تقلب- جعل
fragmentation	قطعه قطعه کردن
framework	چهارچوب
frond-end processor	پردازشگر خط اول
gateway	دروازه
gauge	پیمانه
genuine	دست اول
hacker	هَکِر- نفوذگر

handshaking	دستداد
hash function	تابع درهم ساز
header	سَرآیند
heuristic	تاریخی
hostile	خصمانه
hub	هاب
IDEA	نام یک الگوریتم رمزنگاری
identifier	شناسه
immune	مصون
impersonation	جعل هویت
indispensible	ضروری
information security	امنیت اطلاعات
initiator	آغازگر
inner	درونی
innocuous	بی آزار
integrity	صحت-اصالت
intercept	قطع کردن
internal	داخلی
internet	بین شبکه ای
Internet	اینترنت
Internet Association	انجمن اینترنت
Internet Draft	پیش‌نویس اینترنت
Internet Publication	انتشارات اینترنت
internet security	امنیت اینترنت
Internet society	جامعهٔ اینترنت
Internet Standard	استاندارد اینترنت
intranet	اینترانت
intruder	مهاجم
intrusion	تهاجم
Kerberos	نام یک سرویس اعتبارسنجی
key	کلید
key distribution	توزیع کلید
key exchange	مبادلهٔ کلید
key length	طول کلید
key management	مدیریت کلید
key ring	دسته کلید
key space	فضای کلید
keystone	سنگ بنا
label	برچسب
legitimacy	مشروعیت
legitimate	مشروع- قانونی
lifetime	طول عمر
link	پیوند
loading	بارگذاری
logic bomb	بمب لاجیک
logical connection	اتصال منطقی

logon	اتصال به سیستم
macro	ماکرو
macro virus	ویروس ماکرو
mailbox	صندوق پستی
malicious	بداندیش
malware	بدافزار
mapping	نگاشت
masquerade	بالماسکه
masquerader	نقاب دار
master key	کلید اصلی
mediation	وساطت
memory-resident virus	ویروس مستقر در حافظه
merchant	تاجر
message	پیام
metmorphic virus	ویروس دگردیس
metric	معیار
misfeasor	سوءاستفاده‌کننده
mode	مُود
modification	دستکاری
modular	پیمانه ای
modulo	پیمانه
monitoring	پایش- نظارت
motivation	انگیزش
multiplicative inverse	معکوس ضربی
mutation engine	موتور تغییر
mutual	متقابل
native	بومی
nesting	لانه سازی
network security	امنیت شبکه
newsgroup	گروه خبری
node	گره
nonce	حال فعلی
nonrepudiation	عدم انکار
notation	علامت اختصاری
notification	تذکر
notion	آگاهی
obfuscation	ابهام زائی
object	موضوع
octet	اُکتت
offline	خارج از خط
one-time key	کلید یکبارمصرف
one-way	یکطرفه
online	برخط
orphan	یتیم
outer	بیرونی
overhead	سرباره

overlap	هم‌پوشانی
overt	آشکار
packet	بسته
packet switch	سوئیچ بسته‌ای
pad	لائی
padding	لائی‌گذاری
paradigm	پارادایم
parasitic virus	ویروس انگلی
passive attack	حملهٔ غیرفعال
passphrase	جملهٔ عبور
password	کلمهٔ عبور
patent	ثبت اختراع
pattern	الگو
payload	محموله
peer	نظیر
peer-to-peer	نظیر- به- نظیر
penetration	نفوذ
permutation	جایگشت
plaintext	متن ساده
platform	سیستم عامل - کامپیوتر
polymorphic virus	ویروس چندچهره
port	درگاه
precedence	حق تقدم
primitive root	ریشهٔ اولیه
principal	رئیس
private key	کلید خصوصی
procedure	رَویه
processor	پردازش گر
product system	سیستم ترکیبی
promulgate	اعلام کردن
protocol	پروتکل
proxy	پروکسی
pseudorandom	شبه تصادفی
public key	کلید عمومی
radix-64	تبدیلradix-64
random	تصادفی
range	بُرد
realm	قلمرو
receipt	رسید
recovery	بازیابی
reference monitor	پایشگر مرجع
reliability	قابلیت اعتماد
reliable	قابل اعتماد
replay	بازخوانی
resource allocation	تخصیص منابع
reversible	برگشت‌پذیر

revokation	ابطال- لغو
round	دُور
round-function	تابع دُور
router	مسیریاب
rudimentary	مقدماتی
rule-based	مبتنی بر قاعده
scalable	مقیاس‌پذیر
scrambled	درهم ریخته
secret	سرّی
secret key	کلید سرّی
secure	امن
security	امنیت
security architecture	معماری امنیت
security association	اتحاد امنیتی
security attack	حملهٔ امنیتی
security flaw	نقص امنیتی
security mechanism	سازوکار امنیتی
security service	سرویس امنیتی
seed	بذر
segment	سگمنت
sequence number	شَمارهٔ ردیف
server	سرور
session	اجَلاس
session key	کلید اجلاس
shareware	اشتراک‌افزار
signatory	امضاءکننده
signature	امضاء
smiley	خندانک
sniffer	بو کشنده
socket	سوکت
spam	نامه الکترونیکی ناخواسته
spoofing	جعل - تقلید
spurious	ساختگی- نادرست
stack	پشته
stateful inspection firewall	نوعی دیوار آتش
stealth virus	ویروس پنهان‌شونده
stream	دنباله
stream cipher	رمز دنباله‌ای
subkey	زیرکلید
substitution	جایگزینی
suite	مجموعه
symmetric encryption	رمزنگاری متقارن
tag	دنباله
terminal	پایانه
theft	دزدی
thread	رشته

threat	تهدید
threshold	آستانه
throughput	توان عملیاتی
ticket	بلیت
timely	بهنگام
time-sharing	اشتراک زمانی
timestamp	برچسب زمانی
traditional	سنّتی
traffic analysis	تحلیل ترافیک
trailer	تَه آیند
transaction	سند
transformation	تبدیل
transparent	شفاف
transport	حمل‌ونقل- ترابری
transposition	جابجائی
trapdoor	درب مخفی
trend	رَوَند
trespass	تعرض
trigger	ماشه‌کشی
Trojan Horse	اسب تروا
trust	اعتماد
unique	یکتا
update	به روز درآوردن
utility program	برنامهٔ کمکی
utilization	بهره گیری
vendor	فروشنده
version	نسخه
view	منظر
virtual	مجازی
virtual circuit	مدار مجازی
virus	ویروس
vulneribility	آسیب‌پذیری
web	وِب
webmail	پست مبتنی بر وِب
website	وب سایت
wildcard	عَام - عمومی
worm	کِرم
zombie	زَامبی

واژه‌نامه فارسی– انگلیسی

آستانه	threshold
آسیب‌پذیری	vulneribility
آشکار	overt
آغازگر	initiator
آگاهی	notion
ابطال- لغو	revokation
ابهام زائی	obfuscation
اتحاد	association
اتحاد امنیتی	security association
اتصال	connection
اتصال به سیستم	logon
اتصال‌گرا	connection oriented
اتصال منطقی	logical connection
اثر انگشت	fingerprint
اجلاس	session
احساس نما	emoticon
از نظر محاسباتی امن	computationally secure
اسب تروا	Trojan Horse
استاندارد اینترنت	Internet Standard
استراق سمع- شنود	eavesdropping
اشتراک‌افزار	shareware
اشتراک زمانی	time-sharing
اشکال	bug
اعتبارسنج	authenticator
اعتبارسنجی	authentication
اعتماد	trust
اعلام کردن	promulgate
اُکتت	octet
اکسترانت	extranet
الحاقیه	extension
الگو	pattern
الگوریتم	algorithm
امضاء	signature
امضاءکننده	signatory
امضای دیجیتال	digital signature
امن	secure
امنیت	security
امنیت اطلاعات	information security

امنیت اینترنت	internet security
امنیت رایانه	computer security
امنیت شبکه	network security
انتشار	diffusion
انتشارات اینترنت	Internet Publication
انجمن اینترنت	Internet Association
انسداد	clogging
انگیزش	motivation
اینترانت	intranet
اینترنت	Internet
بارگذاری	loading
بازخوانی	replay
بازخورد	feedback
بازرسی ، ممیزی	audit
بازیابی	recovery
باتلاق	bog
بالماسکه	masquerade
بایت	byte
بدافزار	malware
بداندیش	malicious
بدنه	body
بدون اتصال	connectionless
بذر	seed
برچسب	label
برچسب زمانی	timestamp
برخط	online
بُرد	range
برگشت‌پذیر	reversible
برنامهٔ کمکی	utility program
بسته	packet
بلوک	block
بلیت	ticket
بمب لاجیک	logic bomb
بمباران	flooding
به روز درآوردن	update
بهره برداری	exploit
بهره گیری	utilization
بهنگام	timely
بو کشنده	sniffer
بومی	native
بی آزار	innocuous
بی خطر	benign
بیرونی	outer
بین شبکه‌ای	internet
پارادایم	paradigm
پایانه	terminal

monitoring	پایش- نظارت
reference monitor	پایشگر مرجع
processor	پردازش‌گر
frond-end processor	پردازشگر خط اول
protocol	پروتکل
proxy	پروکسی
email	پست الکترونیک
webmail	پست مبتنی بر وِب
stack	پشته
covert	پنهان
message	پیام
Internet Draft	پیش نویس اینترنت
gauge	پیمانه
modulo	پیمانه
modular	پیمانه ای
affiliation	پیوستگی
link	پیوند
hash function	تابع درهم ساز
round-function	تابع دُور
merchant	تاجر
heuristic	تاریخی
transformation	تبدیل
radix-64	تبدیلradix-64
encryption devices	تجهیزات رمزنگاری
traffic analysis	تحلیل ترافیک
resource allocation	تخصیص منابع
notification	تذکر
aggregate	تراکم
error-detection	تشخیص خطا
random	تصادفی
trespass	تعرض
forgery	تقلب- جعل
trailer	تَه‌آیند
intrusion	تهاجم
threat	تهدید
throughput	توان عملیاتی
key distribution	توزیع کلید
patent	ثبت اختراع
transposition	جابجائی
Internet society	جامعهٔ اینترنت
community	جامعه
substitution	جایگزینی
permutation	جایگشت
detached	جداشده- مجزا
spoofing	جعل - تقلید
impersonation	جعل هویت

جمع رشته‌ای	concatenation
جمع کنترلی	checksum
جملهٔ عبور	passphrase
چکیده	digest
چهارچوب	framework
حال فعلی	nonce
حق تقدم	precedence
حمل‌کننده	dispatcher
حمل‌ونقل- ترابری	transport
حملهٔ امنیتی	security attack
حملهٔ غیرفعال	passive attack
حملهٔ فعال	active attack
حملهٔ همه جانبه	brute-force attack
خارج از خط	offline
خارجی	external
خصمانه	hostile
خطا	flaw
خطا	fallacy
خفیه	clandestine
خَم بیضوی	elliptic curve
خنثی کردن	foil
خندانک	smiley
خوداجرا	autoexecute
داخلی	internal
داده	data
دام	decoy
دامنه- قلمرو	domain
داور	arbiter
درب مخفی	trapdoor
درب مخفی	backdoor
درخواست ارتباط	connection-request
درگاه	port
درهم ریخته	scrambled
دروازه	gateway
درونی	inner
دزدی	theft
دژ	bastion
دست اول	genuine
دستداد	handshaking
دستکاری	modification
دسته	batch
دسته کلید	key ring
دنباله	tag
دنباله	stream
دو- حرفی	digram
دُور	round

dual	دوگانه
daemon	دیو
firewall	دیوارآتش
principal	رئیس
computer	رایانه
receipt	رسید
thread	رشته
cipher	رمز
stream cipher	رمز دنباله‌ای
block cipher	رمز قالبی
decryption	رمزگشائی
encryption	رمزنگاری
symmetric encryption	رمزنگاری متقارن
trend	رَوَند
procedure	رَویه
primitive root	ریشهٔ اولیه
zombie	زامبی
chaining	زنجیرکردن
subkey	زیرکلید
bogus	ساختگی
spurious	ساختگی- نادرست
compatibility	سازگاری
security mechanism	سازوکار امنیتی
end-to-end	سر- به - سر
header	سَرآیند
authentication header	سرآیند اعتبارسنجی
overhead	سرباره
confusion	سردرگُمی
secret	سرّی
server	سِرور
security service	سِرویس امنیتی
segment	سِگمنت
transaction	سَند
keystone	سنگ بنا
traditional	سنّتی
misfeasor	سوءاستفاده‌کننده
packet switch	سوئیچ بسته‌ای
socket	سوکت
end system	سیستم انتهائی
product system	سیستم ترکیبی
expert system	سیستم خبره
platform	سیستم عامل - کامپیوتر
pseudorandom	شبه تصادفی
transparent	شفاف
cryptoanalysis	شکستن رمز
cryptoanalyst	شکنندهٔ رمز

شکننده	cracker
شمارهٔ ردیف	sequence number
شمارنده	counter
شناسه	identifier
صحت-اصالت	integrity
صندوق پستی	mailbox
ضروری	indispensible
طعمه	bait
طول عمر	lifetime
طول کلید	key length
عام - عمومی	wildcard
عامل	agent
عدم انکار	nonrepudiation
علامت اختصاری	notation
علم رمزشناسی	cryptology
علم رمزنگاری	cryptography
فاکس برگردان	fax-back
فروشنده	vendor
فشرده‌سازی	compression
فضای کلید	key space
فقط- متن رمزشده	ciphertext-only
فهرست راهنما	directory
قابل اعتماد	reliable
قابل تصور	concievable
قابلیت اعتماد	reliability
قابلیت دسترسی	availibility service
قانونی	canonical
قراردادی	Conventional
قطع کردن	intercept
قطعه قطعه کردن	fragmentation
قلمرو	realm
کامپایلر	compiler
کتاب کُد	codebook
کُدگشائی	decode
کِرم	worm
کَلاینت	client
کلمهٔ عبور	password
کلید	key
کلید اجلاس	session key
کلید اصلی	master key
کلید خصوصی	private key
کلید سرّی	secret key
کلید عمومی	public key
کلید یکبارمصرف	one-time key
کنترل دست‌یابی	access control
کنترل دست‌یابی منصفانه	discretionary access cotrol

cookie	کوکی
node	گره
newsgroup	گروه خبری
bottleneck	گلوگاه
certificate	گواهی نامه
pad	لائی
padding	لائی‌گذاری
nesting	لانه سازی
compromise	لو رفتن
component	مؤلفه
trigger	ماشه کشی
macro	ماکرو
key exchange	مبادلهٔ کلید
acquirer	مباشر- فراهم کننده
rule-based	مبتنی بر قاعده
mutual	متقابل
ciphertext	متن رمزشده
plaintext	متن ساده
virtual	مجازی
suite	مجموعه
confidentiality	محرمانگی
payload	محموله
distributed enviroments	محیط‌های توزیع‌شده
disruptive	مخل
virtual circuit	مدار مجازی
key management	مدیریت کلید
browser	مرورگر
router	مسیریاب
legitimate	مشروع- قانونی
legitimacy	مشروعیت
immune	مصون
consensus	مطابقت
conformance	مطابقت
authentic	معتبر
multiplicative inverse	معکوس ضربی
security architecture	معماری امنیت
metric	معیار
rudimentary	مقدماتی
feasible	مقدور
context	مقوله
scalable	مقیاس‌پذیر
view	منظر
intruder	مهاجم
mutation engine	موتور تغییر
mode	مُود
object	موضوع

میدان	field
نادیده گرفتن	abort
نام یک ساختار رمزنگاری متقارن	Feistel
نام یک سرویس اعتبارسنجی	Kerberos
نام یک سیستم رمزنگاری	IDEA
نام یک سیستم رمزنگاری	CAST-128
نامتقارن	Asymmetric
نامهٔ الکترونیکی ناخواسته	spam
ناهنجاری	anomaly
نسخه	version
نظیر	peer
نظیر- به- نظیر	peer-to-peer
نفوذ	penetration
نقاب دار	masquerader
نقص امنیتی	security flaw
نگاشت	mapping
نوعی دیوار آتش	stateful inspection firewall
هاب	hub
هشدار	alarm
هَکِر- نفوذگر	hacker
هم پوشانی	overlap
وِب	web
وِب سایت	website
وَساطت	mediation
وصل کردن	append
ویرایش‌گر	editor
ویروس	virus
ویروس انگلی	parasitic virus
ویروس بخش راه اندازی	boot-sector virus
ویروس پنهان شونده	stealth virus
ویروس چندچهره	polymorphic virus
ویروس دگردیس	metmorphic virus
ویروس ماکرو	macro virus
ویروس مستقر در حافظه	memory-resident virus
یتیم	orphan
یک الگوریتم رمزنگاری نامتقارن	RSA
یکبارمصرف	ephermal
یکتا	unique
یکطرفه	one-way

علائم اختصاری

3DES	Triple Data Encryption Standard
AES	Advanced Encryption Standard
AH	Authentication Header
ANSI	American National Standards Institute
AS	Authentication Server
BBS	Bulletin Board System
BCP	Best Current Practice
CBC	Cipher Block Chaining
CC	Common Criteria
CERT	Computer Emergency Response Team
CESG	Communications-Electronics Security group
CFB	Cipher Feedback
CMAC	Cipher-Based Message Authentication Code
CRT	Chinese Remainder Theorem
DDoS	Distributed Denial of Service
DEA	Data Encryption Algorithm
DES	Data Encryption Standard
DH	Diffie-Hellman
DOI	Domain of Interpretation
DoS	Denial of Service
DSA	Digital Signature Algorithm
DSS	Digital Signature Standard
ECB	Electronic Codebook
ESP	Encapsulating Security Payload
FCS	Frame Check Sequence
FIPS	Federal Information Processing Standard
FTP	File Transfer Protocol
HAR	Host Audit Protocol
HTTP	Hyper Text Transfer Protocol
IAB	Internet Architecture Board
IESG	Internet Engineering Steering Group
IETF	Internet Engineering Task Force
IMAP	Internet Mail Access Protocol
IP	Internet Protocol
IPSec	IP Security
ISO	International Organization for Standardization
ISP	Internet Service Provider
ITU	International Telecommunication Union
ITU-T	ITU Telecommunication Standardization Sector
IV	Initialization Vector
KDC	Key Distribution Center
LAN	Local Area Network
MAC	Message Authentication Code
MIB	Management Information Base
MIC	Message Integrity Code
MIME	Multipurpose Internet Mail Extension
MD5	Message Digest, Version 5
MTA	Mail Transfer Agent
MTU	Maximum Transmission Unit
MUA	Mail User Agent
NIST	National Institute of Standards and Technology
NSA	National Security Agency
OFB	Output Feedback
OSI	Open System Interconnection
OSPF	Open Shortest Path First
PCBC	Propagating Cipher Block Chaining
PDU	Protocol Data Unit
PGP	Pretty Good Privacy
PKI	Public Key Infrastructure
POP3	Post Office Protocol, Version 3
PRNG	Pseudorandom Number Generator
RFC	Request for Comments
RNG	Random Number Generator
RSA	Rivest-Shamir-Adelman
SET	Secure Electronic Transaction
SHA	Secure Hash Algorithm
SHS	Secure Hash Standard
S/MIME	Secure MIME
SMTP	Simple Mail Transfer Protocol
SNMP	Simple Network Management Protocol
SNMPv3	Simple Network Management Protocol,Version 3
SPI	Security Parameters Index
SPD	Security Policy Database
SSL	Secure Sockets Layer
ST	Security Target
TCP	Transmission Control Protocol
TFTP	Trivial File Transfer Protocol
TGS	Ticket Granting Service
TLS	Transport Layer Security
TOE	Target of Evaluation
TS	Technical Specification
UDP	User Datagram Protocol
USM	User Security Model
VACM	View-Based Access Control Mode
VPN	Virtual Private Network
WAN	Wide Area Network