فصل هشتم
امنیت شبکه

شبکه‌های کامپیوتری در چند دهه‌ی اولیه‌ی پیدایش‌شان، توسط پژوهشگرانِ دانشگاه‌ها جهت ارسال ایمیل، و نیز توسط کارکنان شرکت‌ها جهت به اشتراک‌گذاری چاپگرها استفاده می‌شدند. در چنین شرایطی امنیت چندان مورد توجه نبود. اما اکنون که میلیون‌ها نفر از شهروندان عادی برای انجام کارهای بانکی، خرید، و امور مالیاتیِ خود از شبکه‌ها استفاده می‌کنند و نقاط ضعف متعددی یکی بعد از دیگری آشکار می‌شوند، مسئله‌ی امنیت سهم عظیمی را به خود اختصاص داده است. در این فصل امنیت شبکه را از چند زاویه مطالعه خواهیم کرد، دام‌های بیشماری را بررسی نموده، و الگوریتم‌ها و پروتکل‌های متعددی که در رابطه با امن‌تر کردنِ شبکه‌ها هستند را مورد بررسی قرار خواهیم داد.

امنیت یک عنوان کلی است و انبوهی از خطاها و لغزش‌ها را پوشش می‌دهد. امنیت در ساده‌ترین شکلش عبارت‌است‌از اطمینان از این‌که افراد کنجکاو نتوانند پیغام‌های مربوط به دیگران را بخوانند و یا حتی بدتر از آن، یعنی این‌که نتوانند به طور مخفیانه این پیغام‌ها را تغییر دهند. امنیت در ارتباط با افرادی است که سعی دارند به سرویس‌های راه دوری که مجوز استفاده از آن‌ها را ندارند، دسترسی پیدا کنند. امنیت همچنین با روش‌هایی در ارتباط است که پیغامی که ادعا می‌کند مثلاً از IRS (اداره‌ی مالیات‌های امریکا) آمده و دستور می‌دهد مبلغ مشخصی تا زمان معینی باید پرداخت شود، آیا واقعاً از IRS است و یا از طرف مافیا صادر شده! همچنین امنیت با مسائل ناشی از پیغام‌های درست و قانونی که ربوده شده و بازنواخت شده‌اند، و نیز افرادی که تلاش می‌کنند ارسال بعضی پیغام‌های خاص از طرف خود را انکار کنند، مواجه می‌باشد.

اغلبِ مسائل امنیتی به صورت عمدی و توسط افرادی ایجاد می‌شوند که سوءنیت دارند و در پی رسیدن به منافع چشمگیر، جلب توجه، یا صدمه زدن به کسی هستند. تعدادی از متداول‌ترین متخلفان در شکل ۸-۱ فهرست شده‌اند. از این فهرست بایستی روشن باشد که امن ساختنِ یک شبکه مستلزم آن است که کاری بیش از رفع خطاهای برنامه‌نویسی انجام گیرد. این کار مستلزم سبقت گرفتن از حریفانی است که غالباً باهوش، از خود گذشته، و در بعضی مواقع نیز برخوردار از بنیه‌ی مالیِ خوبی

متخلف	هدف
دانشجو	برای آن که محض تفریح در ایمیل مردم سرک بکشد
آشوبگر	برای آزمایش سیستم‌های امنیتی؛ سرقت اطلاعات
فروشنده	برای آن که ادعا کند نماینده‌ی تمام اروپاست
شرکت	برای آن که طرح بازاریابیِ رقیبش را کشف کند
کارمند سابق	برای آن که بابت اخراج شدنش انتقام بگیرد
حسابدار	برای اختلاس از یک شرکت
کارگزار بورس	برای تکذیبِ تعهدی که توسط ایمیل به یک مشتری داده شده
سارق هویت	برای سرقت شماره‌ی کارت اعتباری جهت فروش آن
دولت	برای دستیابی به اسرار نظامی یا صنعتی دشمن
تروریست	برای سرقت اسرار صلاح‌های بیولوژیکی

شکل ۸-۱ بعضی افرادی که ممکن است مسائل امنیتی ایجاد کنند، و علت آن.

می‌باشند. همچنین باید روشن باشد که تدابیری که مانع مهاجمان[1] بی‌دقت خواهند شد، تأثیر کمی بر مهاجمانِ جدّی خواهند داشت. بایگانی پلیس نشان می‌دهد که مخرب‌ترین حمله‌ها از جانب غریبه‌هایی که به یک خط تلفن نفوذ می‌کنند نیست، بلکه از جانب خودی‌هایی است که کینه و رنجشی دارند. سیستم‌های امنیتی بایستی به طور متناسب طراحی شوند.

مسائل مربوط به امنیت شبکه تقریباً می‌توانند به چهار ناحیه که در هم آمیخته شده‌اند، تقسیم شوند: رازپوشی (secrecy)، تصدیق هویت (authentication)، عدم انکار (nonrepudiation)، و کنترل یکپارچگی (integrity control). رازپوشی، که محرمانگی (confidentiality) نیز نامیده می‌شود، باید اطلاعات را از دسترس دست‌های کوچک و آلوده‌ی کاربرانِ غیرمُجاز، محفوظ نگه دارد. این همان چیزی است که مردم در مورد امنیت شبکه به ذهنشان متبادر می‌شود. تصدیق هویت در ارتباط با تعیین هویت کسی است که با او در حال گفتگو هستید، قبل از آن‌که اطلاعات حساس را برایش آشکار کنید یا با او وارد معامله‌ای شوید. عدم انکار در ارتباط با امضاهاست: چطور ثابت می‌کنید که مشتری شما واقعاً سفارش ده میلیون قطعه از کالایی را به ازای هر قطعه ۸۹ سنت داده است، چون ممکن است بعداً ادعا کند که قیمت آن ۶۹ سنت بوده؟ یا ممکن است ادعا کند هرگز سفارشی برای آن کالا نداده است. آخرین مورد، یعنی کنترل یکپارچگی مربوط به این موضوع است که چگونه می‌توانید اطمینان داشته باشید پیغامی که به شما رسیده است، واقعاً همان پیغامی است که ارسال شده، و پیغامی نیست که یک متخلف (adversary) با سوءنیت آن را در میانه‌ی راه تغییر داده و یا از خودش درآورده است.

تمام این مباحث (رازپوشی، تصدیق هویت، عدم انکار، و کنترل یکپارچگی) در سیستم‌های سنتی هم وجود دارند اما با تفاوت‌های قابل ملاحظه. یکپارچگی و رازپوشی توسط استفاده از پُست سفارشی[2]

1. Attacker 2. Registered mail

و قفل زدن به اسناد[1] حاصل می‌شوند. امروزه دستبرد زدن به قطار پُست نسبت به روزگار جسی جیمز، دشوارتر شده است.

همچنین مردم معمولاً قادر به درک تفاوت میان یک سند اصل و یک فتوکپی هستند، و این موضوع غالباً برایشان مهم است. برای آزمایش می‌توانید از یک چک معتبر، یک کپی تهیه کنید. اول سعی کنید چک اصلی را در بانکتان نقد کنید. حالا سعی کنید فتوکپی چک را نقد کنید. می‌توانید تفاوت را در رفتار متصدی بانک مشاهده نمایید. در چک‌های الکترونیکی، نمونه‌ی اصل و کپی قابل تشخیص از یکدیگر نمی‌باشند. ممکن است مدتی طول بکشد تا بانک‌ها نحوه‌ی برخورد با این موضوع را فرا بگیرند.

مردم هویت افراد را با روش‌های مختلفی شناسایی می‌کنند، از جمله تشخیص چهره، صدا، و دستخط. اثبات امضا به وسیله‌ی امضا بر روی برگه‌ی آرم‌دار، مهروموم، و امثال این‌ها انجام می‌شود. دستکاری‌ها معمولاً توسط کارشناسان خط، کارشناسان جوهر، و کارشناسان کاغذ قابل کشف هستند. هیچ‌کدام از این گزینه‌ها به صورت الکترونیکی موجود نیستند. روشن است که به راه‌حل‌های دیگری نیاز داریم.

قبل از آن‌که به سراغ خودِ راه‌حل‌ها برویم، ارزش دارد که مدتی را صرفِ این نکته کنیم که امنیت شبکه در کجای پُشته‌ی پروتکل قرار دارد. احتمالاً یک محل منفرد و جداگانه وجود ندارد. در هر یک از لایه‌ها چیزی در این رابطه وجود دارد. در لایه‌ی فیزیکی با حصارکشیِ خطوط انتقال (یا حتی بهتر از آن، با حصارکشیِ فیبرهای نوری) درون لوله‌های مُهروموم شده که از یک گاز بی‌اثر در فشار بالا پُر شده‌اند، می‌توان از استراق سمع جلوگیری نمود. هرگونه تلاشی جهت ایجاد حفره در لوله سبب آزاد شدنِ مقداری گاز خواهد شد و در نتیجه فشار کاهش یافته و یک علامتِ هشدار صادر می‌شود. بعضی سیستم‌های نظامی از این شیوه استفاده می‌کنند.

در لایه‌ی پیوندِ داده می‌توان بسته‌های روی یک خطِ نقطه ـ به ـ نقطه را در حال ترک ماشین، رمزگذاری کرده و هنگام ورود بسته‌ها به ماشینِ دیگر، آن‌ها را رمزبَرداری کرد. در لایه‌ی پیوند داده، تمام جزئیات را می‌توان کنترل نمود درحالی‌که لایه‌های بالاتر از آنچه رخ می‌دهد باخبر نیستند. اما این راه‌حل هنگامی‌که بسته‌ها ناچار به عبور از مسیریاب‌های متعدد باشند، با شکست مواجه می‌شود زیرا بسته‌ها باید در هر مسیریاب رمزبَرداری شوند و لذا در شرایطی مسیریاب را ترک می‌کنند که در برابرِ حمله‌های درونِ مسیریاب بی‌دفاع بوده‌اند. همچنین این راه‌حل اجازه نمی‌دهد که بعضی نشست‌ها محافظت شوند (مانند نشست‌هایی که مستلزم خرید برخط با استفاده از کارت اعتباری هستند) و بقیه‌ی نشست‌ها محافظت نشوند. معهذا، **رمزگذاری پیوند**[2] (که نام این روش است) می‌تواند به سادگی به هر شبکه‌ای اضافه شود و غالباً نیز سودمند می‌باشد.

در لایه‌ی شبکه دیواره‌های آتش می‌توانند نصب شوند تا بسته‌های خوب را نگه داشته و بسته‌های بد را دور بریزند. در این لایه، امنیتِ IP نیز عملیاتی می‌باشد.

---

1. Locking documents 2. Link encryption

در لایه‌ی حمل کلِ اتصالات می‌توانند به صورت انتها ـ به ـ انتها رمزگذاری شوند، یعنی پردازه ـ به ـ پردازه. برای دستیابی به امنیت حداکثری، به امنیت انتها ـ به ـ انتها نیاز داریم.

و بالاخره مباحثی از قبیل تصدیق هویتِ کاربر و عدم انکار فقط می‌توانند در لایه‌ی کاربرد کنترل شوند.

از آن‌جا که امنیت به طور کامل در یک لایه قرار نمی‌گیرد، بنابراین نمی‌توان آن را فقط در یکی از فصل‌های قبلیِ این کتاب گنجاند. به همین دلیل در فصلِ مربوط به خودش بررسی می‌شود.

هرچند این فصل طولانی، فنی، و بنیادین است ولی در عین حال فعلاً تقریباً بی‌ربط به نظر می‌رسد. این موضوع به اثبات رسیده است که اغلبِ شکست‌ها در بحث امنیت، مثلاً در بانک‌ها، به واسطه‌ی رویه‌های امنیتی‌ای است که کار را آسان می‌گیرند و نیز به دلیل وجود کارکنان نالایق است، به‌علاوه‌ی خطاهای پیاده‌سازیِ فوق‌العاده زیاد که به کاربرانِ غیرمُجاز امکانِ ورودِ بدون اجازه را می‌دهند، و به‌علاوه‌ی حمله‌های مهندسی اجتماعی به طوری که مشتریان اِغفال می‌شوند و جزئیات مربوط به حساب‌هایشان را فاش می‌کنند. رواج این نوع مسائلِ امنیتی بیش از حالتی است که تبهکارانِ باهوش به خطوط تلفن نفوذ کرده و پیغام‌های رمزگذاری شده را کدبَرداری می‌کنند. اگر شخصی بتواند با در دست داشتنِ یک کارت ATM که در خیابان پیدا کرده، به یک شعبه‌ی تصادفی از بانکی برود و ادعا کند که کدِ PIN اش را فراموش کرده و یک کارت جدید بگیرد (با توجه به اصل تکریم ارباب رجوع) تمام رمزنگاری‌های دنیا هم مانع سوءاستفاده نخواهند شد. در تهیه‌ی مطالب این فصل، کتاب راساندرسون[1] واقعاً الهام‌بخش بوده است، چرا که صدها مثال از شکست‌های امنیتی در صنایعِ متعدد متعدد را مستند کرده است. تقریباً تمام این شکست‌ها به واسطه‌ی چیزی است که به طور مؤدبانه می‌توان آن را شیوه‌های کسب و کارِ درهم و برهم یا عدم توجه به امنیت نامید. معهذا، بنیاد و شالوده‌ی فنی‌ای که تجارت الکترونیکی بر آن بنا می‌شود (هنگامی‌که تمام این فاکتورها به خوبی عمل کنند) عبارت است از رمزنگاری.

به جز در مورد امنیتِ لایه‌ی فیزیکی، تقریباً کل امنیت شبکه بر مبنای اصول رمزگونه[2] می‌باشد. به همین دلیل مطالعه‌ی خود درباره‌ی امنیت را با بررسیِ جزئی‌ترِ رمزنگاری آغاز خواهیم کرد. در بخش ۸–۱ بعضی اصول پایه را مشاهده خواهیم کرد. در بخش‌های ۸–۲ تا ۸–۵ تعدادی از الگوریتم‌ها و ساختارهای داده‌ی بنیادین که در رمزنگاری به کار می‌روند را بررسی خواهیم کرد. سپس به سراغ بررسی جزئی‌ترِ نحوه‌ی استفاده از این مفاهیم در دستیابی به امنیت شبکه‌ها خواهیم رفت. خلاصه‌ی بعضی اندیشه‌های مرتبط با فن و اجتماع را نیز خواهیم گنجاند.

---

1. Ross Anderson (2008a)

۲. Cryptographic : در واقع منظور "پنهانی" و "نهفته" است ولی به دلیل حفظ عنصر "Crypt" در معادل ـ گزینی، در برابرِ سایر ترکیبات واژه‌های هم‌خانواده‌اش، در این کتاب از واژه‌ی "رمزگونه" استفاده شده است (مترجم).

قبل از شروع به یک نکته‌ی دیگر نیز اشاره می‌کنیم: چه چیزی پوشش داده نشده است؟ تلاش ما تمرکز بر مباحث شبکه‌بندی بوده است، نه مباحث سیستم عامل و نه مباحث کاربردها، هر چند که مرزبندی میان این مباحث کاری دشوار است. برای مثال، در اینجا هیچ مطلبی درباره‌ی تصدیق هویت با استفاده از علائم زیست‌سنجی (بیومتریک)، امنیتِ رمز عبور (password security)، حمله‌های سرریز شدنِ بافر (buffer overflow attacks)، اسب‌های تروا (Trojan horses)، تقلید برای ورود به سیستم (login spoofing)، تزریق کد (code injection) از قبیل اسکریپت‌نویسیِ همراه با پُل زدن (cross-site scripting)، ویروس‌ها، کرم‌ها، و امثال این‌ها وجود ندارد. تمام این موارد به تفصیل در کتاب *سیستم‌های عامل امروزین* (Tanenbaum، ۲۰۰۷) پوشش داده شده‌اند. خواننده‌ی علاقمند را در ارتباط با جنبه‌های سیستمیِ امنیت، به این کتاب ارجاع می‌دهیم. اکنون بیایید گشت‌وگذارمان را شروع کنیم.

۸–۱ رمزنگاری

رمزنگاری[1] از واژه‌ای یونانی به معنای "نوشتنِ سری" می‌آید. رمزنگاری تاریخچه‌ای طولانی و شنیدنی دارد که به هزاران سال قبل برمی‌گردد. در این بخش به عنوان اطلاعات پیش‌زمینه، تنها به بعضی نکات برجسته‌ی آن اشاره خواهیم کرد. به عنوان مرجعی برای تاریخچه‌ی کامل رمزنگاری، مطالعه‌ی کتاب Kahn (۱۹۹۵) توصیه می‌شود. جهت یک بررسی جامع و فراگیر درباره‌ی الگوریتم‌ها، پروتکل‌ها، و کاربردهای امروزی در زمینه‌ی امنیت و رمزگذاری و مطالب مرتبط با آن، Kaufman و همکاران (۲۰۰۲) را مشاهده نمایید. جهت رویکردی با جنبه‌ی ریاضی‌تر، Stinson (۲۰۰۲) را مطالعه کنید. برای رویکردی که کمتر جنبه‌ی ریاضی داشته باشد، Burnett و Paine (۲۰۰۱) توصیه می‌شود.

متخصصان بین رمزها و کدها تمایز قائل می‌شوند. یک **رمز**[2] عبارت‌است‌از تبدیل کاراکتر ـ به ازای ـ کاراکتر یا بیت ـ به ازای ـ بیت، بدون توجه به ساختار پیغام از جنبه‌ی زبان‌شناختی. از طرف دیگر، یک **کد**[3]، واژه‌ای را با واژه یا نمادی دیگر جایگزین می‌کند. این روزها از کدها استفاده نمی‌شود هر چند که گذشته‌ی باشکوهی دارند. موفق‌ترین کدی که تا به حال ابداع شده، در نیروهای مسلح ایالات متحده در حین جنگ دوم جهانی و در منطقه‌ی پاسیفیک به کار می‌رفت. آن‌ها خیلی ساده، با هم به زبان سرخپوستان ناواهو[4] صحبت کرده و برای اصطلاحات ارتشی از واژه‌های مخصوص ناواهو استفاده می‌کردند، مانند استفاده از *chay-dagahi-nail-tsaidi* (به صورت تحت‌اللفظی یعنی "قاتل لاک‌پشت") به جای "جنگ‌افزار ضدتانک". زبان ناواهو بسیار آهنگین و به طرز فوق‌العاده‌ای پیچیده است و هیچ شکل نوشتاری‌ای ندارد. و ضمناً هیچ کس در ژاپن چیزی درباره‌ی آن نمی‌داند.

در سپتامبر ۱۹۴۵ نشریه‌ی *San Diego Union* با گفتن جمله‌ای این کد را توضیح داد: "به مدت سه سال، هر جا که کشتی مارینز[5] پهلو می‌گرفت، ژاپنی‌ها صداهای عجیب و درهمی را دریافت

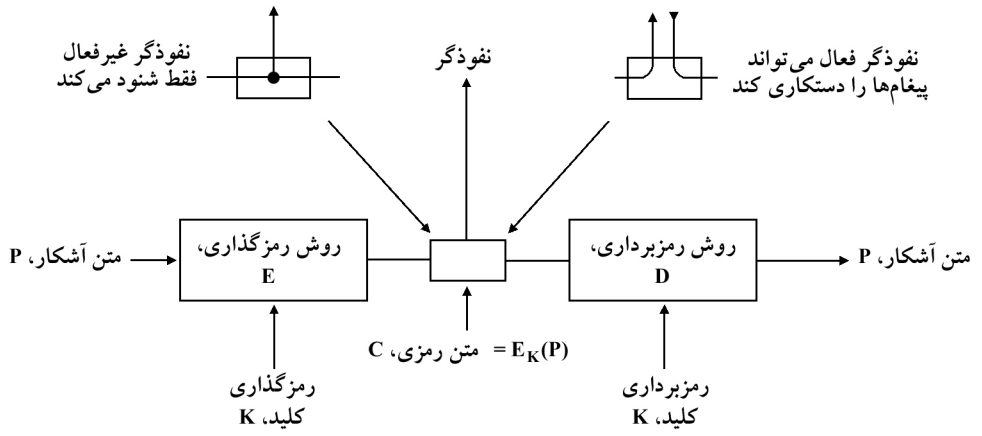
1. Cryptography 2. Cipher 3. Code 4. Navajo Indians 5. Marines

می‌کردند که با اصوات دیگر پخش می‌شد و به آوای راهبان تبتی و صدای خالی شدنِ یک بطری آب داغ شباهت داشت. " ژاپنی‌ها هرگز این کد را نشکستند و بسیاری از افرادی که با کد ناواهویی صحبت می‌کردند به پاس خدمات برجسته و دلاورانه‌شان، به افتخارات بالای نظامی نائل شدند. این حقیقت که ایالات متحده کد ژاپنی‌ها را شکست ولی ژاپنی‌ها هرگز کد ناواهو را نشکستند، نقش حیاتی در پیروزی‌های امریکا در منطقه‌ی پاسیفیک داشت.

۸-۱-۱ مقدمه‌ای بر رمزنگاری

به لحاظ تاریخی چهار گروه از مردم از رمزنگاری استفاده کرده و بر روی هنر رمزنگاری تأثیر داشته‌اند: ارتش، سیاستمداران (diplomatic corps)، وقایع‌نگاران (diarists)، و عشاق (lovers). از این میان، ارتش مهم‌ترین نقش را داشته و طی قرن‌ها این فیلد را شکل داده است. در سازمان‌های ارتشی به طور سنتی، پیغام‌هایی که باید رمزگذاری شوند را جهت رمزگذاری و انتقال، به کارکنان دفترِی ضعیف و سطحِ پایین واگذار می‌کنند. حجم زیاد پیغام‌ها مانع از این می‌شود که این کار بتواند توسط متخصصان زبده انجام گیرد.

تا پیش از ظهور کامپیوتر یکی از محدودیت‌های اصلی در رمزنگاری عبارت بود از توانایی کارمندِ کدگذار در اجرای تبدیلاتِ ضروری که غالباً نیز در میدان جنگ و با تجهیزاتِ محدود انجام می‌شد. محدودیتِ دیگر عبارت بوده است از دشواری در سوئیچ کردنِ سریع از یک روش رمزنگاری به یک روش رمزنگاری دیگر، زیرا این کار مستلزم آموزش تعداد زیادی از افراد می‌باشد. اما چون خطر دستگیریِ یک کارمند کدگذار از سوی دشمن وجود دارد، لذا توانایی در تغییر آنیِ روش رمزنگاری در شرایط اضطراری، از اهمیت زیادی برخوردار است. وجود این تضاد در ضرورت‌ها باعث ایجاد مدل شکل ۸-۲ شده است.



شکل ۸-۲ مدل رمزگذاری (برای یک رمز کلید ـ متقارن).

پیغام‌هایی که باید رمزگذاری شوند، که به آن‌ها **متن آشکار**[1] گفته می‌شود، توسط یک تابع تبدیل می‌شوند که پارامتر این تابع یک **کلید**[2] است. سپس خروجی پروسه‌ی کدگذاری که نامش **متن رمزی**[3] است، منتقل می‌شود (غالباً توسط تلگراف یا رادیو). فرض می‌کنیم که دشمن یا **نفوذگر**[4]، کل متن رمزی را می‌شنود و به دقت آن را کپی می‌کند. اما او برخلاف گیرنده‌ای که مقصدِ پیغام است، کلید رمزبَرداری را نمی‌داند و بنابراین به سادگی نمی‌تواند متن رمزی را رمزبَرداری کند. بعضی اوقات نفوذگر نه تنها می‌تواند کانال ارتباطی را شنود کند (نفوذگرِ غیرفعال[5]) بلکه می‌تواند پیغام‌ها را ضبط کرده و آن‌ها را در زمان دیرتری بازنواخت نماید درحالی‌که قبل از آن‌که پیغام‌ها به گیرنده برسند، پیغام‌های خودش را در آن‌ها تزریق کرده یا پیغام‌های درست را تغییر داده است (نفوذگرِ فعال[6]). هنر شکستنِ رمزها، یا همان **رمزیابی**[7]، و هنر ابداعِ آن‌ها (رمزنگاری) در مجموع با عنوان **رمزشناسی**[8] شناخته می‌شود.

داشتنِ یک شیوه‌ی نشانه‌گذاری برای مرتبط کردنِ متن آشکار، متن رمزی، و کلید در اغلب مواقع سودمند خواهد بود. برای نشان دادنِ این‌که رمزگذاریِ متن آشکار P با استفاده از کلید K، متن رمزی C را خواهد داد، از رابطه‌ی $C = E_K(P)$ استفاده خواهیم کرد. همین‌طور رابطه‌ی $P = D_K(C)$ نشان دهنده‌ی رمزبَرداری از C برای رسیدنِ مجدد به متن آشکار است. بنابراین نتیجه می‌گیریم که:

$$D_K(E_K(P)) = P$$

این نشانه‌گذاری تلویحاً حکایت از آن دارد که E و D دقیقاً تابع ریاضی هستند، که البته همین‌طور هم هست. تنها قسمتِ استادانه‌ی موضوع آن است که هر دوی آن‌ها تابعی از دو پارامترند، که ما یکی از پارامترها (یعنی کلید را) به جای آرگومان، به صورت اندیس نوشته‌ایم تا آن را از پیغام، تمایز داده باشیم.

قاعده‌ی اصلی در رمزنگاری آن است که باید فرض کنیم رمزیاب[9] روش‌های به کار رفته برای رمزگذاری و رمزبَرداری را می‌داند. به عبارت دیگر، رمزیاب جزئیاتِ نحوه‌ی کارِ روش رمزگذاری، یعنی E، و روش رمزبَرداری، یعنی D در شکل ۸-۲ را می‌داند. هر بار که روش قدیمی رمزنگاری در معرض مخاطره قرار می‌گیرد (یا تصور می‌شود که در معرض مخاطره قرار گرفته است)، حجم تلاشی که برای ابداع، آزمایش، و نصب یک الگوریتم جدید لازم است، سبب شده که همواره سرّی نگهداشتنِ الگوریتم رمزگذاری غیرممکن گردد. تصورِ سرّی بودنِ الگوریتم، آن هم هنگامی‌که سرّی نیست، آسیبش بیشتر از خوبی‌اش است.

این‌جاست که مفهوم کلید وارد ماجرا می‌شود. کلید از یک رشته‌ی (نسبتاً) کوتاه تشکیل می‌شود که یکی از چندین رمزگذاری‌های بالقوه را برمی‌گزیند. بر خلافِ روش کلی (general method)، که احتمال دارد هر چند سال یک بار تغییر کند، کلید می‌تواند هر چند بار که لازم باشد تغییر کند. لذا

1. Plaintext 2. Key 3. Ciphertext 4. Intruder 5. Passive intruder 6. Active intruder
7. Cryptanalysis 8. Cryptology 9. Cryptanalyst

مدل پایه‌ی ما ثابت و پایدار است و عموماً با نام روش کلی (یا general method) شناخته می‌شود و توسط یک کلید سرّی که به آسانی تغییر می‌کند، پارامتربندی می‌گردد. این ایده که رمزیاب، الگوریتم‌ها را می‌داند و رازپوشی منحصراً در کلیدها نهفته است، **اصل کیرشهف**[1] نامیده می‌شود. این نام‌گذاری پس از آن بود که آگوست کیرشهف[2] (رمزنگار فلاندری زبانِ ارتش بلژیک) برای اولین بار در سال ۱۸۸۳ آن را مطرح کرد (Kerckhoff، ۱۸۸۳). بر این اساس

اصل کیرشهف: تمام الگوریتم‌ها باید عمومی (public) باشند، فقط کلیدها سرّی هستند.

هر اندازه بر سرّی نبودنِ الگوریتم تأکید شود باز هم کم است. تلاش برای سرّی ماندنِ الگوریتم، که در دنیای تجارت با عنوان **امنیت از طریق گمنام ماندن**[3] شناخته می‌شود، هرگز دردی را دوا نمی‌کند. علاوه بر این، رمزنگار از طریق عمومی‌سازیِ الگوریتم می‌تواند از مشاوره‌ی رایگانِ عده‌ی زیادی از رمزشناسانِ دانشگاهی نیز بهره ببرَد: یعنی افرادی که علاقمند به شکستنِ سیستم هستند تا بتوانند مقاله‌هایی منتشر کنند و به همه نشان دهند که چقدر باهوشند. اگر بعد از گذشتِ مدتی طولانی از انتشار الگوریتم، متخصصان زیادی تلاش کردند تا آن را بشکنند و موفق نشدند، احتمالاً آن الگوریتم به طرز قابل ملاحظه‌ای مقاوم و ایمن است.

از آن‌جا که رازپوشیِ واقعی در کلید است، طول کلید از موارد مهم در طراحی می‌باشد. یک قفل ترکیبیِ ساده را در نظر بگیرید. قاعده‌ی کلی به این صورت است که ارقامی را به ترتیب وارد کنید. البته هر کسی این را می‌داند اما کلید، سرّی است. داشتنِ کلیدی به طول ۲ رقم، به این معنی است که ۱۰۰ احتمال مختلف وجود دارند. یک کلید به طول ۳ یعنی ۱۰۰۰ احتمال، و کلیدی به طول ۶ رقم، یعنی یک میلیون. هر چه کلید طولانی‌تر باشد، **work factor** ای[4] هم که رمزیاب با آن درگیر می‌شود، بیشتر خواهد بود. میزان work factor برای شکستنِ سیستم از طریق جستجوی جامعِ فضای کلید، رابطه‌ی نمایی با طول کلید دارد. رازپوشی (secrecy) از طریق داشتنِ یک الگوریتم قوی (اما عمومی) و یک کلید طولانی حاصل می‌شود. برای آن‌که مانعِ برادر کوچک‌ترتان از خواندنِ ایمیل‌تان شوید، کلیدهای ۶۴ـ بیتی به درد می‌خورند. برای کاربردهای تجاریِ رایج، دست کم از ۱۲۸ بیت باید استفاده شود. برای محافظت از اسناد مهم حکومتی، کلیدهایی با دستِ کم ۲۵۶ بیت، و ترجیحاً بیشتر، لازم هستند.

از دید رمزیاب، مسئله‌ی رمزیابی سه نوع اصلی دارد. هنگامی‌که رمزیاب حجم عمده‌ای از متن رمزی دارد و هیچ متن آشکاری ندارد، با مسئله‌ی **فقط ـ متن رمزی**[5] مواجه است. رمزنگاشت‌هایی که در بخش پازل روزنامه‌ها دیده می‌شوند، مثالی از این نوع مسئله هستند. زمانی که رمزیاب متن‌های

---

1. Kerckhoff's principle 2. Auguste Kerckhoff 3. Security by obscurity

۴. Work factor : (یا "ضریب عملکرد") عبارت است از میزان تلاشی که لازم است تا یک سیستم رمزنگاری شده شکسته شود. معمولاً بر حسب واحد زمان اندازه‌گیری می‌شود (مترجم).

5. Ciphertext-only

رمزی و آشکارِ همسان دارد، این مسئله به نام مسئله‌ی **متن آشکارِ شناخته شده**[1] نامیده می‌شود. و بالاخره، هنگامی‌که رمزیاب توانایی رمزگذاریِ تکه‌هایی از متن آشکار را بر اساس انتخاب خودش دارد، با مسئله‌ی **متن آشکارِ انتخاب شده**[2] مواجهیم. رمزنگاشت‌های روزنامه‌ها می‌توانند به عنوان عملی پیش پا افتاده شکسته شوند، البته درصورتی‌که رمزیاب مُجاز به طرح پرسش‌هایی از این قبیل باشد که "رمزنگاریِ مربوط به ABCDEFGHIJKL چیست؟"

افراد نوآموز در حرفه‌ی رمزنگاری غالباً چنین می‌پندارند که اگر یک رمز بتواند در برابر یک حمله از نوعِ فقط ـ متن رمزی دوام بیاورَد، یعنی آن رمز امن است. این پندار نشانه‌ی بی‌تجربگی است. در بسیاری از موارد، رمزیاب می‌تواند حدس خوبی در قسمت‌هایی از متنِ آشکار داشته باشد. به طور مثال، اولین چیزی که در اغلب کامپیوترها به چشم می‌خورَد درخواستِ "login:" است. اگر رمزیاب به جفت‌های دوتاییِ "متن آشکار ـ متن رمزی" مجهز باشد، کارش بسیار آسان‌تر می‌شود. برای رسیدن به امنیت، رمزنگار باید محتاط باشد و اطمینان حاصل کند که سیستم قابل شکستن نیست، حتی اگر حریف بتواند هر مقدارِ دلخواهی از متن آشکارِ مورد نظر را رمزگذاری کند.

روش‌های رمزگذاری به لحاظ تاریخچه‌ای به دو رده تقسیم شده‌اند: رمزهای جایگزین‌سازی[3] و رمزهای جابجاسازی[4]. در این‌جا به عنوان اطلاعات پیش‌زمینه برای رمزنگاریِ مدرن، هر یک را به اختصار بررسی خواهیم کرد.

۸-۱-۲ رمزهای مبتنی بر جایگزین‌سازی

در یک **رمزِ جایگزین‌سازی**، هر حرف یا گروهی از حروف، به وسیله‌ی حرف دیگری و یا گروه دیگری از حروف جایگزین می‌شود تا ظاهر آن را تغییر دهد. یکی از قدیمی‌ترین رمزهای شناخته شده **رمز سزار**[5] است که منسوب به جولیوس سزار می‌باشد. با این روش، a به D، b به E، c به F، ... و z به C تبدیل می‌شود. به طور مثال *attack* تبدیل به *DWWDFN* می‌شود. در مثال‌های ما، متن آشکار با حروف کوچک داده می‌شود و متن رمزی با حروف بزرگ.

یک تعمیم کوچک و جزئی از رمز سزار، این امکان را می‌دهد که الفبای متن رمزی، به جای سه حرف به اندازه‌ی k حرف جابجا شود. بنابراین در حالتِ کلی، k به منزله‌ی یک کلید برای جابه‌جاییِ حروف به صورت چرخشی است. رمز سزار ممکن است پومپی[6] را فریب داده باشد، اما از آن زمان تا به حال فرد دیگری را نتوانسته فریب بدهد!

---

1. Known plaintext 2. Chosen plaintext

۳. Substitution cipher یا "رمز براساس عوض و بدل‌سازی"

۴. Transposition cipher یا "رمز بر اساس پس و پیش‌گذاری"

5. Caesar cipher

۶. Pompey : سرداری که رقیب جولیوس سزار محسوب می‌شد (مترجم).

بهسازیِ بعدی عبارت است از این‌که هریک از نمادها در متن آشکار (مثلاً برای سادگی همان ۲۶ حرف الفبا را در نظر بگیرید) را وادار کنیم به حرف دیگری نگاشت شوند. برای مثال،

متن آشکار: a b c d e f g h i j k l m n o p q r s t u v w x y z
متن رمزی: Q W E R T Y U I O P A S D F G H J K L Z X C V B N M

سیستم کلی برای جایگزین‌سازیِ نماد ـ به ـ نماد، **رمز جایگزین‌سازیِ تک ـ الفبایی**[1] نامیده می‌شود، به طوری که کلیدِ آن یک رشته‌ی ۲۶ـ حرفی متناظر با الفباست. برای کلیدی که داده شد، متن آشکار *attack* به متن رمزی *QZZQEA* تبدیل خواهد شد.

این سیستم ممکن است در نگاه اول به نظر ایمن برسد، زیرا هر چند رمزیاب، از سیستم کلی آگاهی دارد (یعنی همان جایگزینی حرف ـ به ـ حرف) ولی نمی‌داند از کدام یک از $26! \approx 4 \times 10^{26}$ کلیدِ ممکن، استفاده شده است. در برابر رمز سزار، امتحان کردنِ تمام این کلیدها، رویکرد امیدوار کننده‌ای نیست. حتی اگر ۱ نانوثانیه برای هر کلید طول بکشد و یک میلیون تراشه‌ی کامپیوتری به طور موازی کار کنند، باز هم امتحان کردنِ تمام کلیدها ۱۰ هزار سال به درازا خواهد کشید.

معهذا با داشتنِ یک متن رمزی کوچک، رمز می‌تواند به آسانی شکسته شود. حمله‌ی اصلی (basic attack) از ویژگی‌های آماریِ زبان‌های طبیعی سود می‌برَد. برای مثال در زبان انگلیسی حرف *e* بیشترین استفاده را دارد، پس از آن حروف *t* ، *o* ، *a* ، *n* و *i* قرار دارند و همینطور تا آخر برای بقیه‌ی حروف. رایج‌ترین ترکیباتِ دو ـ حرفی، یا همان **digram ها**[2]، عبارتند از *th* ، *in* ، *er* ، *re* ، و *an* . رایج‌ترین ترکیبات سه ـ حرفی، یا **trigram ها**، عبارتند از *the* ، *ing* ، *and* ، و *ion* .

رمزیابی که سعی دارد یک رمز الفبایی را بشکند، با شمارش فرکانس استفاده ازتمام حروف در متن رمزی، کارش را شروع می‌کند. در وهله‌ی بعد ممکن است بیشترین موردِ تکرار را به حرف *e* و بیشترین تکرارِ بعدی را به حرف *t* انتساب دهد. حالا به trigram ها مراجعه می‌کند تا یک موردِ رایج که به شکل *tXe* باشد را پیدا کند. آنچه قویاً پیشنهاد می‌شود، حرف *h* است. به همین ترتیب، اگر الگوی *thYt* به دفعات تکرار شود، احتمالاً *Y* نشان دهنده‌ی حرف *a* است. با این اطلاعات، رمزیاب می‌تواند در جستجوی تعداد دفعات trigram ای به شکل *aZW* باشد، که به احتمال زیاد، ترکیب *and* خواهد بود. رمزیاب با حدس‌های مربوط به حروف، digramها، و trigram ها و نیز با داشتنِ اطلاعاتی درباره‌ی الگوهای احتمالیِ حروف صدادار و حروف بی‌صدا، متن آشکارِ احتمالی را حرف به حرف می‌سازد.

رویکرد دیگر عبارت است از حدس زدن درباره‌ی یک واژه یا عبارتِ احتمالی. برای مثال، متن رمزی زیر از یک موسسه‌ی حسابداری را در نظر بگیرید (که در قالب گروه‌های ۵ کاراکتری قرار داده شده است):

---

1. Monoalphabetic substitution cipher

۲. Digram ها یا Digraph ها عبارتند از ترکیبات دو ـ حرفی (نوع سه ـ حرفیِ آن هم هست که trigram یا trigraph نام دارد) که روی هم یک آوا تولید می‌کنند و به دفعات زیاد در یک زبان به کار می‌روند (مترجم).

CTBMN BYCTC BTJDS QXBNS GSTJC BTSWX CTQTZ CQVUJ
QJSGS TJQZZ MNQJS VLNSX VSZJU JDSTS JQUUS JUBXJ
DSKSU JSNTK BGAQJ ZBGYQ TLCTZ BNYBN QJSW

مثلاً واژه‌ی *financial* یکی از واژه‌های محتمل در پیغامِ مربوط به این موسسه‌ی حسابداری می‌باشد. با دانستنِ این موضوع که واژه‌ی *financial* دارای یک حرف تکراری (یعنی حرف *i*) است، و این‌که چهار حرف مابین دو حرف *i* وجود دارند، در این محدوده به دنبال حروف تکراری در متن رمزی جستجو می‌کنیم. دوازده مورد در موقعیت‌های ۶ ، ۱۵، ۲۷، ۳۱، ۴۲، ۴۸، ۵۶، ۶۶، ۷۰، ۷۱، ۷۶، و ۸۲ پیدا می‌کنیم. اما فقط دو مورد از این دوازده مورد هستند که حرف بعدیِ آن‌ها با متن آشکارمان متناظر است (یعنی حرف *n* بعد از *i* دارند) که عبارتند از موقعیت‌های ۳۱ و ۴۲. از این دو مورد نیز فقط ۳۱ است که یک موقعیت صحیح دارد، بنابراین *financial* از موقعیت ۳۰ شروع می‌شود. از همین موضوع، با استفاده از آمارهای مربوط به فرکانسِ وقوع حروف در متون انگلیسی، و نیز جستجو به دنبال واژه‌های تقریباً کامل، استنتاج کردن (تا رسیدن به واژه‌های تقریباً کامل) آسان می‌شود.

۸-۱-۳ رمزهای مبتنی بر جابجاسازی

رمزهای جایگزین‌سازی ترتیب نمادها در متن آشکار را حفظ می‌کنند اما آن‌ها را تغییر می‌دهند. برعکس، رمزهای جابجاسازی حروف را مجدداً مرتب می‌کنند اما آن‌ها را تغییر نمی‌دهند. شکل ۸-۳ جابجاسازی ستونی، که یک رمز جابجاسازی رایج است را نشان می‌دهد. در این مثال، کلید عبارت است از MEGABUCK . هدف از این کلید، مرتب‌سازی ستون‌هاست به طوری که ستون ۱ زیر نزدیک‌ترین حرف کلیدی در سمت آغاز حروف باشد، و همین‌طور تا آخرِ کلید. متن آشکار به صورت افقی (یعنی سطری) نوشته می‌شود و در صورت لزوم، حروفی به آن اضافه می‌شوند تا ماتریس پُر شود. متن رمزی به ترتیب ستونی خوانده می‌شود و کار از ستونی آغاز می‌شود که حرف کلیدیِ آن کوچک‌تر است.

M	E	G	A	B	U	C	K
7	4	5	1	2	8	3	6
p	l	e	a	s	e	t	r
a	n	s	f	e	r	o	n
e	m	i	l	l	i	o	n
d	o	l	l	a	r	s	t
o	m	y	s	w	i	s	s
b	a	n	k	a	c	c	o
u	n	t	s	i	x	t	w
o	t	w	o	a	b	c	d

متن آشکار

pleasetransferonemilliondollarsto
myswissbankaccountsixtwotwo

متن رمزی

AFLLSKSOSELAWAIATOOSSCTCLNMOMANT
ESILYNTWRNNTSOWDPAEDOBUOERIRICXB

شکل ۸-۳ یک رمز مبتنی بر جابجاسازی.

برای شکستنِ یک رمز جابجاسازی، رمزیاب ابتدا بایستی بداند که با یک رمز جابجاسازی مواجه است. با نگاه به دفعات تکرار E ، T ، A ، O ، I ، N ، و غیره به آسانی می‌توان مشاهده کرد آیا با الگوی عادیِ متن آشکار تطبیق دارند یا خیر. در صورتی که مطابقت داشته باشند، روشن است که رمز از نوع رمز جابجاسازی می‌باشد زیرا در این‌گونه رمزها، هر حرف نشان دهنده‌ی خودش است و بنابراین توزیعِ تعداد دفعات وقوعِ آن حرف تغییر نمی‌کند.

مرحله‌ی بعدی عبارت‌است‌از حدس زدنِ تعداد ستون‌ها. در بسیاری از موارد، یک واژه یا یک عبارتِ ممکن، از محتوا قابل حدس زدن می‌باشد. به طور مثال فرض کنید که رمزیابِ ما احتمال می‌دهد که عبارت آشکارِ *milliondollars* جایی در پیغام آمده باشد. توجه داشته باشید که digram های *MO* ، *IL* ، *LL* ، *LA* ، *IR* ، و *OS* در متن رمزی، در نتیجه‌ی اجرای این پروسه، یافت می‌شوند. حرف *O* در متن رمزی، به دنبال حرف *M* در متن رمزی قرار می‌گیرد (به عبارت دیگر این دو حرف در ستون ۴ دارای مجاورتِ عمودی هستند) زیرا این دو حرف در عبارتِ احتمالی، به اندازه‌ی فاصله‌ای برابر با طول کلید از یکدیگر جدا شده‌اند. اگر از کلیدی به طول ۷ استفاده شده باشد، به جای digramهای قبلی، digramهای *MD* ، *IO* ، *LL* ، *LL* ، *IA* ، *OR* ، و *NS* وجود خواهند داشت. در حقیقت به ازای هر طول کلید، مجموعه‌ی متفاوتی از digram ها در متن رمزی تولید خواهند شد. با دنبال کردن انواعِ احتمالات، رمزیاب غالباً می‌تواند به آسانی طول کلید را تعیین کند.

مرحله‌ای که باقی می‌مانَد، رتبه‌بندیِ ستون‌هاست. اگر تعداد ستون‌ها، یعنی k، کم باشد، هر کدام از $k(k-1)$ جفت ـ ستون می‌توانند به نوبت بررسی شوند تا ببینیم آیا فرکانس‌های digram آن‌ها با فرکانس مربوط به متن آشکارِ انگلیسی مطابقت دارد یا خیر. جفت ـ ستونی که بیشترین مطابقت را داشته باشد، فرض می‌شود در موقعیت صحیح قرار دارد. اکنون هر یک از ستون‌های باقیمانده به صورت آزمایشی، به عنوان بعدیِ این جفت ـ ستون، بررسی می‌شوند. ستونی که تعداد فرکانس‌های digram و trigram آن دارای بهترین مطابقت باشد، موقتاً به عنوان ستونِ صحیح فرض می‌شود. ستون بعدی هم به همین روش پیدا می‌شود. کل این پروسه تا پیدا شدنِ یک ترتیب‌بندیِ بالقوه ادامه می‌یابد. اگر شانس با ما یار باشد، در این مرحله متن آشکار قابل تشخیص خواهد بود (به این معنی که اگر واژه‌ی *million* وجود داشته باشد، معلوم می‌شود که خطا در کجا بوده است).

بعضی از رمزهای جابجاسازی، یک بلوک ورودی با طول ثابت را پذیرفته و یک بلوک خروجی با طول ثابت را تولید می‌کنند. این رمزها می‌توانند به طور کامل با دادنِ یک فهرست توصیف شوند. این فهرست، ترتیبی که کاراکترها بایستی با آن ترتیب خارج شوند را می‌گوید. مثلاً رمز شکل ۸–۳ می‌تواند به صورت یک بلوک رمز ۶۴ـ کاراکتری دیده شود. خروجی آن عبارت است از ۴، ۱۲، ۲۰، ۲۸، ۳۶، ۴۴، ۵۲، ۶۰، ۵، ۱۳، ... ، ۶۲ . به عبارت دیگر، چهارمین کاراکتر ورودی، یعنی a، اولین کاراکتر خروجی می‌باشد، به دنبالش دوازدهمین کاراکتر، یعنی f، و همین‌طور ادامه می‌یابد.

```
پیغام ۱ :    1001001 0100000 1101100 1101111 1110110 1100101 0100000 1111001 1101111 1110101 0101110
پد ۱ :       1010010 1001011 1110010 1010101 1010010 1100011 0001011 0101010 1010111 1100110 0101011
متن رمزی:    0011011 1101011 0011110 0111010 0100100 0000110 0101011 1010011 0111000 0010011 0000101

پد ۲ :       1011110 0000111 1101000 1010011 1010111 0100110 1000111 0111010 1001110 1110110 1110110
متن آشکار۲ : 1000101 1101100 1110110 1101001 1110011 0100000 1101100 1101001 1110110 1100101 1110011
```

شکل ۸-۴ استفاده از پَد یک بار- مصرف جهت رمزگذاری و امکانِ رسیدن به هر متن آشکارِ محتمل از متن رمزی (با استفاده از پَدهای دیگر).

۸-۱-۴ پَدهای یک بار ـ مصرف

ساختنِ یک رمزِ غیرقابل شکستن کار واقعاً آسانی است؛ روش این کار چندین دهه است که شناخته شده است. ابتدا یک رشته‌ی بیت را به طور تصادفی به عنوان کلید انتخاب کنید. سپس متن آشکار را به یک رشته‌ی بیت تبدیل نمایید، مثلاً با استفاده از نمایشِ ASCII آن. در انتها، XOR این دو رشته‌ی بیت را محاسبه کنید. متن رمزیِ حاصل نمی‌تواند شکسته شود زیرا در یک نمونه‌ی متن رمزی که به اندازه‌ی کافی بزرگ باشد، میزانِ فراوانیِ هر یک از حروف مساوی خواهد بود. این موضوع برای هر کدام از digram ها، هر کدام از trigram ها، و بقیه‌ی موارد نیز برقرار می‌باشد. این روش که با نام **پَدِ یک بار ـ مصرف**[1] شناخته می‌شود، صرفنظر از این‌که نفوذگر چه میزان توان محاسباتی داشته باشد، در برابر تمام حمله‌های حال حاضر و آتی، ایمن است. دلیل این موضوع از نظریه‌ی اطلاعات ناشی می‌شود: هیچ اطلاعاتی به صورت صریح در پیغام وجود ندارد زیرا تمام متن‌های آشکارِ محتمل با طولِ داده شده، احتمالاً معادل هم هستند.

مثالی از نحوه‌ی استفاده از پَدهای یک بار- مصرف در شکل ۸-۴ داده شده است. ابتدا پیغام ۱ (یعنی ".I love you") به ASCII هفت ـ بیتی تبدیل می‌شود. سپس یک پَد یک بار- مصرف، یعنی پَد ۱، انتخاب شده و با پیغام XOR می‌شود تا متن رمزی به دست آید. رمزیاب می‌تواند تمام پَدهای یک بار ـ مصرف را امتحان کند تا ببیند به ازای هر کدام چه متن آشکاری حاصل می‌شود، تا به متن آشکار ۲ برسد ("Elvis lives"). این متن آشکار ممکن است به نظر موجه برسد یا به نظر موجه نرسد (این بحث خارج از چارچوب این کتاب است). در حقیقت، به ازای هر متن آشکارِ ASCII یازده ـ کاراکتری، یک پَد یک بار- مصرف وجود دارد که آن را تولید می‌کند. این همان موضوعی است که با گفتنِ "هیچ اطلاعاتی در متن رمزی نیست" مورد نظرمان است: می‌توانید به هر پیغامی برسید که طولش صحیح باشد.

پَدهای یک بار- مصرف به لحاظ تئوری عالی هستند اما در عمل معایبی دارند. برای شروع، کلید را نمی‌توان از بَر کرد بنابراین هم ارسال‌کننده و هم دریافت‌کننده باید یک کپیِ نوشته شده از کلید را با خود حمل کنند. از آن‌جا که احتمال دارد هر یک از ارسال‌کننده یا دریافت‌کننده دستگیر شوند، لذا

1. One-time pad

واضح است که کلیدِ نوشته شده مناسب نیست. علاوه بر این، حجم کلِ داده‌ای که می‌تواند منتقل گردد، محدود است به حجم کلیدِ در دسترس. اگر جاسوسی به اطلاعات دسترسی پیدا کند و حجم قابل توجهی داده را کشف کند، ممکن است به دلیل آن‌که کلید کاملاً مصرف شده است، قادر به انتقال اطلاعات به ستاد فرماندهی‌اش نباشد. مسئله‌ی دیگر عبارت‌است‌از حساسیت این روش نسبت به کاراکترهای گُم شده یا کاراکترهای درج شده. اگر ارسال‌کننده و دریافت‌کننده از حالتِ همگامی نسبت به یکدیگر خارج شوند، تمام داده‌ای که از آن به بعد می‌رسد، نامفهوم خواهد بود.

با ظهور کامپیوترها، این امکان وجود داشت که پَد یک‌بار_ مصرف برای بعضی کاربردها، شکل عملیاتی به خود بگیرد. مبدأ کلید، می‌توانست یک DVD مخصوص با چندین گیگابایت اطلاعات باشد و درصورتی که درون یک جعبه‌ی فیلم DVD حمل می‌شد و چند دقیقه فیلم نیز در ابتدای آن قرار می‌گرفت، حتی به آن مشکوک هم نمی‌شدند. البته در سرعت‌های شبکه‌های گیگابیت، اجبار در درجِ یک DVD جدید در هر ۳۰ ثانیه، می‌تواند سبب شود کار به درازا کشیده شود. ضمناً قبل از آن‌که هرگونه پیغامی بتواند ارسال گردد، بایستی DVDها به صورت دستی از ارسال‌کننده به دریافت‌کننده حمل شوند. این موضوع نیز کارایی واقعی آن‌ها را کاهش می‌دهد.

رمزنگاری کوانتومی

موضوع جالب این‌جاست که ممکن است برای نحوه‌ی انتقال پَد یک‌بار_ مصرف از طریق شبکه، راه‌حلی وجود داشته باشد. این راه‌حل از مبدأیی بسیار نامحتمل سرچشمه می‌گیرد: مکانیک کوانتوم. این حیطه هنوز هم جنبه‌ی آزمایشی دارد، اما آزمایش‌های اولیه دلگرم‌کننده هستند. اگر این روش بتواند تکمیل شده و به صورتی کارآمد ساخته شود، نهایتاً تمام رمزنگاری‌ها به صورت مَجازی با استفاده از پَدهای یک‌بار_ مصرف انجام خواهند شد زیرا امن بودنِ آن‌ها را می‌توان اثبات کرد. در ادامه به اختصار نحوه‌ی کارِ این روش، یعنی **رمزنگاری کوانتومی**[1] را شرح داده و مشخصاًپروتکلی به نام **BB84** را بررسی خواهیم کرد (Bennet و Brassard، ۱۹۸۴).

فرض کنید یک کاربر به نام آلیس (Alice) می‌خواهد یک پَد یک بار_ مصرف را با یک کاربر دیگر به نام باب (Bob) برقرار نماید. آلیس و باب که شخصیت‌های اصلی داستان ما هستند، **بازیگران اصلی**[2] نامیده می‌شوند. به عنوان مثال، باب یک بانکدار است و آلیس مایل است با او معامله کند. از زمانی که رون ریوست[3] سال‌ها قبل "آلیس" و "باب" را معرفی کرد، این دو نام در تمام مقاله‌ها و کتاب‌های مرتبط با رمزنگاری، به عنوان بازیگران اصلی استفاده می‌شوند (Rivest و همکاران، ۱۹۷۸). رمزنگاران به سنت علاقه دارند. اگر "اندی" و "باربارا" را به عنوان بازیگران اصلی استفاده می‌کردیم، کسی به مطالب این فصل اعتماد نمی‌کرد. لذا به شیوه‌ی مرسوم عمل کردیم.

---

1. Quantum cryptography 2. Principals (موجودیت‌های اصلی) 3. Ron Rivest

اگر آلیس و باب بتوانند یک پَد یک بارـ مصرف را برقرار کنند، می‌توانند به صورتی امن از آن برای ارتباط استفاده کنند. پرسش این است: آن‌ها چطور می‌توانند بدون آن‌که قبلاً DVD ها را تبادل کرده باشند، پَد یک بارـ مصرف را برقرار کنند؟ می‌توانیم فرض کنیم که آلیس و باب در دو انتهای یک فیبر نوری قرار دارند و از طریق این فیبر می‌توانند پالس‌های نوری را ارسال و دریافت کنند. اما یک نفوذگرِ جسور به نام ترودی (Trudy) می‌تواند فیبر را بریده و یک اتصال برای شنودِ مخفیانه به آن اضافه کند. ترودی قادر است تمام بیت‌های ارسالی در هر دو جهت را بخواند. او همچنین قادر است پیغام‌های دروغینی در هر دو جهت ارسال نماید. ممکن است به نظر برسد که این وضعیت برای آلیس و باب درمان‌ناپذیر است، اما رمزنگاری کوانتومی می‌تواند نور جدیدی به این مسئله بتابانَد.

رمزنگاری کوانتومی بر پایه‌ی این حقیقت استوار شده است که نور به شکل بسته‌های کوچکی به نام **فوتون** است که ویژگی‌های منحصربه‌فردی دارند. ضمناً با عبور دادنِ نور از یک فیلتر پولاریزاسیون، می‌توان آن را پولاریزه نمود (که این امر برای مصرف‌کنندگان عینک‌های آفتابی و عکاسان کاملاً شناخته شده است). اگر یک پرتو نور (یا به عبارتی، یک پرتو از فوتون‌ها) از یک فیلتر پولاریزاسیون عبور داده شود، تمام فوتون‌هایی که از ورای فیلتر پدیدار شوند، در جهت محور آن فیلتر (مثلاً در جهت عمودی) پولاریزه خواهند شد. حالا اگر این پرتو از یک فیلتر پولاریزاسیونِ دوم عبور داده شود، شدت نوری که از فیلتر دوم پدیدار خواهد شد، متناسب خواهد بود با توان دومِ کسینوس زاویه‌ی میان محورها. اگر دو محور بر هم عمود باشند، هیچ فوتونی عبور نخواهد کرد. جهتِ مطلق این دو فیلتر مهم نیست؛ تنها موضوعی که به حساب می‌آید، زاویه‌ی میان محور آن‌هاست.

آلیس به منظور ایجاد یک پَد یک بارـ مصرف، به دو مجموعه از فیلترهای پولاریزاسیون نیاز دارد. مجموعه‌ی اول از یک فیلتر عمودی و یک فیلتر افقی تشکیل می‌شود. این گزینه، **قاعده‌ی راست ـ خط**[1] نامیده می‌شود. "قاعده" (مفردِ این واژه "basis" و جمعِ آن "bases" است) فقط یک سیستم مختصات است. دومین مجموعه از فیلترها نیز مشابه قاعده‌ی راست ـ خط است ولی نسبت به آن ۴۵ درجه چرخش دارد، بنابراین یک فیلتر از سمت پایین و چپ به سمت بالا و راست هدایت می‌کند و فیلتر دیگر از سمت بالا و چپ به سمت پایین و راست. این گزینه، **قاعده‌ی قطری**[2] نامیده می‌شود. لذا آلیس دو قاعده در اختیار دارد که می‌تواند آن‌ها را به دلخواه و سریعاً در پرتویی که متعلق به اوست قرار دهد. در عالم واقع، آلیس چهار فیلتر مجزا ندارد، بلکه یک کریستال دارد که پولاریزاسیونِ آن می‌تواند به صورت الکتریکی، با سرعتی زیاد به هر کدام از چهار جهتِ مُجاز سوئیچ شود. باب نیز تجهیزاتی مشابه آلیس در اختیار دارد. اینکه آلیس و باب هر کدام دو قاعدهٔ قابل دسترسی در اختیار دارند، اساسِ رمزنگاری کوانتومی است.

---

1. Rectilinear basis 2. Diagonal basis : قاعده‌ی قطری با زاویه‌ی ۴۵ درجه

اکنون آلیس برای هر قاعده، یک جهت را به عنوان 0 و جهت دیگر را به عنوان 1 مشخص می‌سازد. در مثالی که در زیر نشان داده شده، فرض می‌کنیم 0 را برای جهت عمودی و 1 را برای جهت افقی انتخاب کرده است. او همچنین مستقل از این موضوع، جهت پایین ـ چپ به سمت بالا ـ راست را به عنوان 0 و جهت بالا ـ چپ به سمت پایین ـ راست را به عنوان 1 انتخاب می‌کند. او این انتخاب‌ها را به صورت یک متن آشکار برای باب ارسال می‌کند.

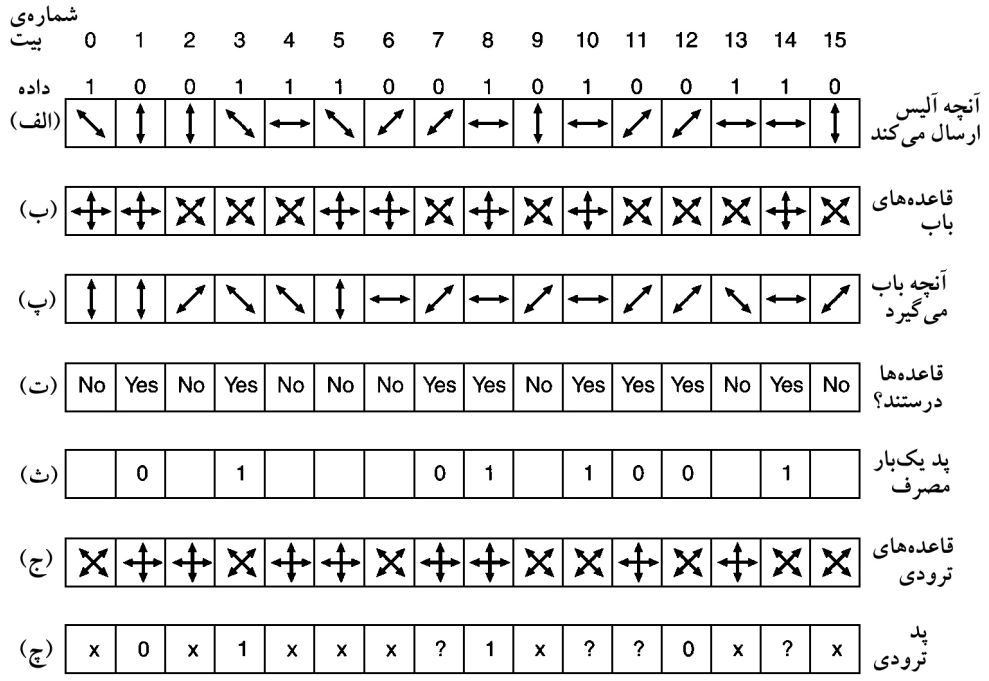
اکنون آلیس یک پَد یک بارـ مصرف برمی‌گزینَد که مثلاً مبتنی بر یک تولیدکننده‌ی عددِ تصادفی می‌باشد (که خودش موضوعِ کاملاً پیچیده‌ای است). آلیس این پَد را بیت به بیت به باب انتقال می‌دهد و برای هر بیت به طور تصادفی یکی از دو قاعده را انتخاب می‌کند. برای ارسال هر بیت، تفنگ فوتونِ آلیس، یک فوتون را ساطع می‌کند که این فوتون متناسب است با قاعده‌ای که آلیس برای آن بیت موردِ استفاده قرار داده است. مثلاً ممکن است آلیس این قاعده‌ها را انتخاب کرده باشد: قطری، راست ـ خطی، راست ـ خطی، قطری، راست ـ خطی، و همین‌طور تا آخر. برای آن‌که پَد یک بارـ مصرف آلیس به شکل 1001110010100110 با این قاعده‌ها ارسال شود، او فوتون‌هایی که در شکل ۸–۵ (الف) نشان داده شده را ارسال خواهد کرد. با داشتنِ پَد یک بارـ مصرف و توالیِ قاعده‌ها، پولاریزاسیونی که به ازای هر بیت به کار می‌رود به طور یکتا تعیین می‌شود. بیت‌های ارسالیِ یک فوتون در هر لحظه، **کیوبیت**[1] نامیده می‌شوند.

باب نمی‌داند از چه قاعده‌هایی استفاده می‌شود بنابراین همان‌طور که شکل ۸–۵(ب) نشان می‌دهد، برای هر فوتونی که می‌رسد، به طور تصادفی یک قاعده برمی‌دارد و فقط از همان استفاده می‌کند. اگر قاعده‌ی درست را انتخاب کند، بیتِ صحیح را به دست می‌آورَد. اگر قاعده‌ی نادرست را انتخاب کند، یک بیت تصادفی به دست می‌آورَد زیرا اگر یک فوتون به یک فیلترِ پولاریزه شده اصابت کند که با پولاریزاسیونِ خودش ۴۵ درجه زاویه داشته باشد، به طور تصادفی و با احتمالِ مساوی به پولاریزاسیونِ آن فیلتر و یا به پولاریزاسیونی که عمود بر فیلتر است، پرش می‌کند. این ویژگیِ فوتون‌ها مهم‌ترین اصلِ در مکانیک کوانتومی است. به این ترتیب، بعضی بیت‌ها صحیح و بعضی بیت‌ها تصادفی هستند اما باب نمی‌داند که کدام به کدام است. نتایج به دست آمده توسط باب در شکل ۸–۵(پ) نشان داده شده‌اند.

باب چطور بفهمد کدام قاعده‌ها را درست گرفته و کدام قاعده‌ها را نادرست گرفته است؟ باب به سادگی در متنی آشکار به آلیس می‌گوید برای هر بیت از چه قاعده‌هایی استفاده کرده و آلیس نیز در متنی آشکار به باب می‌گوید که کدام درست و کدام نادرست است (شکل ۸–۵(ت)). از طریق این اطلاعات،

---

۱. Qubit یا Quantum bit : در **پردازش کوانتومی** یک کیوبیت یا **بیتِ** کوانتومی، واحد پایه برای پردازش کوانتومی و **رمزنگاری کوانتومی** می‌باشد. یک کیوبیت به بیت کلاسیک شباهت‌هایی دارد، اما در کل بسیار متفاوت است. به این صورت که یک بیت کلاسیک *باید* در هر لحظه یا در حالت صفر و یا در حالت یک باشد، ولی یک کیوبیت *می‌تواند* در حالت صفر، یک و یا برهم نهیِ صفر و یک (superposition) نیز باشد (مترجم).



شکل ۸–۵ مثالی از رمزنگاری کوانتومی.

هر دوی آن‌ها می‌توانند یک رشته بیت از مفروضاتِ صحیح بسازند (شکل ۸–۵(ث)). به طور میانگین، طول این رشته بیت نصف طول رشته بیت اولیه خواهد بود اما چون هر دو طرف این موضوع را می‌دانند لذا می‌توانند آن را به عنوان پَد یک بارـ مصرف به کار ببرند. همه‌ی آنچه که آلیس بایستی انجام دهد عبارت‌استاز فرستادن یک رشته بیت که طولش کمی بیش از دو برابر طولِ مورد نظر است، و در نتیجه آلیس و باب یکَ پَد یک بارـ مصرف با طولِ مورد نظر خواهند داشت. کار انجام شده است.

اما صبر کنید! ترودی را فراموش کردیم. فرض کنید او نسبت به آنچه آلیس به باب گفته، کنجکاو است و فیبر را می‌بُرَد تا آشکارساز[1] و فرستنده‌ی[2] خود را کار بگذارد. ولی برایش متأسفیم چون نمی‌داند برای هر فوتون از چه قاعده‌ای استفاده کند. بهترین کاری که می‌تواند انجام دهد آن است که به صورت تصادفی یک قاعده برای هر فوتون بردارد، درست همان کاری که باب انجام می‌دهد. مثالی از انتخاب‌های او در شکل ۸–۵ (ج) نشان داده شده است. هنگامی که باب بعداً (در قالب متن آشکار) می‌گوید که از چه قاعده‌هایی استفاده کرده است و آلیس نیز به او می‌گوید (در قالب متن آشکار) که کدام قاعده‌ها صحیح هستند، حالا ترودی می‌فهمد که کدام یک از انتخاب‌های او درست و کدام نادرست بوده‌اند.در شکل ۸–۵، ترودی بیت‌های ۰، ۱، ۲، ۳، ۴، ۶، ۸، ۱۲، و ۱۳ را درست انتخاب کرده. اما ترودی از پاسخ آلیس در شکل ۸–۵ (ت) می‌داند که فقط بیت‌های ۱، ۳، ۷،

1. Detector 2. Transmitter

۸، ۱۰، ۱۱، ۱۲، و ۱۴ بخشی از پَد یک بارـ مصرف هستند. برای چهار بیت از این بیت‌ها (یعنی ۱، ۳، ۸ ، و ۱۲) حدس ترودی درست بوده و بیت صحیح را به دست آورده است. برای چهار بیتِ دیگر (یعنی ۷، ۱۰، ۱۱، و ۱۴) حدس اشتباه داشته و نمی‌داند چه بیتی فرستاده شده است. بنابراین، باب بر طبق شکل ۸–۵ (ث) می‌داند که پَد یک بارـ مصرف با 01011001 آغاز می‌گردد، اما مطابق شکل ۸–۵ (چ)، تمام آنچه ترودی می‌داند عبارت‌استاز 01?1??0? .

البته آلیس و باب آگاهند که ممکن است ترودی بخشی از پَد یک بارـ مصرف آن‌ها را به دست آورده باشد، لذا مایل خواهند بود که اطلاعاتی که ترودی دارد را کاهش دهند. آن‌ها می‌توانند این کار را با یک تغییر شکل بر روی اطلاعات انجام دهند. برای مثال، آن‌ها می‌توانند پَد یک بارـ مصرف را به بلوک‌های ۱۰۲۴ـ بیتی تقسیم کنند، هر بلوک را به توان ۲ برسانند تا یک عدد ۲۰۴۸ـ بیتی شکل بگیرد، و از الحاق[1] این اعداد ۲۰۴۸ـ بیتی به عنوان پَد یک بارـ مصرف استفاده کنند. ترودی با اطلاع ناقصی که از رشته بیتِ ارسالی دارد هیچ راهی برای تولید توان دومِ آن ندارد، پس یعنی هیچ ندارد. تغییر شکل از پَد یک بارـ مصرف اولیه به یک پَدِ متفاوت با آن، با هدف کاهش دانسته‌های ترودی، **تقویتِ محرمانگی**[2] نامیده می‌شود. در عمل از تغییر شکل‌های پیچیده به جای به توان ۲ رساندن استفاده می‌شود، به طوری که هر بیت خروجی بستگی به هر بیت ورودی داشته باشد.

بیچاره ترودی. نه تنها هیچ ایده‌ای در مورد این‌که پَد یک بارـ مصرف چیست، ندارد بلکه حضور او دیگر یک راز نیست. بعد از تمام این کارها، او مجبور است تمام بیت‌های رسیده را به باب بازپخش نماید تا باب فکر کند در حال گفتگو با آلیس است. زحمتی که بر دوش او افتاده این است که کیوبیتی که می‌گیرد را، با استفاده از پولاریزاسیونی که برای دریافتِ آن کیوبیت به کار می‌بَرد، انتقال دهد. ولی چون نیمی از مواقع در اشتباه است، خطاهای زیادی در پَد یک بارـ مصرف باب ایجاد می‌کند.

هنگامی‌که سرانجام آلیس شروع به ارسال داده می‌کند، داده‌اش را با استفاده از یک کدِ سنگین با امکانِ تصحیح خطای پیش‌رانش، کدگذاری می‌کند. از دید باب، یک بیت خطا در پَد یک بارـ مصرف معادل است با یک بیت خطای انتقال زیرا به هر حال یک بیتِ نادرست دریافت کرده است. اگر تصحیح خطای پیش‌رانشِ مناسبی وجود داشته باشد، علی‌رغم تمام خطاها قادر به بازیافت پیغام اولیه خواهد بود، با این حال او به راحتی می‌تواند تعداد خطاهای تصحیح شده را شمارش کند. اگر این عدد خیلی بیشتر از نرخ خطای مورد انتظار برای تجهیزات باشد، می‌فهمد که ترودی به خط نفوذ کرده و بنابراین می‌تواند واکنش مناسب نشان دهد (مثلاً به آلیس بگوید که به یک کانال رادیویی سوئیچ کند، پلیس را خبر کند، یا واکنشی از این دست). اگر ترودی راهی برای نسخه‌برداریِ عینی از فوتون داشت (به نحوی که یک فوتون برای بررسیِ خودش داشت و یک فوتون برای ارسال به باب) می‌توانست از شناسایی شدن بر حذر بماند، اما فعلاً هیچ راهِ شناخته شده‌ای برای نسخه‌برداریِ عینی از یک فوتون

1. Concatenation 2. Privacy amplification

وجود ندارد. و حتی اگر هم ترودی قادر به نسخه‌برداریِ عینی می‌بود، باز هم چیزی از ارزش رمزنگاریِ کوانتومی در برقراری پَدهای یک بارـ مصرف کاسته نمی‌شد.

گرچه نشان داده شده که رمزنگاری کوانتومی در فیبرهایی به مسافت ۶۰ کیلومتر عمل می‌کند، ولی تجهیزات مربوطه پیچیده و گران‌قیمت هستند. این ایده آینده‌ی خوبی دارد. برای اطلاعات بیشتر درباره‌ی رمزنگاری کوانتومی به Mullins (۲۰۰۲) مراجعه نمایید.

۸–۱–۵ دو اصل بنیادین در رمزنگاری

هر چند در صفحاتی که پیش رو داریم، سیستم‌های رمزنگاری متعدد و متفاوتی را مطالعه خواهیم کرد، درک دو اصل بنیادین در تمام آن‌ها مهم است. پس خوب دقت کنید. البته وقتی آن‌ها را نقض کنید در خطر قرار می‌گیرید.

افزونگی

اولین اصل این است که همه‌ی پیغام‌های رمزگذاری شده باید تا اندازه‌ای افزونگی[1] داشته باشند، یعنی اطلاعاتی که برای درک پیغام، ضروری نیستند. ممکن است با یک مثال دلیل لزوم این افزونگی روشن شود. یک شرکتِ سفارش کالا از طریق پُست را در نظر بگیرید. مثلاً شرکت Couch Potato[2] (با نام اختصاری TCP) با ۶۰,۰۰۰ میلیون محصول. با این تفکر که این شرکت بسیار کارآمد است، برنامه‌نویسان TCP تصمیم گرفتند پیغام‌های سفارش کالا بهتر است شامل یک فیلد ۱۶ـ بایتی نام مشتری و سپس یک فیلد ۳ـ بایتیِ داده (data) باشند (یک بایت برای مقدار کالا و ۲ بایت برای شماره‌ی کالا). این ۳ بایت آخر باید با استفاده از یک کلید بسیار طولانی، فقط توسط مشتری و شرکت TCP رمزگذاری شوند.

ابتدا ممکن است به نظر امن برسد و تا اندازه‌ای نیز همین‌طور است، زیرا نفوذگرانِ غیرفعال نمی‌توانند این پیغام‌ها را رمزبَرداری کنند. ولی متأسفانه هنوز هم یک ضعفِ مخرب در آن هست که این روش را به درد ـ نخور می‌کند. فرض کنید یک کارمند که به تازگی از خدمت منفصل شده بخواهد بابت اخراجش، شرکت TCP را تنبیه کند. او قبل از ترک شرکت، لیست مشتریان را با خود برمی‌دارد. او شبانه برنامه‌ای می‌نویسد که لیستی از سفارشات دروغین با استفاده از نام‌های مشتریانِ واقعی تولید کند. از آنجایی که لیست کلیدها را ندارد، یک عدد ۳ـ بایتیِ تصادفی در ۳ بایتِ آخر قرار می‌دهد و صدها سفارش به شرکت TCP ارسال می‌کند.

هنگامی‌که این پیغام‌ها برسند، کامپیوتر شرکت TCP از نام مشتری‌ها استفاده می‌کند تا کلید را مشخص کرده و پیغام را رمزبَرداری نماید. طفلک TCP، تقریباً تمام ۳ بایت در پیغام‌ها معتبر هستند،

---

1. Redundancy

۲. The Couch Potato : این نام در واقع عبارتی به معنای "معتاد به تلویزیون" است که در این کتاب به عنوان نام شرکت انتخاب شده است (مترجم).

بنابراین کامپیوتر شروع به چاپ دستورهای حمل می‌کند. هر چند ممکن است به نظر عجیب برسد که یک مشتری ۸۳۷ ننوی بچه یا ۵۴۰ جعبه‌ی شنی مخصوص بازی‌های کودکان سفارش دهد، ولی از دید کامپیوتر ممکن است مشتری در حال گرفتنِ مجوزِ افتتاح تعدادی زمین بازیِ زنجیره‌ای باشد. به این ترتیب، یک نفوذگرِ فعال (کارمند پیشین) می‌تواند مشکل زیادی ایجاد کند، هر چند او نمی‌تواند بفهمد کامپیوترش چه پیغام‌هایی تولید کرده است.

این مسئله می‌تواند با اضافه کردنِ افزونگی به تمام پیغام‌ها حل شود. برای مثال، اگر پیغام‌های سفارش کالا به ۱۲ بایت برسند، ۹ بایتِ اول از هر پیغام باید صفر باشند و این حمله دیگر عمل نمی‌کند زیرا کارمند پیشین دیگر نمی‌تواند یک رشته‌ی طولانی از پیغام‌های معتبر تولید کند. نتیجه‌ی اخلاقیِ این داستان آن است که تمام پیغام‌ها باید افزونگی قابل توجهی داشته باشند تا نفوذگرانِ فعال نتوانند پیغام‌های به دردنخورِ تصادفی ارسال کرده و سیستم را مجبور کنند تا آن‌ها را به عنوان پیغام‌های معتبر تفسیر کند.

اما اضافه‌کردنِ افزونگی، شکستنِ پیغام‌ها را برای رمزیاب‌ها آسان‌تر می‌سازد. فرض کنید رقابت زیادی در حرفه‌ی سفارشات پُستی وجود دارد، و رقیب اصلی Couch Potato یعنی شرکت Sofa Tuber بسیار علاقمند است بداند تعداد فروش جعبه‌های شنیِ مخصوص بازی‌های کودکان چقدر بوده است، پس Sofa Tuber وارد خط تلفن شرکت TCP می‌شود. در نظام اولیه که پیغام‌های ۳ـ بایتی داشت، رمزیابی تقریباً غیرممکن بود زیرا بعد از حدس زدنِ کلید، رمزیاب هیچ راهی برای تعیین صحت کلید نداشت، چون که تقریباً تمام پیغام‌ها به لحاظ فنیِ صحیح (legal) بودند. با نظام ۱۲ـ بیتیِ جدید، تعیین پیغام معتبر از غیرمعتبر برای رمزیاب، عملی آسان است. بنابراین:

اصل شماره‌ی ۱ در رمزنگاری: پیغام‌ها بایستی حاوی مقداری افزونگی باشند.

به عبارت دیگر، در رمزبَرداری از یک پیغام، گیرنده باید به آسانی با بررسی پیغام و احتمالاً انجام یک محاسبه‌ی ساده قادر باشد بگوید آیا پیغام معتبر هست یا خیر. این افزونگی لازم است تا از این‌که نفوذگرانِ فعال آت و آشغال ارسال کنند و به دریافت‌کننده حقه بزنند، جلوگیری شود. نفوذگران با ارسال آشغال، دریافت‌کننده را وادار می‌کنند تا این مطالب بی‌ارزش را رمزبَرداری کرده و بر روی "متن آشکار" کار کند. اما همین افزونگی کارِ شکستن سیستم را برای نفوذگرانِ غیرفعال بسیار آسان‌تر می‌سازد. پس می‌بینیم که اوضاع تا حدودی وخیم است. علاوه بر این، افزونگی هرگز نباید به شکل n عدد 0 در ابتدا یا انتهای پیغام باشد زیرا تأثیر اجرای الگوریتم‌های رمزگونه بر روی چنین پیغام‌هایی منجر به نتایجی می‌شود که قابل پیش‌بینی هستند و کار رمزیاب‌ها را آسان‌تر می‌کند. یک چند ـ جمله‌ایِ CRC خیلی بهتر از یک‌سری 0 است چون در عین حال که دریافت‌کننده می‌تواند آن را راحت‌ترِ راستی‌آزمایی کند، ولی زحمت بیشتری برای رمزیاب خواهد داشت. حتی بهتر است از یک

رمزگونه با روش درهم‌سازی (cryptographic hash) استفاده شود. این مفهوم را بعدتر بررسی خواهیم کرد. فعلاً آن را به عنوان یک روش که از CRC بهتر است، در نظر بگیرید.

اگر موقتاً به موضوع رمزنگاری کوانتومی برگردیم، می‌توانیم مشاهده کنیم که چگونه افزونگی در آن‌جا نقش دارد. چون ترودی فوتون‌ها را رهگیری می‌کند (interception) لذا بعضی بیت‌ها در پَد یک بار ـ مصرفِ باب غلط خواهند شد. باب در پیغام‌هایی که می‌رسند به افزونگی نیاز دارد تا وجود خطا را مشخص نماید. یک شکل ناپخته و اولیه از افزونگی عبارت‌است‌از تکرارِ دوباره‌ی همان پیغام. اگر دو کپی عین هم نباشند، باب متوجه می‌شود که یا نویز زیادی در فیبر بوده و یا آن‌که شخصی در حال کنجکاوی در این عملیاتِ انتقال است. البته این‌که هر چیزی را دو بار ارسال کنیم خیلی افراط‌گرایانه است؛ استفاده از کد همینگ یا کد رید ـ سالومون برای تشخیص و تصحیح خطا، بسیار کارآمدتر است. اما منظورِ اصلی ما این است که نشان دهیم افزونگی لازم است تا یک پیغام معتبر از یک پیغام غیرمعتبر، تمیز داده شود مخصوصاً در مواجهه با یک نفوذگرِ فعال.

تازگی

اصل رمزگونه‌ی دوم عبارت‌است‌از این‌که بایستی با به‌کارگیریِ تدابیری مطمئن شویم که هر پیغامی که دریافت می‌شود، می‌تواند از نظر تازگی (یعنی این‌که آن پیغام، اخیراً ارسال شده باشد) راستی‌آزمایی شود. لزوم وجود این تدبیر از این جهت است که نفوذگرانِ فعال از بازنواختِ پیغام‌های قدیمی ممانعت شوند. در صورتی که چنین تدبیرهایی وجود نداشته باشند، کارمند اخراجیِ ما می‌تواند خط تلفنِ TCP را قطع کرده و پیغام‌های معتبری که قبلاً ارسال شده‌اند را تکرار کند. بنابراین

اصل شماره‌ی ۲ در رمزنگاری: نیاز به روشی برای دفع حمله‌های بازنواخت[1] وجود دارد.

چنین تدبیری باعث می‌شود هر پیغامی یک برچسب زمانی داشته باشد که مثلاً فقط به مدت ۱۰ ثانیه معتبر باشد. دریافت‌کننده می‌تواند پیغام‌ها را فقط در حدود ۱۰ ثانیه نگه دارد، و پیغام‌هایی که جدیداً می‌رسند را با قبلی‌ها مقایسه کند تا بتواند پیغام‌های تکراری را فیلتر نماید. پیغام‌های قدیمی‌تر از ۱۰ ثانیه می‌توانند بیرون ریخته شوند زیرا هر بازنواختی که ارسالش مربوط به بیش از ۱۰ ثانیه قبل باشد، به عنوان پیغامی که بیش از اندازه قدیمی شده، مردود می‌شود. بعداً تدابیری غیر از برچسب زمانی را هم بررسی خواهیم کرد.

۸–۲ الگوریتم‌های کلید ـ متقارن

رمزنگاریِ امروزی از همان ایده‌ی پایه در رمزنگاری سنتی (یعنی جابجاسازی و جایگزین‌سازی) استفاده می‌کند، اما تأکید متفاوتی دارد. به طور سنتی، رمزنگاران از الگوریتم‌های ساده استفاده کرده‌اند. امروزه عکسِ این موضوع صحت دارد: هدف این است که الگوریتم رمزگذاری آنقدر پیچیده و پُرپیچ

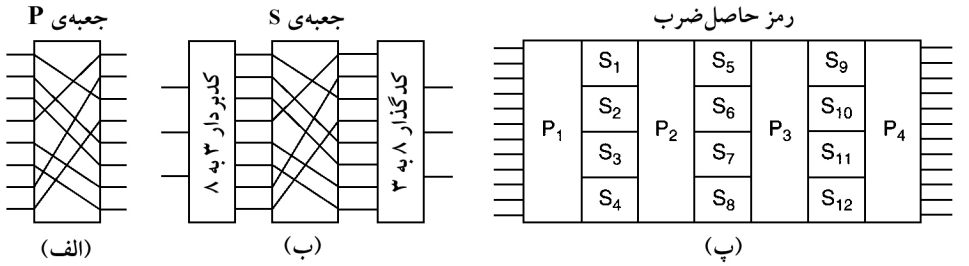
1. Replay attack

و خم شود که حتی اگر رمزیاب بتواند به انتخاب خودش، حجم زیادی از متنِ رمز شده را به دست آورَد، باز هم بدون کلید قادر به فهمیدنِ متن نباشد.

اولین کلاس از الگوریتم‌های رمزگذاری که در این فصل مطالعه خواهیم کرد، **الگوریتم‌های کلید ـ متقارن**[1] نامیده می‌شوند زیرا هم در رمزگذاری و هم در رمزبَرداری از یک کلید یکسان استفاده می‌کنند. شکل ۸-۲ کار یک الگوریتم کلید ـ متقارن را نشان می‌دهد. ما مشخصاً بر روی **رمزهای بلوکی**[2] تمرکز خواهیم کرد. این رمزها یک بلوک n ـ بیتی از متن آشکار را به عنوان ورودی می‌گیرند و با استفاده از کلید، آن را به یک بلوک n ـ بیتی از متن رمزی تبدیل می‌کنند.

الگوریتم‌های رمزگونه می‌توانند توسط سخت‌افزار (به دلیل سرعتِ بیشتر) یا توسط نرم‌افزار (به دلیل انعطاف‌پذیریِ بیشتر) پیاده‌سازی شوند. هرچند بیشترین توجه ما معطوف به الگوریتم‌ها و پروتکل‌ها می‌باشد (که مستقل از پیاده‌سازیِ واقعی هستند) ولی مایلیم چند کلمه‌ای هم راجع به ساخت سخت‌افزار رمزگونه صحبت کنیم. جابجاسازی‌ها و جایگزین‌سازی‌ها می‌توانند با مدارهای الکتریکیِ ساده پیاده‌سازی شوند. شکل ۸-۶(الف) دستگاهی به نام **جعبه‌ی P** (یا **جعبه‌ی جایگشت**[3]) را نشان می‌دهد که برای انجام یک جابجاسازی روی یک ورودی ۸ ـ بیتی به کار رفته است. اگر ۸ بیتی که داریم از بالا به پایین به صورت 01234567 در نظر گرفته شوند، خروجی این جعبه‌ی P برابر با 36071245 خواهد بود. با انجام سیم‌بندی داخلیِ مناسب، یک جعبه‌ی P می‌توان ساخت که هر جابجاسازی‌ای را انجام دهد، و این کار با سرعت واقعی نور انجام خواهد شد زیرا هیچ محاسبه‌ای نیاز ندارد و فقط انتشار سیگنال داریم. این طراحی از اصل کیرشهف تبعیت می‌کند: مهاجم می‌داند روش کلی عبارت‌است‌از پس و پیش کردنِ بیت‌ها. آنچه نمی‌داند این است که کدام بیت به کجا می‌رود.

همان‌طور که در شکل ۸-۶(ب) نشان داده شده، جایگزین‌سازی‌ها توسط **جعبه‌های S** انجام می‌شوند. در این مثال یک متن آشکار ۳ـ بیتی وارد می‌شود و یک متن رمزی ۳ـ بیتی خارج می‌گردد. ورودی ۳ـ بیتی یکی از هشت خط موجود در مرحله‌ی اول (یا طبقه‌ی اول first stage) را انتخاب کرده و آن را به مقدار 1 تنظیم می‌کند؛ تمام خطوط دیگر 0 هستند. مرحله‌ی بعدی یک جعبه‌ی P است.



شکل ۸-۶ عناصر اصلی در ضرب کردنِ رمزها. **(الف)** جعبه‌ی P . **(ب)** جعبه‌ی S . **(پ)** حاصل‌ضرب.

1. Symmetric-key algorithm 2. Block cipher 3. Permutation-box

مرحله‌ی سوم مجدداً خط ورودیِ انتخاب شده را به شکل دودویی، کد می‌کند. با سیم‌بندیِ نشان داده شده، اگر هشت عددِ پایه‌ی ۸ ("پایه‌ی ۸" یا همان "هشت ـ هشتی") به صورت 01234567 یکی بعد از دیگری، ورودی مدار باشند در این صورت ترتیبِ خروجی 24506713 خواهد بود. به عبارت دیگر عدد 0 با 2 ، عدد 1 با 4 ، و بقیه نیز به همین ترتیب جایگزین می‌شوند. بار دیگر با سیم‌بندیِ مناسب جعبه‌ی P درون جعبه‌ی S، هر جایگزین‌سازی‌ای قابل اجرا خواهد بود. مهم آن است که چنین دستگاهی می‌تواند در سخت‌افزار ساخته شود و باعث دستیابی به سرعت بالایی شود زیرا کدگذارها و کدبَردارها فقط یک یا دو تأخیر gate (در حد نانوثانیه) دارند و زمان انتشار در جعبه‌ی P می‌تواند کمتر از ۱ پیکوثانیه باشد.

قدرت واقعی این عناصر پایه‌ای زمانی آشکار می‌شود که یک سری کامل از جعبه‌ها را به شکل آبشار قرار دهیم تا همان‌طور که در شکل ۸-۶ (پ) نشان داده شده، یک **رمز حاصل‌ضربی**[1] تشکیل شود. در این مثال ۱۲ خط ورودی در مرحله‌ی اول نسبت به هم جابجا (پس و پیش) می‌شوند (P_1). در مرحله‌ی دوم ورودی حداکثر به چهار گروه ۳ـ بیتی شکسته می‌شود. هر یک از این گروه‌ها مستقل از بقیه، جایگزین‌سازی می‌شوند (S_1 تا S_4). این آرایش روش شبیه‌سازیِ یک جعبه‌ی S بزرگ‌تر، از چندین جعبه‌ی S کوچک‌تر را نشان می‌دهد. علت سودمندی این روش آن است که پیاده‌سازی سخت‌افزاریِ جعبه‌های S کوچک، شدنی است (مثلاً یک جعبه‌ی S ۸ ـ بیتی می‌تواند به عنوان یک جدول جستجو با ۲۵۶ درایه در نظر گرفته شود)، اما ساخت جعبه‌های S بزرگ کار دست و پاگیری است (مثلاً یک جعبه‌ی S ۱۲ـ بیتی در مرحله‌ی میانی‌اش دست کم به $2^{12} = 4096$ سیم متقاطع (crossed wires) نیاز دارد). هرچند این روش عمومیت کمی دارد ولی هنوز هم قدرتمند است. اگر تعداد مراحل زیاد و کافی در رمز حاصل‌ضربی قرار داده شود، آنگاه خروجی می‌تواند تابعی بسیار پیچیده از ورودی باشد.

رمزهای حاصلضربی که بر روی ورودی‌های k ـ بیتی عمل می‌کنند تا خروجی‌های k ـ بیتی بسازند، بسیار متداولند. مقدار k معمولا بین ۶۴ تا ۲۵۶ است. یک پیاده‌سازی سخت‌افزاری به جای آن‌که مانند شکل ۸-۶ (پ) فقط ۷ مرحله داشته باشد، معمولاً دست کم ۱۰ مرحله‌ی فیزیکی دارد. پیاده‌سازی نرم‌افزاری به صورت یک حلقه با دست کم هشت تکرار برنامه‌ریزی می‌شود، به طوری‌که هر حلقه جایگزین‌سازی‌های از نوع جعبه‌ی S را بر روی زیرـ بلوک‌های مربوط به بلوک‌های داده‌ی ۶۴ـ بایتی تا ۲۵۶ـ بایتی انجام می‌دهد. به دنبال این عمل، جایگشتی صورت می‌گیرد که خروجی‌های جعبه‌های S را مخلوط می‌کند. غالباً یک جایگشت مشخصِ راه‌انداز و نیز یک جایگشت مشخص در انتهای کار وجود دارد. تکرارها اصطلاحا **راند**[2] (یا نوبت اجرا) نامیده می‌شوند.

---

1. Product cipher 2. Round

۸–۲–۱ DES ــ استاندارد رمزگذاری داده

در ژانویه‌ی ۱۹۷۷ دولت ایالات متحده یک رمز حاصل‌ضربی که توسط IBM توسعه یافته بود را به عنوان استاندارد رسمی خود برای اطلاعاتِ طبقه‌بندی نشده، انتخاب کرد. این رمز که نامش **DES** بود (**استاندارد رمزگذاری داده**[1]) در ارتباط با محصولات امنیتی، از طرف صاحبان صنعت مورد استفاده‌ی گسترده قرار گرفت. این رمز دیگر به شکل اولیه‌اش ایمن نیست، ولی هنوز هم شکلِ ویرایش شده‌اش مورد استفاده می‌باشد. اکنون نحوه‌ی کار DES را شرح خواهیم داد.

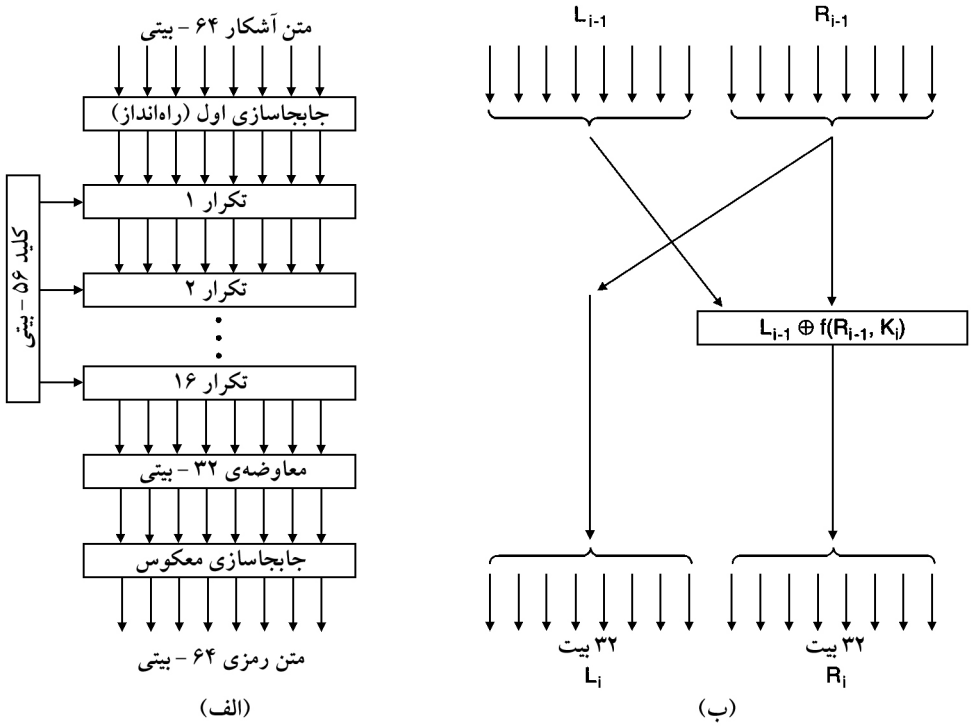
یک نمای کلی از DES در شکل ۸–۷(الف) نشان داده می‌شود. متن آشکار در بلوک‌های ۶۴ ـ بیتی رمز می‌شود و نتیجه‌اش ۶۴ بیت متن رمزی است. الگوریتم مربوطه که توسط یک کلید ۵۶ ـ بیتی پارامتربندی می‌شود، ۱۹ مرحله‌ی مجزا دارد. مرحله‌ی اول عبارت‌است‌از یک جابجاسازیِ مستقل از کلید بر روی متن آشکار ۶۴ ـ بیتی. مرحله‌ی آخر دقیقاً معکوس این جابجاسازی است. مرحله‌ی ماقبل آخر، سمت چپ‌ترین ۳۲ بیت را با سمت راست‌ترین ۳۲ بیت، تعویض می‌کند (exchange). شانزده مرحله‌ی باقیمانده به لحاظ عملکرد همانند هستند اما به وسیله‌ی توابع متفاوتی از کلید، پارامتربندی می‌شوند. این الگوریتم چنان طراحی شده که اجازه می‌دهد رمزبَرداری با همان کلیدِ رمزگذاری انجام شود (این ویژگی برای تمام الگوریتم‌های کلید ـ متقارن، الزامی است) و فقط ترتیب اجرای مراحل کار معکوس می‌شود.

در شکل ۸–۷(ب) عملیات مربوط به یکی از این مراحلِ بینابینی، به تصویر درآمده است. هر مرحله دو ورودی ۳۲ ـ بیتی را می‌گیرد و دو خروجی ۳۲ ـ بیتی تولید می‌کند. خروجی سمت چپ به سادگی کپیِ ورودی سمت راست است. خروجی سمت راست عبارت‌است‌از XOR بیت‌های مربوط به ورودی سمت چپ و یک تابع از ورودی سمت راست و کلیدِ مربوط به این مرحله (یعنی K_i). تقریباً تمام پیچیدگی الگوریتم در این تابع نهفته است.

تابع از چهار مرحله تشکیل شده که به ترتیب اجرا می‌شوند. ابتدا یک عدد ۴۸ ـ بیتیِ E با استفاده از گسترش دادنِ R_{i-1} ۳۲ ـ بیتی (بر اساس یک جابجاسازیِ ثابت و قانونِ تکثیر و نسخه‌برداری) ساخته می‌شود. سپس E و K_i با هم XOR می‌شوند. سپس این خروجی به ۸ گروهِ ۶ ـ بیتی اِفراز می‌شود. هر گروه به یک جعبه‌ی S متفاوت تغذیه می‌شود. هر کدام از ۶۴ ورودی‌های ممکن در یک جعبه‌ی S، به یک خروجیِ ۴ ـ بیتی نگاشت می‌شوند. در آخر، این ۴ × ۸ بیت از یک جعبه‌ی P عبور داده می‌شوند.

در هر یک از ۱۶ تکرار، از یک کلید متفاوت استفاده می‌شود. قبل از شروع به کار الگوریتم، یک جابجاسازیِ ۵۶ ـ بیتی بر روی کلید اِعمال می‌گردد. درست قبل از هر تکرار، کلید به دو واحد ۲۸ ـ بیتی اِفراز می‌شود. هر یک از این دو واحد به سمت چپ چرخش دارند و تعداد بیت‌های چرخش

1. Data Encryption Standard



شکل ۸-۷ استاندارد رمزگذاری داده. **(الف)** نمای عمومی. **(ب)** جزئیات مربوط به یک تکرار. علامت + که دورش یک دایره است به معنای XOR می‌باشد.

بستگی دارد به تعداد تکرار. با اِعمال کردنِ یک جابجاسازیِ ۵۶ ـ بیتیِ دیگر، از این کلیدِ چرخش یافته، K_i استخراج می‌شود. در هر راند، از این ۵۶ بیت یک زیرمجموعه‌ی ۴۸ ـ بیتیِ متفاوت استخراج و پس و پیش (Permute) می‌شود.

بعضی اوقات از روشی به نام **سفیدکاری**[1] برای قدرتمندتر کردنِ DES استفاده می‌شود. این روش شامل XOR کردنِ یک کلید تصادفی ۶۴ ـ بیتی با متن رمزیِ حاصله است (قبل از آن‌که جابجاسازی را روی آن انجام دهند). عمل سفیدکاری به سادگی می‌تواند با اجرای عملیات عکس، برطرف گردد (اگر دریافت‌کننده دو کلید برای سفیدکاری داشته باشد). چون این روش به نحو چشمگیری به تعداد بیت‌های طول کلید اضافه می‌کند، لذا یک جستجوی جامع در فضای کلید، زمان بسیار زیادی صرف می‌کند. توجه داشته باشید که برای هر بلوک، از کلید سفیدکاریِ یکسانی استفاده می‌شود (به عبارت دیگر تنها یک کلید سفیدکاری وجود دارد).

روش DES از همان روزی که کارش را شروع کرده، اطرافش بحث و مجادله بوده است. این روش بر مبنای لوسیفِر[2] بود که یک رمز توسعه یافته و به ثبت رسیده توسط IBM بود، با این تفاوت

1. Whitening 2. Lucifer

که رمز IBM به جای کلید ۵۶ ـ بیتی، از یک کلید ۱۲۸ ـ بیتی استفاده می‌کرد. هنگامی‌که دولت فدرال ایالات متحده تصمیم به استانداردسازیِ یک رمز برای کاربردهای طبقه‌بندی نشده گرفت، IBM را "دعوت" به "بحث و بررسی" موضوع با NSA (آژانس امنیت ملی) کرد. آژانس NSA بازوی دولت ایالات متحده در شکستنِ کد است و بیشترین تعداد کارکنان متخصص ریاضیدان و رمزشناس در جهان را در اختیار دارد. آژانس NSA آنقدر سرّی است که در موردش این لطیفه را گفته‌اند:

سؤال: NSA کوته‌نوشتِ چه عبارتی است؟

جواب: No Such Agency (چنین آژانسی وجود ندارد).

البته در حقیقت NSA کوته‌نوشت National Security Agency (آژانس امنیت ملی) می‌باشد.

بعد از آن‌که چنین مباحثی درگرفت، IBM کلید را از ۱۲۸ بیت به ۵۶ بیت کاهش داد و تصمیم گرفت فرآیند طراحی DES را سرّی نگه دارد. خیلی از مردم تصور کردند که طول کلید به این دلیل کاهش یافت که NSA بتواند DES را بشکند، اما هیچ سازمان دیگری با بودجه‌ای کمتر از بودجه‌ی NSA، نتواند این کار را بکند. هنگامی‌که یکی از کارکنان NSA با احتیاط به IEEE گفت که یک کنفرانسِ از قبل طراحی شده درباره‌ی رمزنگاری را لغو کند، این کار خیالِ مردم را راحت‌تر نکرد. البته NSA همه چیز را انکار کرد.

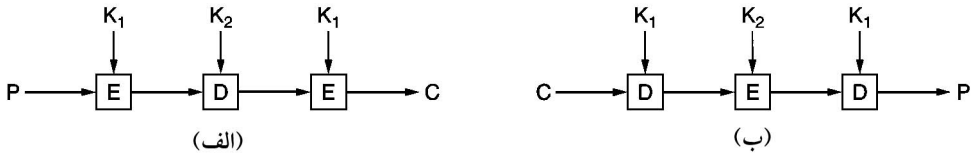
در سال ۱۹۷۷، دو پژوهشگر رمزنگاری در استنفورد به نام‌های Diffie و Hellman (۱۹۷۷) ماشینی برای شکستنِ DES طراحی کردند و ارزیابی‌شان این بود که با ۲۰ میلیون دلار امکان ساخت دارد. این ماشین با گرفتنِ یک تکه‌ی کوچک از متن آشکار و متن رمزیِ منطبق با آن، با انجام جستجوی دقیق و جامعِ فضای آدرسی با 2^{56} درایه، ظرف ۱ روز کلید را پیدا می‌کرد. امروزه، این بازی تمام شده است. چنین ماشینی وجود دارد، فروخته می‌شود، و هزینه‌ی ساخت آن کمتر از ۱۰،۰۰۰ دلار است (Kumar و همکاران، ۲۰۰۶).

DES سه‌گانه[1]

اوایل سال ۱۹۷۹ شرکت IBM متوجه شد که طول کلید DES بیش از اندازه کوتاه است و روشی برای افزایش موثر آن، با استفاده از رمزگذاری سه‌گانه ابداع کرد (Tuchman، ۱۹۷۹). این روش که در استاندارد بین‌المللی 8732 گنجانیده شده، در شکل ۸–۸ به تصویر درآمده است. در این‌جا از دو کلید و سه مرحله استفاده می‌شود. در مرحله‌ی اول، متن آشکار با استفاده‌ی عادی از DES و با کلید K_1 رمزگذاری می‌شود. در مرحله‌ی دوم، DES در مود رمزبَرداری و با استفاده از K_2 به عنوان کلید، به کار می‌رود. در آخر، یک رمزگذاری DES دیگر با K_1 انجام می‌شود.

این طراحی بلافاصله دو سؤال را به ذهن می‌آورَد. اول آن‌که، چرا به جای سه کلید فقط از دو کلید استفاده می‌شود؟ دوم آن‌که، چرا به جای **EEE** (**رمزگذاری رمزگذاری رمزگذاری**[2]) از **EDE**

1. Triple DES 2. Encrypt Encrypt Encrypt



شکل ۸-۸ (الف) رمزگذاری سه‌گانه با استفاده از DES . **(ب)** رمزبَرداری.

(**رمزگذاری رمزبَرداری رمزگذاری**[1]) استفاده می‌شود؟ دلیل استفاده از دو کلید آن است که حتی بدگمان‌ترین رمزنگاران نیز عقیده دارند که ۱۱۲ بیت فعلاً برای کاربردهای تجاریِ عادی مناسب است. (البته در میان رمزنگاران، بدگمان بودن یک حُسن است و عیب محسوب نمی‌شود.) رفتن به سراغ ۱۶۸ بیت فقط یک سربار مدیریتیِ غیرضروری و نیز لزوم حمل یک کلیدِ دیگر، به ازای یک سود کم، به مجموعه اضافه می‌کند.

دلیل رمزگذاری، رمزبَرداری، و دوباره رمزگذاری عبارت‌است‌از سازگاری با سیستم‌های DES تک ـ کلیدیِ موجود. هم تابع رمزگذاری و هم تابع رمزبَرداری، نگاشت‌هایی مابین مجموعه‌هایی از اعداد ۶۴ ـ بیتی هستند. از نقطه نظر رمزنگاری، این دو نگاشت به لحاظ قدرتمندی، یکسان هستند. اما اگر به جای EEE از EDE استفاده شود، کامپیوتری که از رمزگذاری سه‌گانه استفاده می‌کند می‌تواند با کامپیوتری که از رمزگذاری تکی استفاده می‌کند، ارتباط برقرار نماید و کافی است $K_1 = K_2$ قرار دهد. این ویژگی به رمزگذاری سه‌گانه اجازه می‌دهد تا به تدریج جایگزین DES شود. این ویژگی اهمیت زیادی برای IBM و مشتریانش دارد ولی رمزنگارانِ دانشگاهی علاقه‌ای به آن ندارند.

۸-۲-۲ AES ــ استاندارد رمزگذاری پیشرفته

با نزدیک شدنِ DES به پایان عمر مفیدش، حتی با DES سه‌گانه، **NIST** (**موسسه‌ی ملی استانداردها و فناوری**[2]) مقرر کرد که دولت به یک استاندارد رمزنگاری جدید برای کاربردهای طبقه‌بندی نشده، نیاز دارد (NIST آژانسی مربوط به وزارت اقتصاد ایالات متحده است که مسئول تصویب استانداردها برای دولت فدرال ایالات متحده می‌باشد). موسسه‌ی NIST کاملاً از جنجالی که در اطراف DES برپا بود، باخبر بود و به خوبی می‌دانست اگر استاندارد جدیدی را فقط اعلام هم بکند، هر کسی که اطلاعی درباره‌ی رمزنگاری داشته باشد خودبه‌خود خیال خواهد کرد که NSA یک درِ پشتی به این استاندارد جدید خواهد داشت، بنابراین هر آنچه که با این استاندارد، رمزگذاری شود توسط NSA قابل خواندن خواهد بود. در چنین شرایطی، احتمالاً هیچ کسی از این استاندارد استفاده نخواهد کرد و استاندارد جدید، کاملاً از بین خواهد رفت.

به همین دلیل، NIST رویکرد کاملاً متفاوتی نسبت به رویه‌ی دیوان‌سالاریِ دولتی در پیش گرفت: حمایت مالی از (رقابت در) رمزنگاری. در ژانویه‌ی سال ۱۹۹۷ پژوهشگران سراسر دنیا جهت

1. Encrypt Decrypt Encrypt 2. National Institute of Standards and Technology

ارائه‌ی طرح‌هایشان برای یک استاندارد جدید، به نام **AES** (**استاندارد رمزگذاریِ پیشرفته**[1]) دعوت شدند. قوانین مسابقه عبارت بودند از:

۱. الگوریتم می‌بایستی یک رمز بلوکیِ متقارن باشد.
۲. طرح کلی می‌بایستی قابل دسترسی عمومی باشد.
۳. کلیدهای به طول ۱۲۸، ۱۹۲، و ۲۵۶ می‌بایستی حمایت شوند.
۴. هم پیاده‌سازی نرم‌افزاری و هم پیاده‌سازی سخت‌افزاری می‌بایستی امکان‌پذیر باشند.
۵. الگوریتم می‌بایستی قابل دسترسی عمومی باشد یا آن‌که دسترسی بر اساس مجوزی که به شکل غیرتبعیض‌آمیز داده می‌شود، انجام‌پذیر باشد.

پانزده طرح جدی ارائه شدند و کنفرانس‌های عمومی‌ای برگزار گردید که این طرح‌ها در آن‌ها ارائه شدند و از حاضران در جلسات خواسته شد تا اشکالات این طرح‌ها را پیدا کنند. در آگوست ۱۹۹۸، NIST پنج فینالیست، عمدتاً بر مبنای امنیت، کارایی، سادگی، انعطاف‌پذیری، و نیازمندی‌های حافظه (که برای سیستم‌های توکار اهمیت دارد) را انتخاب نمود. کنفرانس‌های بیشتری برگزار شدند و هدف‌گذاری‌های بیشتری انجام شد.

در اکتبر سال ۲۰۰۰، موسسه‌ی NIST اعلام کرد که ریندال (Rijndael) (ارائه شده توسط Joan Daeman و Vincent Rijmen) برگزیده شده است. نام ریندال از اسامی مؤلفان آن گرفته شده است: Daemen + Rijman. در نوامبر ۲۰۰۱ ریندال تبدیل به استاندارد AES دولت ایالات متحده شد و با عنوان FIPS[2] و به شماره‌ی ۱۹۷ ثبت گردید. از آن‌جا که رقابت بسیار آشکار و بدون پنهان‌کاری بود، ویژگی‌های فنی ریندال و این حقیقت که تیم برنده از دو رمزنگار جوان بلژیکی تشکیل شده بود (که بعید بود فقط برای رضایتِ خاطر NSA وادار به ساخت این طرح شده باشند)، باعث شد که ریندال تبدیل به روش رمزنگاریِ برتر در جهان شود. رمزگذاری و رمزبَرداری AES اکنون بخشی از مجموعه‌ی دستوالعمل برای بعضی از ریزپردازنده‌ها (مانند اینتل) است.

ریندال طول کلیدها و اندازه‌ی بلوک از ۱۲۸ بیت تا ۲۵۶ بیت در گام‌های ۳۲ ـ بیتی را حمایت می‌کند. طول کلید و طول بلوک می‌توانند مستقلاً انتخاب شوند. با این وجود، AES مشخص می‌کند که اندازه‌ی بلوک بایستی ۱۲۸ بیت و طول کلید بایستی ۱۲۸، ۱۹۲، یا ۲۵۶ بیت باشد. معلوم نیست آیا کسی کلیدهای ۱۹۲ـ بیتی استفاده کند یا خیر، لذا عملاً AES دو گزینه دارد: یک بلوک ۱۲۸ ـ بیتی با یک کلید ۱۲۸ـ بیتی و یک بلوک ۱۲۸ـ بیتی با یک کلید ۲۵۶ـ بیتی.

در برخوردمان با الگوریتم، فقط حالت ۱۲۸/۱۲۸ را بررسی می‌کنیم زیرا به احتمال زیاد این مورد جنبه‌ی تجاری خواهد یافت. یک کلید ۱۲۸ـ بیتی، یک فضای $2^{128} \approx 3 \times 10^{38}$ ای برای کلیدها

1. Advanced Encryption Standard
۲. Federal Information Processing Standard : "استاندارد پردازش اطلاعات فدرال"

را در اختیارمان قرار می‌دهد. حتی اگر NSA ساخت ماشینی با یک میلیارد پردازنده‌ی موازی را مدیریت کند، به طوری‌که هر پردازنده در هر ثانیه توانایی ارزیابی یک کلید را داشته باشد، در حدود 10^{10} سال طول می‌کشد تا چنین فضای کلیدی تماماً جستجو شود. تا آن زمان خورشید خاموش شده و مردم بایستی نتایجی که بدست آمده را در نور شمع بخوانند!

رینداِل

از بُعد ریاضی، رینداِل بر مبنای نظریه‌ی گالویس[1] استوار است و به این ترتیب ویژگی‌های امنیتیِ آن قابل اثبات است. البته بدون ورود به بحث ریاضیات، می‌توان رینداِل را به عنوان کدی به زبان C نیز در نظر گرفت.

رینداِل نیز همانند DES از جایگزین‌سازی و جابجاسازی استفاده می‌کند ولی چندین راند را به کار می‌برد. تعداد راندها به اندازه‌ی کلید و اندازه‌ی بلوک بستگی دارد. تعداد راندها برای کلیدهای ۱۲۸ـ بیتی با بلوک‌های ۱۲۸ـ بیتی، ۱۰ است و به ازای بزرگ‌ترین کلید و بزرگ‌ترین بلوک تا ۱۴ هم می‌رسد. اما برخلاف DES، برای آن‌که پیاده‌سازی هم در سخت‌افزار و هم در نرم‌افزار کارآمد باشد، تمام عملیات بر روی تمام بایت‌ها اِعمال می‌شوند. یک طرح کلی از کد در شکل ۸-۹ داده شده است. توجه داشته باشید که این کد برای نمونه است. پیاده‌سازی‌های خوب در ارتباط با کد امنیتی، کارهای بیشتری انجام می‌دهند مثلاً صفر کردنِ حافظه‌ی حساس (sensitive memory) بعد از آن‌که مورد استفاده قرار می‌گیرد. برای مثال‌هایی در این باره به Ferguson و همکاران (۲۰۱۰) مراجعه نمایید.

```
#define LENGTH 16                                   /* # bytes in data block or key */
#define NROWS 4                                     /* number of rows in state */
#define NCOLS 4                                     /* number of columns in state */
#define ROUNDS 10                                   /* number of iterations */
typedef unsigned char byte;                         /* unsigned 8-bit integer */

rijndael(byte plaintext[LENGTH], byte ciphertext[LENGTH], byte key[LENGTH])
{
  int r;                                            /* loop index */
  byte state[NROWS][NCOLS];                         /* current state */
  struct {byte k[NROWS][NCOLS];} rk[ROUNDS + 1];    /* round keys */

  expand_key(key, rk);                              /* construct the round keys */
  copy_plaintext_to_state(state, plaintext);        /* init current state */
  xor_roundkey_into_state(state, rk[0]);            /* XOR key into state */

  for (r = 1; r <= ROUNDS; r++) {
      substitute(state);                            /* apply S-box to each byte */
      rotate_rows(state);                           /* rotate row i by i bytes */
      if (r < ROUNDS) mix_columns(state);           /* mix function */
      xor_roundkey_into_state(state, rk[r]);        /* XOR key into state */
  }
  copy_state_to_ciphertext(ciphertext, state);      /* return result */
}
```

شکل ۸-۹ طرحی کلی از رینداِل در C.

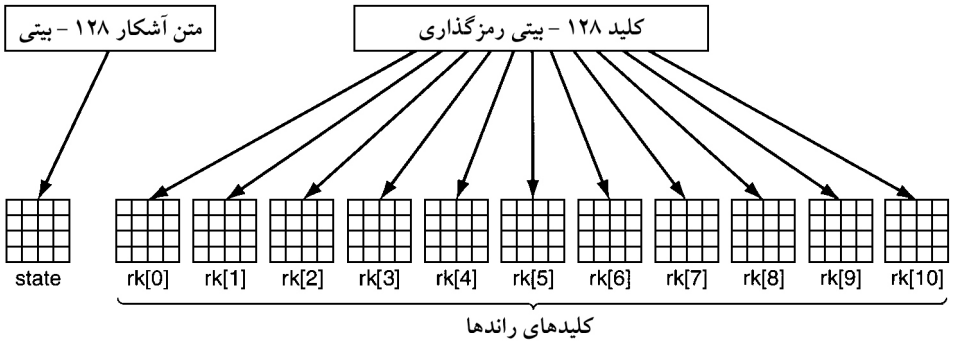
1. Galois

تابع *rijndael* سه پارامتر دارد. این پارامترها عبارتند از: *plaintext* که یک آرایه‌ی ۱۶ـ بایتی است و داده‌ی ورودی را در خود دارد؛ *ciphertext* که یک آرایه‌ی ۱۶ـ بایتی است وخروجیِ رمز شده را برخواهد گرداند؛ و *key* که یک کلید ۱۶ـ بایتی است. در حین محاسبات، حالتِ فعلی داده در یک آرایه از نوع بایت که در این‌جا *state* نامیده شده، نگهداری می‌شود. اندازه‌ی *state* برابر است با *NCOLS* × *NROWS*. اندازه‌ی این آرایه برای بلوک‌های ۱۲۸ـ بایتی برابر با ۴ × ۴ بایت می‌شود. با ۱۶ بایت، یک بلوک داده‌ی کاملِ ۱۲۸ـ بیتی می‌تواند ذخیره شود.

مقدار اولیه‌ی آرایه‌ی *state* همان متن آشکار است که در هر گام مقدارش ویرایش می‌شود. در بعضی گام‌ها، جایگزین‌سازیِ بایت ـ به ازای ـ بایت انجام می‌شود. در گام‌های دیگر، بایت‌ها درون آرایه پس و پیش می‌شوند. از تبدیلات دیگری نیز استفاده می‌شود. در انتهای کار، محتواهای *state* به عنوان متن رمزی برگردانده می‌شوند.

کدی که در شکل ۸–۹ آمده، با گسترش کلید درون ۱۱ آرایه با اندازه‌ی مساویِ *state* شروع می‌شود. این آرایه‌ها در *rk* ذخیره می‌شوند، که آرایه‌ای از ساختارهاست (منظور از "ساختارها" نوع داده‌ی struct در زبان C است. مترجم.) به طوری که هر کدام دربردارنده‌ی یک آرایه‌ی حالت می‌باشد. یکی از آن‌ها در شروع محاسبه استفاده می‌شود و ۱۰ تای دیگر در طی ۱۰ راندِ الگوریتم (یک راند در هر ثانیه) استفاده می‌شوند. محاسبه‌ی این راندها پیچیده‌تر از آن است که در این‌جا به آن بپردازیم. کافی است بگوییم که راند مربوط به کلیدها با چرخش تکراری و نیز XOR کردنِ گروه‌های مختلفی از بیت‌های کلید انجام می‌شود. برای جزئیات این مطلب به Daemen و Rijmen (۲۰۰۲) مراجعه نمایید.

گام بعدی عبارت‌است‌از کپی کردن متن آشکار به داخل آرایه‌ی *state* تا در خلال راندها بتواند پردازش شود. عمل کپی به ترتیب ستونی انجام می‌شود به این صورت که اولین ۴ بایت به ستون صفر می‌روند، ۴ بایت بعدی به ستون ۱ می‌روند، و همین‌طور تا آخر ادامه می‌یابد. هم ستون‌ها و هم سطرها از صفر شماره‌گذاری می‌شوند، درحالی‌که شماره‌گذاری راندها از ۱ است. این تنظیم اولیه برای آرایه‌های ۱۲ـ بایتی به ابعاد ۴ × ۴، در شکل ۸–۱۰ نشان داده شده است.



شکل ۸–۱۰ ایجاد آرایه‌های *state* و *rk*.

قبل از آن‌که محاسبه‌ی اصلی آغاز شود دو مرحله‌ی دیگر هم وجود دارند: [0]*rk* با *state* به صورت بایت به بایت XOR می‌شود. یعنی هر کدام از ۱۶ بایت در *state* با XOR خودش و بایتِ متناظرش در [0]*rk* جایگزین می‌شود.

اکنون نوبت قسمتِ جالب ماجراست. حلقه به اندازه‌ی ۱۰ تکرار اجرا می‌شود (یک تکرار به ازای هر راند) و *state* را در هر تکرار، تغییر می‌دهد. محتواهای هر راند در چهار مرحله تولید می‌شوند. مرحله‌ی ۱ یک جایگزین‌سازیِ بایت ـ به ازای ـ بایت بر روی *state* انجام می‌دهد. هر بایت به نوبه‌ی خود به عنوان ایندکس (شاخص) به یک جعبه‌ی S به کار می‌رود تا مقدار آن را با محتواهای درایه‌ی آن جعبه‌ی S جایگزین سازد. این مرحله یک رمز جایگزین‌سازی عادی (straight) تک ـ الفبایی (monoaphabetic) است. برخلاف DES که چندین جعبه‌ی S دارد، ریندال فقط یک جعبه‌ی S دارد.

گام ۲ هر کدام از چهار سطر را به سمت چپ می‌چرخاند. سطر صفر، صفر بایت می‌چرخد (به عبارت دیگر، تغییری نمی‌کند)، سطر ۱ یک بایت می‌چرخد، سطر ۲ دوبایت می‌چرخد، و سطر ۳ سه بایت می‌چرخد. این گام، در مقام مقایسه با جایگشت‌های شکل ۸-۶، محتوای داده‌ی جاری را در اطراف بلوک پخش می‌کند.

گام ۳ هر ستون را مستقل از بقیه‌ی ستون‌ها، مخلوط می‌کند. عمل اخطلاط با استفاده از ضرب ماتریسی انجام می‌شود. در این عمل، ستون جدید عبارت‌است‌از حاصل‌ضرب ستون قدیم و یک ماتریسِ ثابت. عمل ضرب با استفاده از فیلد متناهیِ گالویس انجام می‌شود، یعنی $GF(2^8)$. گرچه این روش ممکن است به نظر پیچیده باشد، ولی الگوریتمی وجود دارد که اجازه می‌دهد هر عنصر از ستون جدید، با استفاده از دو جستجوی جدولی و سه XOR محاسبه گردد (Daemen و Rijmen، ۲۰۰۲، پیوست E).

در انتها، گام ۴ کلیدی که مربوط به این راند است را در آرایه‌ی *state* (برای استفاده در راند بعدی) XOR می‌کند.

از آن‌جا که هر یک از گام‌ها برگشت‌پذیر هستند، عمل رمزبَرداری فقط با اجرای الگوریتم به صورت معکوس، امکان‌پذیر می‌باشد. با این وصف ترفندی در این‌جا وجود دارد: عمل رمزبَرداری می‌تواند با اجرای الگوریتم رمزگذاری ولی با استفاده از جداول مختلف انجام شود.

این الگوریتم نه تنها با هدف دستیابی به امنیت در سطح بالا طراحی شده، بلکه هدف از آن، سرعت بالا نیز می‌باشد. یک پیاده‌سازیِ نرم‌افزاری خوب بر روی یک ماشین 2-GHz ای بایستی قابلیت دستیابی به نرخ رمزگذاریِ برابر با 700 Mbps را داشته باشد. این نرخ برای رمزگذاریِ بیشتر از ۱۰۰ ویدیو از نوع MPEG-2 به صورت بی‌درنگ، کافی است. پیاده‌سازی‌های سخت‌افزاری از این هم سریع‌ترند.

۸-۲-۳ مودهای رمز

علی‌رغم تمام این پیچیدگی، AES (یا DES، یا هر رمزِ بلوکی‌ای که مرتبط با این بحث است) اساساً یک رمز جایگزین‌سازیِ تک ـ حرفی است که از کاراکترهای بزرگ استفاده می‌کند (کاراکترهای ۱۲۸ـ

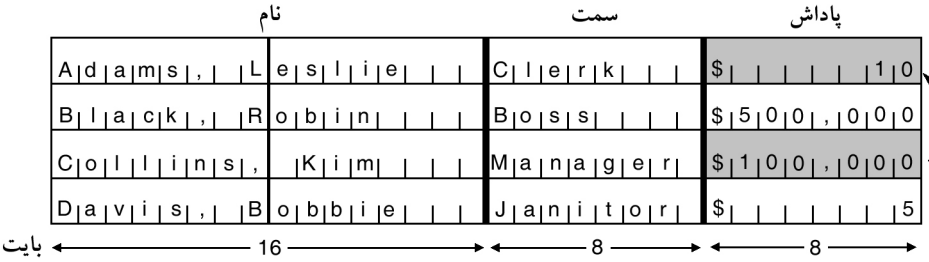
بیتی برای AES و کاراکترهای ۶۴ ـ بیتی برای DES). هر وقت یک بلوک از متن آشکارِ مشخص به سیستم داده شود، یک بلوک از متن رمزیِ مشخص هم از سیستم حاصل می‌شود. اگر متن آشکارِ *abcdefgh* را ۱۰۰ بار با یک کلید DES رمزگذاری کنید، هر ۱۰۰ بار همان متن رمزی را دریافت خواهید کرد. یک نفوذگر می‌تواند از این ویژگی استفاده کرده و رمز را تخریب کند.

مود مربوط به کتاب کد الکترونیکی

برای آن‌که ببینیم چگونه می‌توان از خاصیت تک ـ حرفیِ رمز جایگزین‌سازیِ تک ـ حرفی استفاده کرده و تا حدودی رمز را بی‌اثر نمود، از DES (سه‌گانه) استفاده خواهیم کرد زیرا نشان دادنِ بلوک‌های ۶۴ـ بیتی آسان‌تر از بلوک‌های ۱۲۸ـ بیتی است، اما AES نیز دقیقاً همین مشکل را دارد. راه ساده برای استفاده از DES جهت رمزگذاری یک تکه‌ی طولانی از متن آشکار، عبارت‌است‌از شکستنِ متن به بلوک‌های پشت سر هم ۸ـ بایتی (یعنی ۶۴ بیت) و این‌که آن‌ها را یکی بعد از دیگری، با یک کلیدِ یکسان رمزگذاری کنیم. آخرین تکه از متن آشکار، در صورت ضرورت، تا ۶۴ بیت بسط می‌یابد. این روش با عنوان **مود ECB** (**مود مربوط به کتاب کد الکترونیکی**[1]) شناخته می‌شود. این نامگذاری در مقایسه با کتاب‌های کد قدیمی است. در این کتاب‌ها فهرستی از کلمات متن آشکار به همراه متن رمزیِ آن‌ها (که معمولاً یک عدد ۵ـ رقمی بود) قرار داشت.

در شکل ۸-۱۱ ابتدای یک فهرست کامپیوتری از پاداش سالانه‌ی کارکنان یک شرکت قرار دارد. شرکت قصد دارد کارکنان را از پاداش سالانه‌شان مطلع سازد. این فایل از رکوردهای متوالیِ ۳۲ـ بایتی تشکیل می‌شود. فرمت رکوردها عبارت‌است‌از: ۱۶ بایت نام، ۸ بایت سِمت، و ۸ بایت مقدار پاداش. هر کدام از ۱۶ بلوکِ ۸ـ بایتی (که از ۰ تا ۱۵ شماره‌گذاری شده‌اند) توسط DES (سه‌گانه) رمزگذاری می‌شوند.

لسلی[2] به تازگی دعوایی با رئیس داشته و انتظار پاداش زیادی را ندارد. برعکس، کیم[3] محبوب رئیس است و همه هم این موضوع را می‌دانند. لسلی می‌تواند بعد از رمزگذاریِ فایل و قبل از ارسال آن به بانک، به آن دسترسی پیدا کند. آیا لسلی خواهد توانست این وضع غیرعادلانه را فقط با گرفتن فایلِ رمزگذاری شده، جبران کند؟



شکل ۸-۱۱ متن آشکار از فایلی که به صورت ۱۶ بلوک DES رمزگذاری شده است.

1. Electronic Code Book mode 2. Leslie 3. Kim

در کل، مشکلی وجود ندارد. تمام آنچه لسلی بایستی انجام دهد، گرفتنِ یک کپی از بلوک ۱۲ ام از متن رمزی می‌باشد (که شامل پاداش کیم است) و این‌که آن را با بلوک چهارمِ متن رمزی (که شامل پاداش لسلی است) عوض کند. لسلی حتی بدون آن‌که محتوای بلوک ۱۲ ام را بداند، می‌تواند حدس بزند که امسال کریسمس شادتری خواهد داشت. (یک امکان نیز کپی کردنِ بلوک هشتم از متن رمزی است که مبلغ پاداشش بیشتر است، اما خیلی احتمال دارد که موضوع کشف شود؛ به علاوه، لسلی چندان هم طماع نیست!)

مود زنجیر کردنِ بلوک رمزی

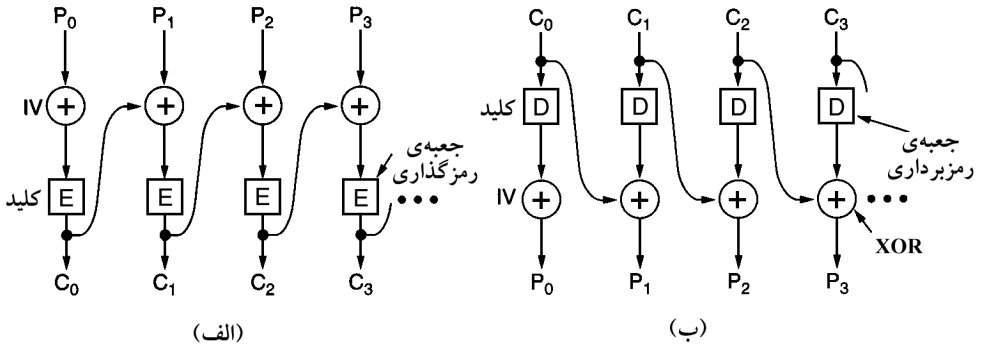
برای خنثی کردنِ این نوع حمله، تمام رمزهای بلوکی می‌توانند به روش‌های مختلفی به هم زنجیر شوند. به این ترتیب جایگزین کردنِ یک بلوک به روشی که لسلی انجام داد، سبب خواهد شد متن آشکارِ رمزبَرداری شده‌ای که با بلوکِ جایگزین شده، آغاز شده است، بی‌ارزش و دورانداختنی (garbage) شود. یک روش برای به هم زنجیر کردن، عبارت‌است‌از **زنجیر کردنِ بلوک رمزی**[1]. در این روش که در شکل ۸-۱۲ نشان داده شده، هر یک از بلوک‌های متن آشکار، قبل از آنکه رمزگذاری شود، با بلوک متن رمزیِ قبلی‌اش XOR می‌شود. در نتیجه یک بلوک از متن آشکار، هر بار به بلوک متن رمزیِ مشابه با بارِ قبلی نگاشت نمی‌شود و رمزگذاری، دیگر به صورت یک رمزِ جایگزین‌سازیِ تک ـ حرفیِ بزرگ نخواهد بود. اولین بلوک با یک بُردار **IV** (**بُردار راه‌اندازی**[2]) که به روش تصادفی انتخاب شده، XOR می‌شود. این بُردار (به صورت متن آشکار) همراه با متن رمزی منتقل می‌شود.

برای آن‌که نحوه‌ی کار در مود زنجیر کردنِ بلوک رمزی را ببینیم، مثال شکل ۸-۱۲ را بررسی می‌کنیم. بررسی را با محاسبه‌ی $C_0 = E(P_0 \text{ XOR } IV)$ شروع می‌کنیم. سپس $C_1 = E(P_1 \text{ XOR } C_0)$ را محاسبه می‌کنیم، و همین‌طور تا آخر ادامه می‌دهیم. عمل رمزبَرداری نیز برای انجام پروسه‌ی عکس، از XOR استفاده می‌کند به طوری که $P_0 = IV \text{ XOR } D(C_0)$ ، و همین‌طور تا آخر. توجه نمایید که رمزگذاری بلوک i تابعی است از تمام متن آشکار در بلوک‌های 0 تا i-1 ، بنابراین یک متن آشکار بر حسب آن‌که در کجا واقع شود متن رمزیِ متفاوتی را تولید می‌کند. یک عمل انتقال از همان نوعی که لسلی انجام داد، نتیجه‌اش بی‌معنی خواهد شد زیرا دو بلوک در عملیات دخالت دارند که شروع آن‌ها فیلد پاداش لسلی است. برای یک مأمور امنیتیِ تیزبین، این موردِ عجیب ممکن است عاملی باشد برای تحقیقات بیشتر.

زنجیر کردنِ بلوک رمزی این مزیت را هم دارد که حاصلِ یک بلوک متن آشکار، همیشه همان بلوک متن رمزی نخواهد بود، که نتیجه‌اش آن است که رمزیابی دشوارتر خواهد شد. در حقیقت، دلیل اصلیِ استفاده از این روش نیز همین موضوع است.

---

1. Cipher block chaining 2. Initialization Vector



شکل ۸-۱۲ زنجیر کردنِ بلوک رمزی. **(الف)** رمزگذاری. **(ب)** رمزبَرداری.

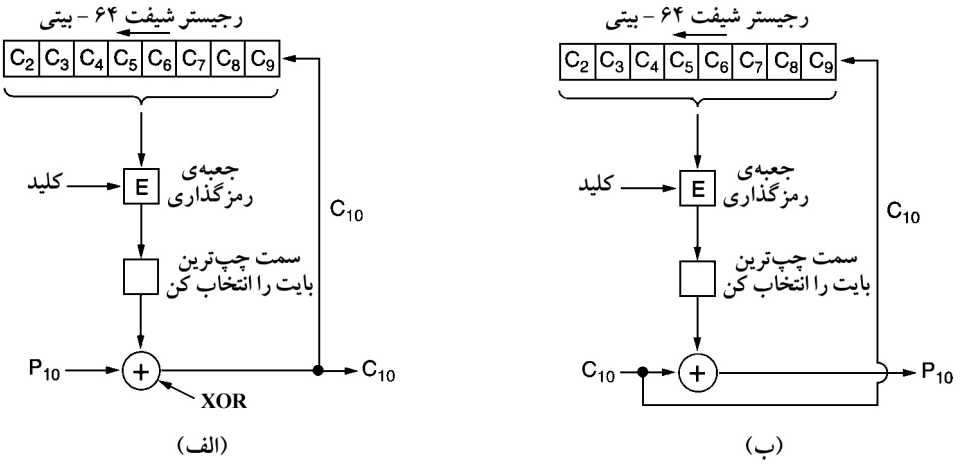
مود بازخوردِ رمز

با این وجود، زنجیر کردنِ بلوک رمزی این اِشکال را دارد که قبل از آن‌که رمزبَرداری شروع شود، لازم است کلِ یک بلوک ۶۴ـ بیتی برسد. همان‌طور که در شکل ۸-۱۳ نشان داده شده، برای رمزگذاریِ بایت ـ به ـ بایت، **مود بازخوردِ رمز**[1] که از DES (سه‌گانه) استفاده می‌کند، به کار می‌رود. برای AES نیز ایده عیناً به همین صورت است به جز آن‌که از یک رجیسترِ شیفت ۱۲۸ـ بیتی استفاده می‌شود. در این شکل، حالت ماشینِ رمزگذاری بعد از رمزگذاری و ارسالِ بایت ۰ تا بایت ۹، نشان داده می‌شود. هنگامی‌که بایتِ ۱۰ از متن آشکار می‌رسد، همان‌طور که در شکل ۸-۱۳(الف) نشان داده شده، الگوریتم DES بر روی رجیستر شیفتِ ۶۴ـ بیتی عمل می‌کند تا یک متن رمزی ۶۴ـ بیتی تولید کند. سمت‌چپ‌ترین بایت از متن رمزی استخراج شده و با P_{10} ، XOR می‌شود. این بایت بر روی خط انتقال، منتقل می‌شود. علاوه بر این، رجیسترِ شیفت به اندازه‌ی ۸ بیت به چپ شیفت پیدا می‌کند. این کار باعث می‌شود تا C_2 از انتهای سمت چپ بیرون بیفتد و C_{10} در محلِ انتهای سمت راست (که همین حالا با حرکتِ C_9 به سمت چپ، باز شده است) درج می‌شود.

توجه نمایید که محتوای رجیسترِ شیفت به کل تاریخچه‌ی متن آشکار وابسته است، بنابراین اگر الگویی چندین بار در متن آشکار تکرار شود، هر بار به شکلی متفاوت در متن رمزی رمزگذاری خواهد شد. همانند زنجیر کردنِ بلوک رمزی، در این‌جا نیز به یک بُردارِ راه‌اندازی برای آغاز کار نیاز داریم.

رمزبَرداری با مود بازخوردِ رمز نیز به همان روش رمزگذاری عمل می‌کند. مشخصاً، محتوای رجیسترِ شیفت *رمزگذاری* می‌شود، نه *رمزبَرداری*. به همین دلیل بایتی که برای XOR شدن با C_{10} (برای رسیدن به P_{10}) انتخاب می‌شود، همان بایتی است که ابتدای کار با P_{10} (برای رسیدن به C_{10}) XOR می‌شد. مادام که دو رجیستر شیفت مثل هم باشند، عمل رمزبَرداری درست کار می‌کند. این مورد در شکل ۸-۱۳(ب) نشان داده شده است.

1. Cipher feedback mode



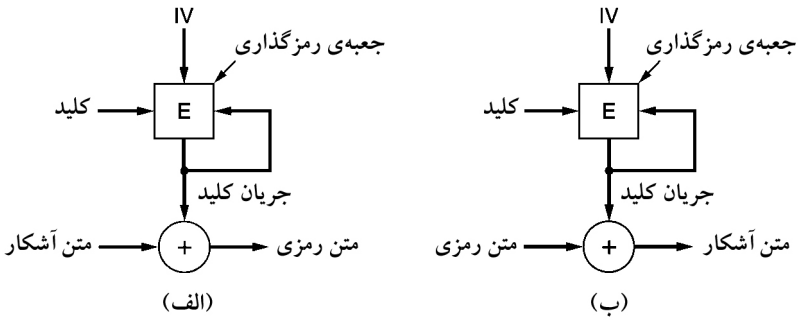
شکل ۸-۱۳ مود بازخوردِ رمز. **(الف)** رمزگذاری. **(ب)** رمزبَرداری.

مسئله‌ای که درباره‌ی مود بازخوردِ رمز وجود دارد آن است که اگر یک بیت از متن رمزی در حین انتقال به طور ناگهانی پس و پیش شود، در زمانی که بایتِ بد درون رجیستر شیفت قرار دارد، آن ۸ بایتی که رمزبَرداری شده‌اند نیز خراب خواهند شد. به محض آن‌که بایتِ بد از رجیستر شیفت بیرون انداخته شود، متن آشکاری که صحیح است مجدداً تولید خواهد شد. بنابراین اثرِ یک بیتِ منفرد که پس و پیش شده باشد، نسبتاً موضعی خواهد بود و بقیه‌ی پیغام را تباه نخواهد کرد. البته بیت‌هایی که درون رجیستر شیفت هستند خراب خواهند شد، یعنی محدوده‌ی تأثیرش به عرض رجیستر شیفت بستگی دارد.

مود رمزِ جریانی

کاربردهایی وجود دارند که حتی این تأثیر که خطای انتقال ۱ بیت، سبب دور ـ ریز شدنِ ۶۴ بیت از متن آشکار شوند نیز برایشان خیلی زیاد است. برای این کاربردها، گزینه‌ی چهارمی هم وجود دارد: **مود رمز جریانی**[1]. این مود با رمزگذاریِ یک بُردار راه‌انداز عمل می‌کند و از یک کلید برای رسیدن به یک بلوک خروجی استفاده می‌کند. سپس این بلوک خروجی رمزگذاری می‌شود و از کلید برای رسیدن به بلوک خروجی دوم استفاده می‌کند. بعد این بلوک رمزگذری می‌شود تا به بلوک سوم برسد، و همین‌طور کار ادامه می‌یابد. با سلسله بلوک‌های خروجی (که بزرگیِ آن‌ها به اندازه‌ی دلخواه است و **جریان کلید**[2] نامیده می‌شوند) همانند یک پدِ یک بارـ مصرف برخورد می‌شود و با متن آشکار XOR می‌شوند تا به متن رمزی برسیم (شکل ۸-۱۴(الف)). توجه داشته باشید که از IV (بُردار راه‌انداز) فقط در مرحله‌ی اول استفاده می‌شود. بعد از آن، این خروجی است که رمزگذاری می‌شود. همچنین توجه داشته باشید که جریان کلید از داده مستقل است، لذا می‌تواند (در صورتی که لازم باشد) از قبل محاسبه شود. ضمناً این روش کاملاً به خطاهای انتقال، غیرحساس است. شیوه‌ی رمزبَرداری در شکل ۸-۱۴(ب) نشان داده شده است.

1. Stream cipher mode 2. Keystream



شکل ۸-۱۴ یک رمز جریانی. **(الف)** رمزگذاری. **(ب)** رمزبَرداری.

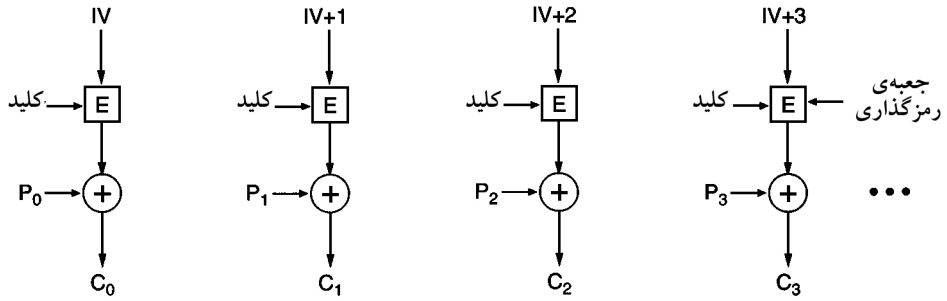
عمل رمزبَرداری به این صورت است که در طرفِ گیرنده همان جریانِ کلید تولید می‌شود. چون جریان کلید فقط به IV و به کلید وابسته است، از خطاهای انتقال در متن رمزی متأثر نمی‌شود. لذا یک خطای ۱ـ بیتی در متن رمزیِ منتقل شده، فقط یک خطای ۱ـ بیتی در متن آشکاری که رمزبَرداری شده است، تولید می‌کند.

خیلی مهم است که از یک زوجِ (کلید، IV)[1] برای یک رمز جریانی، دو بار استفاده نشود زیرا این کار باعث می‌شود که هر بار همان جریانِ کلید تولید شود. اگر دوبار از یک جریان کلید استفاده شود، متن رمزی در معرض **حمله‌ی استفاده‌ی مجدد از جریان کلید**[2] قرار می‌گیرد. فرض کنید که بلوک متن آشکار، یعنی P_0 ، با جریان کلید رمزگذاری شده و $P_0 \text{ XOR } K_0$ به دست آمده. بعداً یک بلوک متن آشکارِ دومی نیز به نام Q_0 ، با همان جریان کلید رمزگذاری و $Q_0 \text{ XOR } K_0$ حاصل شده است. نفوذگری که هر دوی این بلوک‌های متن رمزی را به دست آورده باشد، به سادگی می‌تواند آن‌ها را در هم XOR کند تا به $P_0 \text{ XOR } Q_0$ برسد. یعنی کلید را حذف کند. حالا نفوذگر XOR دو بلوک متن آشکار را در اختیار دارد. اگر یکی از آن‌ها معلوم شود یا بتواند حدس زده شود، آن دیگری هم معلوم خواهد شد. در هر صورت، XOR دو جریانِ متن آشکار می‌تواند با استفاده از ویژگی‌های آماری در پیغام‌ها، مورد حمله قرار گیرد. مثلاً در متون انگلیسی، رایج‌ترین کاراکتر در جریان، احتمالاً XOR دو فاصله‌ی خالی (space) خواهد بود، در وهله‌ی بعد XOR یک فاصله‌ی خالی و حرف "e" خواهد بود، و قس علیهذا. خلاصه آن‌که، اگر به XOR دو متن آشکار مجهز باشیم، در این صورت رمزیاب شانس زیادی دارد که هر دو متن را بتواند با حدس و گمان استنتاج کند.

مود شمارنده

مسئله‌ای که به جز مود کتاب کدِ الکترونیکی، همه‌ی مودهای دیگر به آن دچار هستند آن‌است‌که دستیابی تصادفی به داده‌ی رمزگذاری شده، غیرممکن است. به عنوان مثال فرض کنید یک فایل در حال انتقال از طریق یک شبکه است و سپس به همان شکلِ کد شده، بر روی دیسک ذخیره می‌شود. این

1. (key, IV) pair 2. Keystream reuse attack



شکل ۸–۱۵ رمزگذاری با استفاده از مودِ شمارنده.

کار در صورتی که کامپیوترِ دریافت‌کننده یک کامپیوتر نوت‌بوک باشد (که به راحتی می‌تواند سرقت شود)، ممکن است یک شیوه‌ی عملِ قابل قبول باشد. ذخیره کردنِ تمام فایل‌های مهم و حیاتی به شکل کدگذاری شده، سبب می‌شود اگر کامپیوتر به دست نااهلان بیفتد، شانس تخریب اطلاعاتِ سرّی بسیار اندک باشد.

با این وجود، فایل‌های روی دیسک همواره به صورت غیرترتیبی (nonsequential) مورد دستیابی قرار می‌گیرند، مخصوصاً فایل‌های موجود در پایگاه‌های داده. درصورتی‌که فایلی با استفاده از زنجیر کردنِ بلوکِ رمزی کدگذاری شده باشد، برای دسترسی به یک بلوک تصادفی ابتدا باید تمام بلوک‌های قبل از آن رمزبَرداری شوند، و این یعنی یک طراحی گران‌قیمت. به همین دلیل همان‌گونه که در شکل ۸–۱۵ نشان داده شده، مود دیگری نیز ابداع شده است: **مود شمارنده**[1]. در این‌جا، متن آشکار مستقیماً رمزگذاری نمی‌شود. به جای این کار، بُردارِ راه‌انداز به‌علاوه‌ی یک مقدارِ ثابت، رمزگذاری می‌شود و متن رمزیِ حاصل با متن آشکار XOR می‌شود. اگر به ازای هر بلوک جدید، شماره‌ی بُردارِ راه‌انداز را یکی به جلو ببریم، رمزگذاری کردنِ یک بلوک در هر نقطه‌ای از فایل کار آسانی خواهد شد و نیازی نیست ابتدا تمام بلوک‌های قبلیِ آن بلوک، رمزگذاری شوند.

گرچه مود شمارنده سودمند است، لیکن نقطه ضعفی نیز دارد که ارزش اشاره کردن را دارد. فرض کنید در آینده مجدداً از همان کلید K برای یک متن آشکارِ متفاوت (نسبت به متن قبلی) ولی با همان IV استفاده شود. همچنین فرض کنید که یک نفوذگر همه‌یِ متن رمزیِ مربوط به هر دو مورد را زیر نظر داشته باشد. در هر دو مورد جریان کلیدها مثل هم هستند. این کار باعث می‌شود که رمز در معرض یک حمله‌ی استفاده‌ی مجدد از جریان کلید قرار بگیرد. این حمله از همان نوع حمله است که در مورد رمزهای جریانی (بخش قبل) دیدیم. همه‌ی کاری که رمزیاب باید انجام دهد عبارت‌است‌از XOR کردنِ دو متن رمزی با یکدیگر تا تمام حفاظتی که رمزنگاری ایجاد کرده است، از میان برداشته شود و به XORِ متن‌های آشکار دست پیدا کند. این ضعف به معنیِ آن نیست که مود شمارنده ایده‌ی

1. Counter mode

بدی است بلکه فقط به این معناست که هم کلیدها و هم بُردارهای راه‌انداز بایستی جداگانه و به صورت تصادفی انتخاب شوند. حتی اگر تصادفاً از یک کلید دو بار استفاده شود، اگر هر بار IV تفاوت داشته باشد، باز هم متن آشکار در امان خواهد بود.

۸-۲-۴ رمزهای دیگر

روش AES (ریندال) و روش DES معروف‌ترین الگوریتم‌های رمزنگاریِ کلید ـ متقارن هستند، و همچنین گزینه‌های استاندارد در صنعت می‌باشند (حتی اگر تنها دلیلش، در امان بودن باشد). (اگر از AES در محصولتان استفاده کنید و رمز آن شکسته شود، کسی شما را سرزنش نخواهد کرد اما اگر از یک رمز غیراستاندارد استفاده کنید و بعداً این رمز شکسته شود، مطمئناً همه شما را سرزنش خواهند کرد.) با این وصف شایان ذکر است که رمزهای کلید ـ متقارنِ متعددی ابداع شده‌اند. بعضی از این رمزها درون محصولاتِ مختلف به صورت توکار قرار دارند. تعدادی از رایج‌ترینِ آن‌ها در شکل ۸-۱۶ فهرست شده‌اند. امکان استفاده از ترکیباتی از این رمزها نیز وجود دارد، برای مثال AES روی Twofish، به این ترتیب برای بازیابیِ داده باید هر دو رمز شکسته شوند.

۸-۲-۵ رمزیابی

قبل از آن‌که از بحث رمزنگاری کلید ـ متقارن خارج شویم، خوب است دست کم چهار محور توسعه (development) در رمزیابی را مورد توجه قرار دهیم. اولین محور توسعه **رمزیابیِ تفاضلی**[1] است (Biham و Shamir، ۱۹۹۷). از این روش می‌توان برای حمله به هر رمز بلوکی‌ای استفاده نمود. عملکرد آن به این صورت است که با یک جفت بلوک متن آشکار که فقط در یک تعداد کمِ بیت با هم تفاوت دارند کار را شروع می‌کند، و به دقت بررسی می‌کند که با پیشرویِ رمزگذاری در هر تکرارِ داخلی،

رمز	مولف	طول کلید	توضیحات
DES	IBM	۵۶ بیت	برای استفاده‌ی امروزی بسیار ضعیف است
RC4	رونالد ریوست (Ronald Rivest)	۱ تا ۲۰۴۸ بیت	احتیاط: بعضی کلیدها ضعیف هستند
RC5	رونالد ریوست (Ronald Rivest)	۱۲۸ تا ۲۵۶ بیت	خوب است، اما دارای حق انحصاری است
AES	دیمن و ریجمن (Daemen and Rijmen)	۱۲۸ تا ۲۵۶ بیت	بهترین گزینه
Serpent	اندرسون، بی‌هام، نودسن (Anderson. Biham, Knudsen)	۱۲۸ تا ۲۵۶ بیت	بسیار قوی
DES سه‌گانه	IBM	۱۶۸ بیت	خوب است، اما قدیمی شده
Twofish	بروس شنی‌یِر (Bruce Schneier)	۱۲۸ تا ۲۵۶ بیت	بسیار قوی؛ بسیار مورد استفاده است

شکل ۸-۱۶ بعضی الگوریتم‌های رمزنگاریِ کلید ـ متقارن متداول.

1. Differential cryptanalysis

چه اتفاقی می‌افتد. در بسیاری از موارد، بعضی از الگوهای بیتی نسبت به بقیه متداول‌ترند. این امر می‌تواند منجر به حمله‌های مبتنی بر احتمالات (probabilistic attacks) بشود.

دومین محور توسعه که توجه به آن سودمندمی‌باشد، **رمزیابی خطی**[1] است (Matsui، ۱۹۹۴). این رمزیاب فقط با 2^{43} متن آشکارِ شناخته شده می‌تواند DES را بشکند. عملکردِ آن عبارت‌است از XOR کردنِ بیت‌های خاصی از متنِ آشکار و متن رمزی با همدیگر و بررسی نتیجه‌ی این عمل. با انجامِ این عمل به صورت مکرر، نیمی از بیت‌ها بایستی 0 باشند و نیمی 1. اما غالباً رمزها گرایشی هر چند کوچک به یک سمت یا به سمتِ دیگر از خود نشان می‌دهند. این گرایش می‌تواند برای کاهش work factor به کار گرفته شود. برای جزئیات این مطلب به مقاله‌ی ماتسوئی[2] مراجعه نمایید.

سومین محور توسعه عبارت‌است‌از استفاده از اندازه‌گیریِ میزان مصرفِ توان الکتریکی برای یافتنِ کلیدهای سرّی. کامپیوترها معمولاً از تقریباً ۳ ولت برای نمایش بیت 1 و از صفر ولت برای نمایش بیت 0 استفاده می‌کنند. بنابراین پردازش یک 1 نسبت به پردازش یک 0 انرژی بیشتری صرف می‌کند. اگر یک الگوریتم رمزنگاری از حلقه‌ای تشکیل شده باشد که در آن حلقه، بیت‌ها به ترتیب پردازش شوند در این صورت یک مهاجم که کلاک اصلیِ n-GHz ای را با یک کلاک کُند (مثلاً 100- Hz ای) جایگزین کند و گیره‌های سوسماری[3] بر روی پین تغذیه‌ی CPU و پین زمین قرار دهد، می‌تواند توان مصرفی توسط هر یک از دستوالعمل‌های ماشین را به دقت پایش کند. استنتاج کردنِ کلید از این اطلاعات، بسیار آسان است. برای آن‌که این نوع از رمزیابی نقش بر آب شود، فقط بایستی کدکردنِ الگوریتم با دقت و به زبان اسمبلی انجام شود تا اطمینان داشته باشیم که مصرف توان مستقل از کلید و همچنین مستقل از همه‌ی کلیدهای راندِ مجزا، خواهد بود.

چهارمین محور توسعه عبارت‌است‌از تحلیل زمان‌بندی. الگوریتم‌های رمزگونه پُر از دستورات if ای هستند که بیت‌ها را در کلیدهای راند بررسی می‌کنند. اگر بخش‌های مربوط به then و else به لحاظ زمانی که صرف می‌کنند، با هم اختلاف داشته باشند، با کُند کردنِ کلاک و مشاهده‌ی طول مدتِ گام‌های مختلف، باز هم این امکان وجود خواهد داشت که کلیدهای راند را بتوان ردیابی و استنتاج کرد. با پیدا کردنِ تمام کلیدهای راند، معمولاً کلید اصلی را نیز می‌توان محاسبه نمود. ممکن است تحلیل‌های توان و زمان‌بندی به نظر عجیب و غریب باشند ولی عملاً روش‌هایی قدرتمند هستند که می‌توانند هر رمزی که طراحیِ غیرمقاوم در برابرِ آن‌ها دارند را بشکنند.

۸-۳ الگوریتم‌های کلید ـ عمومی

به لحاظ تاریخی، توزیعِ کلیدها همواره ضعیف‌ترین پیوند در اغلبِ سیستم‌های رمزنگاری[4] بوده است. صرفنظر از میزان قدرتِ یک سیستم رمزنگاری، اگر یک نفوذگر می‌توانست کلید را سرقت کند، نشان

1. Linear cryptanalysis 2. Matsui 3. Alligator clip 4. Cryptosystem

می‌داد که سیستم فاقد ارزش است. رمزیاب‌ها همواره مسلّم می‌دانستند که کلید رمزگذاری و کلید رمزبَرداری یکی هستند (یا آن‌که به سادگی از یکدیگر قابل استنتاج می‌باشند). اما کلید باید میان تمام کاربران سیستم توزیع می‌شد. بنابراین به نظر می‌رسید که یک مشکل اساسی وجود داشته باشد. باید کلیدها در برابر سرقت محافظت می‌شدند، اما در عین حال بایستی توزیع هم می‌شدند، پس نمی‌شد آن‌ها را در گاو صندوق بانک محبوس کرد.

در سال ۱۹۷۶ دو پژوهشگر در دانشگاه استنفورد به نام‌های دیفیو هِلمَن[1] یک نوع سیستم رمزنگاریِ جدید و کاملاً تحول‌آفرین را ارائه دادند. سیستمی که کلیدهای رمزگذاری و رمزبَرداری در آن چنان متفاوت بودند که کلید رمزبَرداری را نمی‌شد به سادگی از کلید رمزگذاری استنتاج کرد. در طرح آن‌ها، الگوریتم رمزگذاریِ (با کلید (keyed)) به نام E ، و الگوریتم رمزبَرداریِ (با کلید) به نام D، بایستی سه شرط را برآورده می‌کردند. این شرط‌ها را می‌توان به سادگی به صورت زیر بیان نمود:

۱. $D(E(P)) = P$.

۲. استنتاج کردنِ D از E فوق‌العاده دشوار است.

۳. E نمی‌تواند با حمله به یک متن آشکارِ انتخابی، شکسته شود.

اولین شرط می‌گوید اگر D را به یک پیغامِ رمزگذاری شده‌ی $E(P)$ اِعمال کنیم، پیغامِ اولیه‌ی متن آشکار (یعنی P) را بازخواهیم یافت. بدون این خصوصیت، گیرنده‌ی قانونیِ آن، قادر به رمزبَرداری از متن رمزی نخواهد بود. شرط دوم به خودی خود گویا است. علت ضرورتِ شرط سوم آن است که نفوذگران مایلند با یک متن دلخواه، الگوریتم را آزمایش کنند (که چگونگیِ آن را به زودی خواهیم دید). در چنین شرایطی دلیلی وجود ندارد که کلید رمزگذاری، نتواند عمومی باشد.

این روش چنین عمل می‌کند. شخصی، مثلاً آلیس، که می‌خواهد پیغام‌های سرّی دریافت کند، ابتدا دو الگوریتم طراحی می‌کند که شروط بالا را برآورده کنند. الگوریتم رمزگذاری و کلیدِ آلیس، عمومی می‌شوند و نام **رمزنگاری کلید ـعمومی**[2] نیز به همین دلیل است. مثلاً آلیس ممکن است کلید عمومی‌اش را در صفحه‌ی خانگی‌ای که در وب دارد قرار دهد. از علامت نوشتاریِ E_A به معنای "الگوریتم رمزگذاری‌ای که توسط کلید عمومیِ آلیس پارامتربندی شده است" استفاده می‌کنیم. به طور مشابه، الگوریتم رمزبَرداریِ (سرّی‌ای) که توسط کلید خصوصیِ آلیس پارامتربندی شده است، به نام D_A می‌باشد. باب نیز همین عمل را انجام می‌دهد، یعنی E_B را عمومی کرده ولی D_B را سرّی نگه می‌دارد.

اکنون ببینیم آیا می‌توانیم مسئله‌ی برقراریِ یک کانال امن مابین آلیس و باب که قبلاً هرگز تماسی با هم نداشته‌اند را حل کنیم یا خیر. فرض می‌شود که هم کلید رمزگذاری آلیس (E_A) و هم کلید رمزگذاری باب (E_B) فایل‌های عمومیِ قابل خواندن می‌باشند. حالا آلیس اولین پیغامش یعنی P را برمی‌دارد، $E_B(P)$ را محاسبه می‌کند و آن را به باب ارسال می‌نماید. سپس باب این پیغام را با اِعمال

1. Diffie و Hellman (۱۹۷۶) 2. Public-key cryptography

کردنِ کلید سرّی‌اش (D_B) رمزبَرداری می‌کند [به عبارت دیگر باب $D_B(E_B(P)) = P$ را محاسبه می‌کند]. هیچ‌کس دیگری قادر به خواندنِ پیغام رمزگذاری شده‌ی $E_B(P)$ نیست زیرا فرض شده که سیستم رمزگذاری قدرتمند است و به همین دلیل، استخراج D_B از E_B ای که به صورت عمومی شناخته شده است، بسیار دشوار می‌باشد. به منظور ارسال یک پاسخ مثل R، باب $E_A(R)$ را منتقل می‌کند. آلیس و باب حالا می‌توانند ارتباط امنی داشته باشند.

در این‌جا بهتر است به واژه‌شناسی توجه کنیم. در رمزنگاریِ کلید ـ عمومی لازم است که هر کاربر دو کلید داشته باشد: یک کلید عمومی که تمام دنیا برای رمزگذاری پیغام‌های ارسالی به آن کاربر از آن استفاده می‌کند، و یک کلید *خصوصی* که کاربر برای رمزبَرداری از پیغام‌ها به آن نیاز دارد. به این کلیدها با عنوان کلیدهای *عمومی* و *خصوصی* ارجاع خواهیم داد، و این کلیدها را از کلیدهای *سرّی* که برای رمزنگاریِ کلید ـ متقارنِ قراردادی به کار می‌روند، متمایز می‌کنیم.

RSA ۱-۳-۸

تنها مشکلی که داریم این است که نیاز به پیدا کردنِ الگوریتم‌هایی داریم که تمام سه شرط بالا را برآورده سازند. به واسطه‌ی مزایای بالقوه‌ای که رمزنگاریِ کلید ـ عمومی دارد، پژوهشگران متعددی به شدت روی آن کار می‌کنند و هم‌اکنون تعدادی الگوریتم نیز منتشر شده‌اند. یک گروه در دانشگاه M.I.T به یک روش خوب دست یافته‌اند (Rivest و همکاران، ۱۹۷۸). این روش به نام سه مبدع اولیه‌اش (یعنی Rivest، Shamir، و Adleman) شناخته می‌شود: **RSA** . این روش در برابر تمام حمله‌هایی که برای شکستنِ آن در طول بیش از ۳۰ سال انجام شده است، مقاومت کرده و به نظر بسیار قدرتمند می‌رسد. بخشِ اعظمِ امنیت عملاً بر اساس این الگوریتم پایه‌ریزی شده. به همین دلیل این سه نفر در سال ۲۰۰۲ جایزه‌ی تورینگ[1] را از آنِ خود کردند. اِشکال اصلیِ این روش آن است که به کلیدهایی با طول دست کم ۱۰۲۴ بیت نیاز دارد (در برابر ۱۲۸ بیتی که در الگوریتم‌های کلید ـ متقارن مورد نیاز است). این موضوع سبب کُندیِ زیاد این روش می‌شود.

روش RSA بر پایه‌ی اصولی از نظریه‌ی اعداد قرار دارد. در این‌جا نحوه‌ی استفاده از این روش را به طور خلاصه بیان می‌کنیم؛ برای جزئیات روش به مقاله‌ی مربوطه مراجعه نمایید.

۱. دو عدد اولِ بزرگ p و q انتخاب کنید (نوعاً ۱۰۲۴ بیتی).
۲. دو رابطه‌ی $n = p \times q$ و $z = (p-1) \times (q-1)$ را محاسبه کنید.
۳. یک عدد که نسبت به z اول باشد انتخاب کنید و آن را d بنامید.
۴. عدد e را چنان بیابید که $e \times d = 1 \bmod z$.

با این پارامترها که از قبل محاسبه شده‌اند، آماده‌ی شروع رمزگذاری هستیم. متن آشکار را (که به صورت یک رشته‌ی بیتی فرض شده) به بلوک‌هایی تقسیم کنید، به طوری که هر پیغام متن آشکار،

1. 2002 ACM Turing Award

به نام P ، در فاصله‌ی $0 \le P < n$ قرار بگیرد. برای انجام این کار، متن آشکار را به بلوک‌های k ـ بیتی گروه‌بندی کنید به طوری که k بزرگ‌ترین عدد صحیحی باشد که رابطه‌ی $2^k < n$ برای آن برقرار است.

جهت رمزگذاری یک پیغام مانند P، رابطه‌ی $C = P^e \pmod n$ را محاسبه کنید. برای رمزبَرداری از C، رابطه‌ی $P = C^d \pmod n$ را محاسبه کنید. می‌توان اثبات کرد که برای تمام P هایی که در یک رِنج مشخص باشند، توابع رمزگذاری و رمزبَرداری، معکوس همند. به منظور انجام رمزگذاری، نیاز به e و n دارید. به منظور انجام رمزبَرداری، نیاز به d و n دارید. بنابراین کلید عمومی شامل زوج (e , n) است و کلید خصوصی شامل زوج (d , n) است.

امنیت این روش مبتنی بر دشواری در مضرب‌گیری (یا فاکتورگیری factoring) از اعداد بزرگ است. اگر رمزیاب موفق شود از n (که به صورت عمومی شناخته شده است) مضرب بگیرد، p و q را هم می‌تواند پیدا کند، و با داشتنِ آن‌ها z هم پیدا می‌شود. وقتی به اطلاعاتِ مربوط به z و e مجهز باشد، آنگاه با استفاده از الگوریتم اقلیدس[1] d هم می‌تواند معلوم شود. خوشبختانه ریاضیدانان دست کم ۳۰۰ سال برای مضرب‌گیری از اعداد بزرگ تلاش کرده‌اند و تجاربِ آن‌ها نشان می‌دهد که این مسئله بینهایت دشوار است.

بر طبق نظریه‌ی ریوست و همکارانش، مضرب‌گیری از اعداد ۵۰۰ ـ رقمی نیاز به 10^{25} سال کار طاقت‌فرسا دارد. در هر دو حالت، آن‌ها بهترین الگوریتم موجود و یک کامپیوتر که زمان اجرای دستورالعمل در آن برابر با ۱ میکروثانیه باشد را فرض کردند. درصورتی‌که یک میلیون تراشه به صورت موازی اجرا شوند، به طوری که زمان اجرای دستورالعمل در هر تراشه برابر با ۱ نانوثانیه باشد، باز هم انجام این محاسبه 10^{16} سال طول خواهد کشید. حتی اگر کامپیوترها در هر ده سال، به اندازه‌ی ۱۰ برابر سریع‌تر شوند، باز هم مضرب‌گیری از یک عدد ۵۰۰ ـ رقمی سال‌های زیادی طول خواهد کشید، و تازه در آن هنگام نسل‌های بعد از ما می‌توانند به سادگی p و q را باز هم بزرگ‌تر انتخاب کنند.

یک مثال آموزشیِ معمولی از نحوه‌ی کار الگوریتم RSA در شکل ۸–۱۷ نشان داده شده است. برای این مثال، $p = 3$ و $q = 11$ انتخاب کرده‌ایم که $n = 33$ و $z = 20$ را نتیجه می‌دهد. یک مقدار مناسب برای d عبارت‌است‌از $d = 7$ چون 7 و 20 هیچ مضرب مشترکی ندارند. با این انتخاب‌ها، e را می‌تواند با حل کردنِ معادله‌ی $7e = 1 \pmod{20}$ پیدا کرد، که نتیجه‌ی این کار $e = 3$ می‌شود. متن رمزیِ C که متناظر با یک متن آشکارِ P باشد با رابطه‌ی $C = P^3 \pmod{33}$ به دست می‌آید. متن رمزی در دریافت کننده، با استفاده از قانونِ $P = C^7 \pmod{33}$ رمزبَرداری می‌شود. شکل، به عنوان یک مثال، رمزگذاری متن آشکار "SUZANNE" را نشان می‌دهد.

از آن‌جا که اعداد اولی که برای این مثال انتخاب شده‌اند بسیار کوچک هستند، P بایستی کمتر از ۳۳ باشد، لذا هر بلوک از متن آشکار می‌تواند فقط یک کاراکترِ منفرد داشته باشد. نتیجه‌ی کار عبارت‌است‌از یک رمز جایگزین‌سازیِ تک ـ حرفی، یعنی چندان نتیجه‌ی تأثیرگذاری نیست. اگر به جای

1. Euclid's algorithm

متن آشکار (P)			متن رمزی (C)		بعد از رمزبرداری	
نمادی	عددی	P^3	P^3 (mod 33)	C^7	C^7 (mod 33)	نمادی
S	19	6859	28	13492928512	19	S
U	21	9261	21	1801088541	21	U
Z	26	17576	20	1280000000	26	Z
A	01	1	1	1	01	A
N	14	2744	5	78125	14	N
N	14	2744	5	78125	14	N
E	05	125	26	8031810176	05	E
محاسبات ارسال‌کننده			محاسبات دریافت‌کننده			

شکل ۸-۱۷ یک مثال از الگوریتم RSA.

این کار، p و q را تقریباً در حدود 2^{512} انتخاب کرده بودیم، n را تقریباً برابر با 2^{1024} می‌داشتیم و به این ترتیب هر بلوک می‌توانست تا ۱۰۲۴ بیت یا ۱۲۸ کاراکترِ ۸ ـ بیتی باشد (مقایسه کنید با ۸ کاراکترِ DES و ۱۶ کاراکترِ AES).

بایستی به این نکته نیز اشاره شود که استفاده از RSA به شکلی که ما توضیح دادیم، مشابه استفاده از الگوریتم تقارن در مودِ ECB است ــ بلوک‌های ورودیِ یکسان به این دو الگوریتم، بلوک خروجیِ یکسانی را نتیجه می‌دهد. بنابراین به نوعی زنجیره برای رمزگذاری داده نیاز داریم. اما در عمل، اغلبِ سیستم‌های مبتنی بر RSA اساساً از رمزنگاری کلید ـ عمومی استفاده می‌کنند تا کلیدهای نشستِ یک بارـ مصرف (برای استفاده با یک الگوریتم کلید ـ متقارن، مانند AES یا DES سه‌گانه) را توزیع کنند. الگوریتم RSA برای رمزگذاری واقعی با حجم داده‌ی زیاد، بیش از حد کُند است ولی برای توزیع کلید خیلی مورد استفاده قرار می‌گیرد.

۸-۳-۲ سایر الگوریتم‌های کلید ـ عمومی

گرچه از RSA زیاد استفاده می‌شود، ولی به هیچ وجه تنها الگوریتم کلید ـعمومیِ شناخته شده، نیست. اولین الگوریتم کلید ـعمومی، الگوریتم نَپسَک[1] ("کوله‌پشتی") بود (Merkle و Hellman، ۱۹۷۸). در این الگوریتم ایده این است که شخصی مالک تعداد زیادی اشیا است که وزن آن‌ها با هم یکسان نیست. شخصِ مالک عمل کد کردنِ پیغام را با انتخابِ مخفیانه‌ی زیرمجموعه‌ای از اشیا و قرار دادنِ آن‌ها در کوله‌پشتی انجام می‌دهد. وزنِ کلیِ اشیایی که در کوله‌پشتی هستند، کلید عمومی را می‌سازد (براساس فهرست تمام اشیایِ احتمالی و وزنِ متناظر با هر کدام از آن‌ها). فهرست اشیایِ داخل کوله‌پشتی مخفی نگه داشته می‌شود. با اضافه کردنِ یک سری محدودیت‌های خاص، کشف یک فهرست احتمالی از اشیا همراه با وزن هر یک از آن‌ها، به نظر غیرممکن بوده و اساس الگوریتم کلید ـ عمومی را تشکیل می‌داد.

1. Knapsack

مبدع این الگوریتم، یعنی رالف مرکِل[1]، کاملاً اطمینان داشت که این الگوریتم نمی‌تواند شکسته شود. به همین دلیل برای هر کسی که بتواند این الگوریتم را بشکند یک پاداش ۱۰۰ دلاری تعیین کرد. آدی شامیر ("Adi Shamir" که معرّف حرف "S" در RSA است) بلافاصله این الگوریتم را شکست و پاداش را گرفت. مِرکل با عزمی راسخ الگوریتم را تقویت کرد و این بار یک پاداش ۱۰۰۰ دلاری برای شکستنِ آن تعیین کرد. رونالد ریوست ("Ronald Rivest" که معرف حرف "R" در RSA است) این الگوریتم جدید را شکست و پاداش را دریافت کرد. مِرکل دیگر پاداشی ۱۰۰۰۰ دلاری برای یک نگارش جدید اعلام نکرد، لذا "A" (یعنی لئونارد آدِلمن "Leonard Adleman") بدشانس بود. در هر حال، الگوریتم کوله‌پشتی به عنوان یک الگوریتم امن شناخته نشد و در عمل چندان مورد استفاده قرار نگرفت.

نظام‌های دیگری که روی بحث کلید ـ عمومی طراحی شدند، بر مبنای دشوار بودنِ محاسبه‌ی لگاریتم‌های گسسته[2] قرار داشتند. الگوریتم‌هایی که از این اصل استفاده می‌کنند توسط El Gamal (۱۹۸۵) و Schnorr (۱۹۹۱) ابداع شده‌اند.

چند نظام دیگر هم وجود دارند، مثلاً نظام‌هایی که بر مبنای منحنی‌های بیضوی قرار دارند (Menezes و Vanstone، ۱۹۹۳). اما دو رده‌ی اصلی آن‌هایی هستند که بر مبنای مضرب‌گیری از اعداد بزرگ و محاسبه‌ی لگاریتم‌های گسسته به پیمانه‌ی یک عدد اولِ بزرگ هستند. به نظر می‌رسد که حل این مسائل بسیار دشوار باشد ــ ریاضیدانان سال‌هاست که روی این مسائل کار می‌کنند، بدون آن‌که پیشرفت مهمی کرده باشند.

۸–۴ امضاهای دیجیتال

صحت قانونیِ بسیاری از اسناد رسمی، مالی، و اسناد دیگر، از طریق وجود یا عدم وجود یک امضای دستیِ تأیید شده، تعیین می‌گردد و فتوکپی‌ها اعتباری ندارند. برای آن‌که سیستم‌های پیغام به صورت کامپیوتری درآمده و جای حمل و نقل فیزیکیِ اسناد کاغذ ـ و ـ جوهری را بگیرند، باید روشی برقرار شود که امکان امضای اسناد را به طریقی غیرقابل جعل فراهم سازد.

مسئله‌ی ابداع یک جایگزین برای امضای دستی، مسئله‌ی مهمی است. در اصل آنچه مورد نیاز است، سیستمی است که از طریق آن یک طرف بتواند پیغامی امضا شده را به طرف دیگر ارسال کند به طوری که شرایط زیر برقرار باشند:

۱. دریافت کننده بتواند هویتِ ادعا شده از طرف ارسال کننده را راستی‌آزمایی کند.
۲. ارسال کننده نتواند محتوای پیغام را حاشا کند.
۳. دریافت کننده احتمالاً آن پیغام را از خودش درنیاورده باشد.

1. Ralph Merkle 2. Discrete logarithm

شرط اول به عنوان مثال، در سیستم‌های مالی ضرورت دارد. هنگامی‌که کامپیوتر یک مشتری به کامپیوتر یک بانک سفارش خرید یک تُن طلا می‌دهد، ضرورت دارد کامپیوتر بانک اطمینان حاصل کند که کامپیوترِ ارسال کننده واقعاً به مشتری‌ای تعلق دارد که هزینه‌ی مربوطه از حساب بانکی‌اش کسر می‌شود. به عبارت دیگر بانک باید صحت هویت مشتری را تصدیق کند (و مشتری نیز باید صحت هویت بانک را تأیید کند).

شرط دوم به منظور محافظت از بانک در برابر کلاهبرداری ضرورت دارد. فرض کنید بانک یک تُن طلا خریداری کند و بلافاصله بهای طلا به شدت سقوط کند. یک مشتری ریاکار ممکن است دعوایی بر علیه بانک اقامه کرده و مدعی شود که هرگز سفارشی برای خرید طلا نداده است. هنگامی‌که بانک پیغام را به دادگاه ارائه کند، ممکن است مشتری ارسال آن پیغام را حاشا کند. این خصوصیت که هیچ‌کدام از دو طرفِ یک تماس نتوانند بعداً امضایشان را حاشا کنند، **عدم انکار**[1] نامیده می‌شود. نظام‌های مربوط به امضای دیجیتال که اکنون مطالعه خواهیم کرد، در تأمین این خصوصیت موثر هستند.

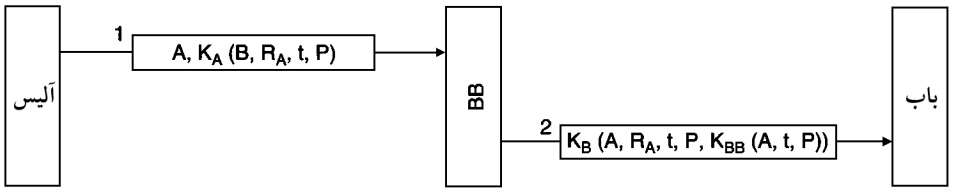
شرط سوم برای محافظت مشتری در مواقعی است که بهای طلا به سرعت افزایش می‌یابد و بانک سعی دارد یک پیغامِ امضادار بسازد که نشان دهد مشتری به جای یک تُن طلا، یک شمش طلا از بانک خواسته است. در این سناریوی کلاهبرداری، بانک بقیه‌ی طلا را برای خودش برمی‌دارد.

۸–۴–۱ امضاهای کلید ـ متقارن

یک رویکرد در ارتباط با امضاهای دیجیتال عبارت‌است‌از داشتنِ یک مرجع مرکزی که همه چیز را بداند و همه نیز به او اعتماد داشته باشند، مثلاً به نام "برادر بزرگ" (*BB*: Big Brother). به این ترتیب هر کاربر یک کلید سرّی انتخاب کرده و آن را به صورت دستی به دفتر کار *BB* می‌بَرد. بنابراین فقط آلیس و *BB* از کلید سرّیِ آلیس (K_A) اطلاع دارند.

هنگامی‌که آلیس قصد ارسال یک پیغام حاوی متنِ آشکاری که امضا شده است (مثلاً P) را به بانکدارش (یعنی باب) دارد، آلیس $K_A(B, R_A\, t, P)$ را تولید می‌کند. در این‌جا B شناسه‌ی باب است، R_A یک عدد تصادفی است که توسط آلیس انتخاب شده است، t یک برچسب زمانی است تا در مورد تازه بودنِ پیغام اطمینان حاصل شود، و $K_A(B, R_A\, t\, , P)$ پیغامِ رمزگذاری شده با کلید آلیس (یعنی K_A) است. آلیس پیغام را همان‌طور که در شکل ۸–۱۸ نشان داده شده، ارسال می‌کند. در این هنگام *BB* مشاهده می‌کند که پیغام از طرف آلیس است. پیغام را رمزبَرداری می‌کند و همان‌طور که نشان داده شده، آن را به باب ارسال می‌کند. پیغامی که به باب می‌رسد حاوی متن آشکارِ پیغام آلیس است، همچنین حاوی پیغامِ امضا شده‌ی $K_{BB}(A, t, P)$ می‌باشد. اکنون باب درخواست آلیس را اجرا می‌کند.

1. Nonrepudiation



شکل ۸–۱۸ امضاهای دیجیتال با "برادر بزرگ".

اگر آلیس بعداً ارسال پیغام را حاشا کند چه اتفاقی می‌افتد؟ مرحله‌ی ۱ به این ترتیب است که همه از هم شکایت می‌کنند (دست کم در ایالات متحده که این طور است). نهایتاً وقتی کار به دادگاه برسد و آلیس ارسال پیغامِ مورد بحث را قاطعانه تکذیب کند، قاضی از باب می‌خواهد ثابت کند که از کجا می‌تواند مطمئن باشد پیغامِ مورد بحث از طرف آلیس بوده و از طرف ترودی فرستاده نشده است. باب ابتدا به این نکته اشاره می‌کند که BB پیغامی را از آلیس نخواهد پذیرفت مگر آن‌که آن پیغام با K_A رمزگذاری شده باشد، لذا هیچ احتمالِ آن نمی‌رود که ترودی یک پیغام غلط از طرف آلیس به BB ارسال کند بدونِ آن‌که BB بلافاصله اینِ موضوع را کشف نماید. سپس باب با هیجان مدرک الف را ارائه می‌دهد، یعنی: $K_{BB}(A, t, P)$. باب می‌گوید این یک پیغامِ امضا شده توسط BB است که ثابت می‌کند آلیس پیغام P را به باب ارسال کرده. زمانی که BB شهادت دهد که باب حقیقت را می‌گوید، قاضی رأی را به نفع باب صادر می‌کند و پرونده مختومه می‌شود.

یک مسئله‌ی بالقوه که در ارتباط با پروتکلِ امضا در شکل ۸–۱۸ وجود دارد عبارت‌است‌از این‌که ترودی تمام پیغام‌ها را بازنواخت می‌کند (یعنی اصل شماره‌ی ۲ که در بخش ۸–۱–۵ گفتیم، خدشه‌دار می‌شود). برای آن‌که این مسئله را به حداقل برسانیم، در طول مسیر از برچسب‌های زمانی استفاده می‌شود. علاوه بر این، باب می‌تواند تمام پیغام‌های اخیر را کنترل کند و ببیند آیا در هیچ کدام از آن‌ها از R_A استفاده شده یا خیر. اگر استفاده شده باشد، چون یک پیغامِ بازنواخت است، حذف می‌شود. توجه داشته باشید که باب با توجه به برچسب زمانی، پیغام‌های خیلی قدیمی را رد خواهد کرد. برای محافظت در برابر حمله‌های بازنواختِ فوری (instant replay attacks)، باب فقط R_A مربوط به هر پیغامِ رسیده را کنترل می‌کند تا ببیند آیا چنین پیغامی در یک ساعتِ گذشته از طرف آلیس دریافت شده یا خیر. اگر دریافت نشده باشد، باب مطمئناً می‌تواند فرض کند که این درخواست، جدید است.

۸–۴–۲ امضاهای کلید ـ عمومی

یک مسئله‌ی ساختاری که در ارتباط با استفاده از رمزنگاری کلید ـ متقارن برای امضاهای دیجیتال مطرح می‌باشد، آن است که همه ناچارند به "برادر بزرگ" اعتماد کنند. علاوه بر این، "برادر بزرگ" مجبور است تمام پیغام‌های امضا شده را بخواند. منطقی‌ترین کاندیداها برای اجرای خدمتگزارِ "برادر بزرگ"، دولت، بانک‌ها، مؤسسات ذی‌حسابی، و مشاوران حقوقی می‌باشند. متأسفانه هیچ کدام از

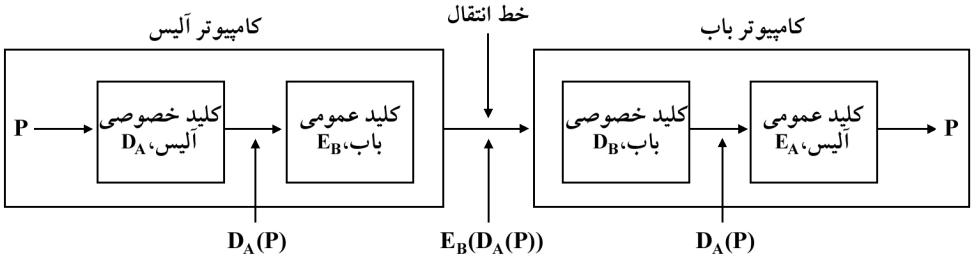
این‌ها اعتماد همه‌ی شهروندان را جلب نمی‌کنند. بنابراین خیلی خوب می‌شود اگر برای امضا کردنِ اسناد نیازی به یک مرجعِ قابل اعتماد نباشد.

خوشبختانه رمزنگاری کلید ـ عمومی می‌تواند در این حوزه کمک بزرگی بکند. بیایید فرض کنیم که رمزنگاری کلید ـ عمومی و الگوریتم‌های رمزبَرداری دارای ویژگی به صورت $E(D(P)) = P$ هستند، که البته علاوه بر ویژگیِ معمول آن‌ها یعنی $D(E(P)) = P$ می‌باشد. (الگوریتم RSA این ویژگی را دارد، لذا این فرض یک فرضِ دور از ذهن نیست.) با این فرض آلیس می‌تواند یک پیغامِ متن آشکار، یعنی P را با انتقال $E_B(D_A(P))$ ، به باب ارسال نماید. کاملاً دقت کنید که آلیس کلید (خصوصیِ) خودش، یعنی D_A را می‌داند، همان‌طور که کلید عمومی باب، یعنی E_B را می‌داند. بنابراین ساختنِ این پیغام از عهده‌ی آلیس برمی‌آید.

هنگامی‌که باب پیغام را دریافت می‌کند، همان‌طور که در شکل ۸-۱۹ نشان داده شده، آن را با کلید خصوصی‌اش تبدیل می‌کند تا به $D_A(P)$ برسد. باب این متن را در یک محل امن ذخیره می‌کند و سپس E_A را به آن اِعمال می‌کند تا به متن آشکار اصلی دست بیابد.

برای آن‌که نحوه‌ی عمل ویژگی امضا را متوجه شویم، فرض کنید آلیس بعداً ارسال پیغام P به باب را تکذیب می‌کند. وقتی این پرونده به دادگاه برسد، باب می‌تواند هم P و هم $D_A(P)$ را ارائه کند. قاضی می‌تواند به راحتی این موضوع را راستی‌آزمایی کند که باب حقیقتاً یک پیغام معتبر داشته که با D_A رمزگذاری شده بوده. این کار به سادگی با اِعمال E_A به آن، انجام می‌شود. چون باب کلید خصوصی آلیس را نمی‌داند، تنها راهی که باب می‌توانسته یک پیغامِ رمزگذاری شده با این کلید خصوصی را به دست آورده باشد آن است که آلیس واقعاً آن را ارسال کرده باشد. آلیس در زندان (به دلیل سوگند دروغ و کلاهبرداری) وقت زیادی خواهد داشت تا الگوریتم‌های کلید ـ عمومیِ جدید و جالبی را ابداع کند.

هر چند استفاده از رمزنگاریِ کلید ـ عمومی برای امضاهای دیجیتال، روشی جالب است ولی مسائلی هم وجود دارند که مربوط به محیطی هستند که در آن‌ها عمل می‌کنند (نسبت به الگوریتمِ مبنا). یکی آن‌که، باب مادام که D_A سرّی بماند می‌تواند ثابت کند که پیغام توسط آلیس ارسال شده بوده. اگر آلیس کلید خصوصی‌اش را فاش کند، این استدلال دیگر اثر ندارد زیرا هر کسی می‌توانسته پیغام را فرستاده باشد، حتی خودِ باب.



شکل ۸-۱۹ امضاهای دیجیتال با استفاده از رمزنگاری کلید ـ عمومی.

این مسئله مثلاً ممکن است زمانی بروز کند که باب کارگزارِ آلیس (در بازار بورس) باشد. فرض کنید آلیس به باب می‌گوید سهام یا اوراق قرضه‌ی به‌خصوصی را خریداری کند. بلافاصله بعد از این کار، سقوط ارزش با شیبی تند حادث می‌شود. آلیس با هدف تکذیب پیغامی که به باب ارسال کرده، با عجله نزد پلیس رفته و ادعا می‌کند منزلش مورد سرقت قرار گرفته و کامپیوتری که کلید او در آن بوده، دزدیده شده است. بر اساس قوانینِ جاری در ایالت یا کشورِ آلیس، ممکن است آلیس قانوناً مسئول شناخته شود یا شناخته نشود، مخصوصاً اگر ادعا کند تا زمان برگشت از محل کارش به منزل، از سرقت مطلع نشده و چند ساعت بعد از سرقت، موضوع را فهمیده است.

مسئلهٔ بعدی در ارتباط با نظام امضا آن است که اگر آلیس تصمیم بگیرد کلیدش را تغییر دهد، چه اتفاقی می‌افتد؟ روشن است که انجام این کار، قانونی است و احتمالاً تغییر کلید به صورت دوره‌ای ایده‌ی خوبی هم هست. اگر دادگاهی در این باره برپا شود، همان‌طور که در بالا گفتیم، قاضی E_A *جاری* را به $D_A(P)$ اِعمال خواهد کرد و می‌بیند که این کلید، P را تولید نمی‌کند. باب در این لحظهٔ بسیار ساده‌لوح به نظر خواهد رسید.

در اصل برای امضاهای دیجیتال از هر الگوریتم کلید ـ‌عمومی‌ای می‌توان استفاده نمود. استاندارد صنعتیِ واقعی، الگوریتم RSA است. بسیاری از محصولات امنیتی از آن استفاده می‌کنند. با این وجود در سالِ ۱۹۹۱، NIST استفاده از شکل دیگری از الگوریتم کلید ـ‌عمومی به نام الجمل[1] را برای **DSS** جدیدش (**استانداردِ امضای دیجیتال**[2]) پیشنهاد کرد. الگوریتم الجمل امن بودنش رااز دشواری در محاسبات الگوریتم‌های گسسته به دست می‌آورَد (به جای دشواری در مضرب‌گیری از اعداد بزرگ).

طبق معمول هر زمان دولت سعی می‌کند درباره‌ی استانداردهای رمزنگاری تصمیم‌گیری کند، غوغایی به پا می‌شود. استاندارد DSS به دلایل زیر مورد انتقاد بود:

۱. بیش از اندازه سرّی (NSA پروتکل را برای استفاده از الجمل طراحی کرد).
۲. بیش از اندازه کُند (۱۰ تا ۴۰ بار کُندتر از RSA در کنترل امضاها).
۳. بیش از اندازه جدید (الجمل هنوز مورد تحلیل کلی قرار نگرفته بود).
۴. بیش از اندازه ناامن (کلید ثابتِ ۵۱۲-بیتی).

در بازنگریِ بعدی، نکته‌ی چهارم از دور خارج شد چون کلیدهای تا ۱۰۲۴ بیت مُجاز شدند. معهذا دو نکته‌ی اول همچنان باقی هستند.

۸-۴-۳ چکیده‌های پیغام

یک نقدی که به روش‌های امضا وارد می‌شود آن است که این روش‌ها غالباً دو تابع مجزا را با هم کوپل می‌کنند: تصدیق هویت[3] و رازپوشی[4]. تصدیق هویت در اغلب مواقع ضروری است ولی رازپوشی همیشه ضروری نیست. ضمناً بیشتر اوقات اگر سیستمِ مورد بحث فقط تصدیق هویت در تجارت را

1. El Gamal 2. Digital Signature Standard 3. Authentication 4. Secrecy

تأمین کند ولی رازپوشی را خیر، در این صورت گرفتنِ یک مجوز صادرات آسان‌تر می‌شود. در زیر یک نظام تصدیق هویت را شرح خواهیم داد که نیازی به رمزگذاریِ کل پیغام ندارد.

این نظام بر پایه‌ی ایده‌ی تابع درهم‌سازِ یک ـ طرفه[1] استوار است، به این صورت که یک تکه‌ی طولانیِ دلخواه از متن آشکار برداشته می‌شود و از آن، یک رشته‌ی بیتی با طول ثابت محاسبه می‌گردد. این تابعِ درهم‌ساز (یعنی MD) غالباً **چکیده‌ی پیغام**[2] نامیده می‌شود و دارای چهار ویژگی مهم است:

۱. با داشتنِ P، محاسبه‌ی $MD(P)$ آسان می‌باشد.
۲. با داشتنِ $MD(P)$ عملاً پیدا کردنِ P غیرممکن است.
۳. با داشتنِ P هیچ کس قادر به یافتنِ P' ای نیست که $MD(P') = MD(P)$ باشد.
۴. تغییر در ورودی حتی در یک بیت از ورودی سبب تولید یک خروجی کاملاً متفاوت می‌شود.

جهت برآورده کردنِ ضابطه‌ی ۳ بایستی طول بخشِ درهم شده (hash) دست کم ۱۲۸ بیت و ترجیحاً بیشتر از ۱۲۸ بیت باشد. جهت برآورده کردنِ ضابطه‌ی ۴ بایستی تابعِ درهم‌ساز (hash function) کاملاً بیت‌ها را درهم و برهم کند، نه مثل آنچه که در الگوریتم رمزگذاریِ کلید ـ متقارن دیدیم. محاسبه‌ی یک چکیده‌ی پیغام از یک تکه‌ی متن آشکار خیلی سریع‌تر از رمزگذاریِ آن متن آشکار با یک الگوریتم کلید ـ عمومی است، بنابراین چکیده‌های پیغام می‌توانند جهت افزایش سرعت الگوریتم‌های امضای دیجیتال به کار روند. برای مشاهده‌ی چگونگیِ این مورد، دوباره پروتکل امضای شکل ۸–۱۸ را در نظر بگیرید. به جای امضا کردنِ P با $K_{BB}(A, t, P)$، در این‌جا BB چکیده‌ی پیغام را به وسیله‌ی اِعمال کردنِ MD به P محاسبه می‌کند، و $MD(P)$ به دست می‌آید. سپس BB به جای $K_{BB}(A, t, P)$، $K_{BB}(A, t, MD(P))$ را به عنوان فقره‌ی پنجم در فهرستِ رمزگذاری شده با K_B، ضمیمه می‌کند.

اگر دعوایی پیش بیاید، باب می‌تواند هم P و هم $K_{BB}(A, t, MD(P))$ را تولید کند. بعد از آن که "برادر بزرگ" آن را برای قاضی رمزبَرداری کند، باب $MD(P)$ را در اختیار دارد، که اعتبار و اصل بودنِ آن ضمانت شده است و همان P مورد ادعاست. در هر حال، از آنجا که برای باب کاملاً غیرممکن است که پیغام دیگری بیابد که همین نتیجه‌ی درهم شده را بدهد، لذا قاضی به آسانی متقاعد خواهد شد که باب حقیقت را می‌گوید. استفاده از چکیده‌های پیغام با این روش، هم باعث صرفه‌جویی در زمان رمزگذاری است و هم هزینه‌های حمل پیغام را کاهش می‌دهد.

همان‌طور که در شکل ۸–۲۰ نشان داده شده، چکیده‌های پیغام در سیستم‌های رمزگذاریِ کلید ـ عمومی نیز کار می‌کنند. در این‌جا آلیس ابتدا چکیده‌ی پیغام را برای متن آشکارش محاسبه می‌کند. سپس این چکیده‌ی پیغام را امضا کرده و هم چکیده‌ی امضا شده و هم متن آشکار را به باب ارسال می‌کند. اگر ترودی P را در مسیر عوض کند، باب هنگام محاسبه‌ی $MD(P)$ این موضوع را متوجه خواهد شد.

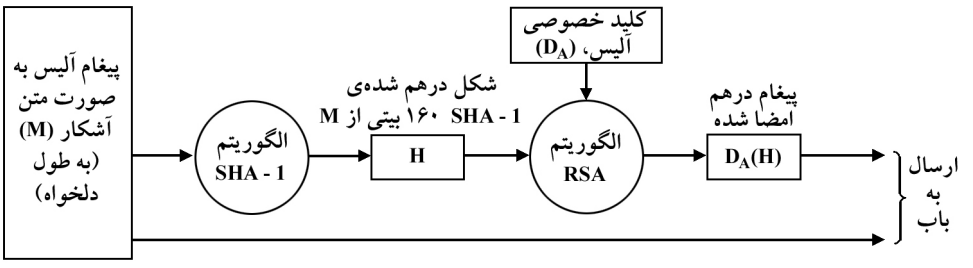
1. One-way hash function 2. Message digest

SHA-1 و SHA-2

شکل ۸–۲۰ امضاهای دیجیتال با استفاده از چکیده‌های پیغام.

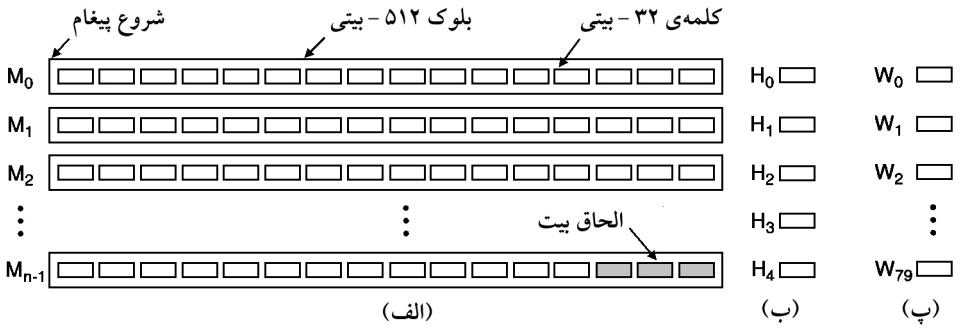
انواع توابع چکیده‌ی پیغام ایجاد شده‌اند. یکی از پُرکاربردترین آن‌ها **SHA-1** است (**درهم‌سازِ امن ـ الگوریتم ۱**[1]) (NIST، ۱۹۹۳). همانند تمام چکیده‌های پیغام، این روش نیز با درهم ریختنِ بیت‌ها عمل می‌کند، به نحوی که میزان پیچیدگی مناسب باشد و هر بیتِ خروجی متأثر از هر بیت ورودی باشد. روش SHA-1 توسط NSA توسعه یافته و از طرف NIST[2] در FIPS 180-1[3] معرفی شده است. این روش داده‌ی ورودی را در بلوک‌های ۵۱۲ ـ بیتی پردازش می‌کند و یک چکیده‌ی پیغام ۱۶۰ـ بیتی تولید می‌کند. یک روش برای آن‌که آلیس بتواند یک پیغام غیرسرّی ولی امضا شده به باب ارسال کند در شکل ۸–۲۱ نشان داده شده. در این روش، متن آشکارِ آلیس به الگوریتم SHA-1 داده می‌شود تا یک پیغام درهم شده‌ی SHA-1 ۱۶۰ـ بیتی به دست بیاید. آلیس سپس خروجیِ درهم شده را با کلید خصوصیِ RSA خودش امضا می‌کند، و هم پیغامِ متن آشکار و هم نتیجه‌ی درهمِ امضا شده را به باب ارسال می‌کند.

باب بعد از دریافت پیغام، خودش تابع درهم‌سازِ SHA-1 را محاسبه می‌کند و کلید عمومی آلیس را نیز به پیغام درهمِ امضا شده اِعمال می‌کند تا به پیغام درهمِ اولیه، یعنی H برسد. اگر این دو با هم تطبیق داشته باشند، این پیغام به عنوان یک پیغام معتبر شناخته می‌شود. از آن‌جا که برای ترودی هیچ امکانی وجود ندارد که پیغامِ متن آشکار را در حین انتقال پیغام دستکاری کرده و یک پیغام جدید تولید کند، سپس آن را درهم‌سازی کرده و درون H قرار دهد، لذا باب به سادگی می‌تواند هر تغییری که توسط ترودی در پیغام اِعمال شود را تشخیص دهد. برای پیغام‌هایی که جامعیتِ آن‌ها مهم است ولی محتوای آن‌ها سرّی نیست، از نظامی که در شکل ۸–۲۱ نمایش داده شده زیاد استفاده می‌شود. به ازای هزینه‌ای نسبتاً کوچک که بابت محاسبات می‌باشد، ضمانت می‌شود که هر گونه تغییراتی که در حین انتقال در پیغامِ متن آشکار ایجاد شوند، می‌تواند با احتمال بالایی کشف شود.



شکل ۸–۲۱ استفاده از SHA-1 و RSA برای امضا کردنِ پیغام‌های غیرسرّی.

1. Secure Hash Algorithm 1
2. National Institute of Standards and Technology (موسسهٔ ملی استانداردها و فناوری)
3. Federal Information Processing Standards Publication 180-1



شکل ۸-۲۲ (الف) یک پیغام که با اضافه کردنِ بیت، به مضربی از ۵۱۲ بیت رسانده شده است. (ب) متغیرهای خروجی. (پ) آرایهٔ کلمه.

اکنون نگاهی اجمالی به نحوه‌ی عملکرد SHA-1 می‌اندازیم. این الگوریتم در ابتدا با عمل اضافه کردنِ بیت[1] آغاز می‌شود، یعنی یک بیت 1 به انتهای پیغام اضافه می‌کند. سپس هر تعداد بیت 0 که لازم است اضافه می‌کند (دست کم ۶۴ تا) تا طول پیغام مضربی از ۵۱۲ بیت شود. سپس یک عدد ۶۴ ـ بیتی که شامل طول پیغام (قبل از اضافه کردنِ بیت) می‌باشد، با ۶۴ بیتِ مرتبه‌ی پایین OR می‌کند. در شکل ۸-۲۲ عمل اضافه کردنِ بیت به سمت راست پیغام نشان داده شده است زیرا متن انگلیسی و شکل‌ها از سمت چپ به راست هستند (به عبارت دیگر، گوشه‌ی پایین و راست معمولاً به عنوان انتهای شکل در نظر گرفته می‌شود). این محل در کامپیوترها عمدتاً متناظر با ماشین‌های بزرگ ـ در ـ انتها است (از قبیل SPARC و IBM 360 و نسل‌های بعدیِ آن‌ها). اما SHA-1 همیشه عمل اضافه کردنِ بیت به انتهای پیغام را انجام داده و کاری به این‌که بیتِ با مرتبه‌ی بزرگ در کدام طرف قرار داشته باشد، ندارد.

الگوریتم SHA-1 در حین محاسبات، پنج متغیر ۳۲ـ بیتی را نگه می‌دارد (از H_0 تا H_4) که نتیجه‌ی درهم‌سازی شده در آن‌ها انباشته می‌شود. این متغیرها در شکل ۸-۲۲(ب) نشان داده می‌شوند. مقادیر اولیه‌ای که در این متغیرها قرار می‌گیرند، مقادیر ثابتی هستند که طبق استاندارد مشخص شده‌اند.

اکنون هر یک از بلوک‌های M_0 تا M_{n-1} به نوبت پردازش می‌شوند. در ارتباط با بلوک جاری، همان‌طور که در شکل ۸-۲۲(پ) نشان داده شده، ابتدا ۱۶ کلمه به ابتدای یک آرایه‌ی ۸۰ ـ کلمه‌ایِ جانبی[2]، یعنی W، کپی می‌شوند. سپس بقیه‌ی ۶۴ کلمه‌ای که در W هستند با استفاده از فرمول زیر پُر می‌شوند:

$$W_i = S^1(W_{i-3} \text{ XOR } W_{i-8} \text{ XOR } W_{i-14} \text{ XOR } W_{i-16}) \quad (16 \le i \le 79)$$

در این فرمول $S^b(W)$ نشان دهنده‌ی چرخش دایره‌ایِ کلمه‌ی ۳۲ـ بیتی (W) به سمت چپ و به اندازه‌ی b بیت می‌باشد.حالا به پنج متغیر چرکنویس (A تا E) به ترتیب مقدار اولیه‌ی H_0 تا H_4 داده می‌شود.

1. Padding 2. Auxiliary

محاسبه‌ی واقعی می‌تواند به صورت زیر و با استفاده از شبه ـ C بیان شود:

```
for (i = 0; i < 80; i++) {
    temp = S^5(A) + f_i (B, C, D) + E + W_i + K_i;
    E = D;  D = C;  C = S^30(B);  B = A;  A = temp;
}
```

در این کد، ثابت‌های K_i طبق استاندارد تعریف می‌شوند. توابع مخلوط کنِ f_i به صورت زیر تعریف می‌شوند:

$$f_i(B,C,D) = (B \text{ AND } C) \text{ OR } (\text{NOT } B \text{ AND } D) \qquad (0 \le i \le 19)$$
$$f_i(B,C,D) = B \text{ XOR } C \text{ XOR } D \qquad (20 \le i \le 39)$$
$$f_i(B,C,D) = (B \text{ AND } C) \text{ OR } (B \text{ AND } D) \text{ OR } (C \text{ AND } D) \qquad (40 \le i \le 59)$$
$$f_i(B,C,D) = B \text{ XOR } C \text{ XOR } D \qquad (60 \le i \le 79)$$

هنگامی‌که تمام ۸۰ تکرارِ حلقه تکمیل گردد، A تا E به ترتیب به H_0 تا H_4 اضافه می‌شوند.

اکنون که اولین بلوک ۵۱۲ ـ بیتی پردازش شد، بلوک بعدی شروع می‌شود. آرایه‌ی W از بلوک جدید مجدداً مقداردهیِ اولیه می‌گردد، اما H همان مقداری که بوده باقی می‌مانَد. وقتی این بلوک تمام شد، بعدی شروع می‌شود، و کار به همین ترتیب ادامه می‌یابد تا همه‌ی بلوک‌های ۵۱۲ ـ بیتیِ پیغام کارشان به پایان برسد. هنگامی‌که پردازش آخرین بلوک نیز خاتمه یافت، پنج کلمه‌ی ۳۲ـ بیتی که در آرایه‌ی H هستند به صورت یک پیغام درهم‌سازی شده‌ی رمزیِ ۱۶۰ـ بیتی، به عنوان خروجی داده می‌شوند. کد کامل C برای SHA-1 در RFC 3174 داده شده است.

نگارش‌های جدیدی از SHA-1 توسعه یافته‌اند که پیغام‌های درهم‌سازی شده‌ای با طول ۲۲۴، ۲۵۶، ۳۸۴، و ۵۱۲ بیت ایجاد می‌کنند. این نگارش‌ها در مجموع SHA-2 نامیده می‌شوند. نه تنها این پیغام‌های درهم‌سازی شده نسبت به پیغام‌های درهم‌سازی شده‌ی SHA-1 طول بیشتری دارند، بلکه تابع چکیده نیز تغییر کرده است تا مانعی باشد در برابر بعضی ضعف‌های موجود در SHA-1 . هنوز از SHA-2 به صورت فراگیر استفاده نمی‌شود اما احتمالِ استفاده‌ی فراگیر از آن درآینده وجود دارد.

MD5

برای آن‌که بحثمان را کامل کنیم، چکیده‌ی رایج دیگری را بررسی خواهیم کرد. پنجمین سری از چکیده‌های پیغام MD5 است که به وسیله‌ی رونالد ریوست طراحی شده (Rivest، ۱۹۹۲). اگر بخواهیم خیلی خلاصه بگوییم، بیت‌هایی به پیغام اضافه می‌شوند تا طول پیغام به ۴۴۸ بیت برسد (پیمانه‌ی ۵۱۲). سپس طول اصلیِ پیغام به صورت یک عدد integer ۶۴ ـ بیتی به آن الحاق می‌شود تا طول کلیِ ورودی مضربی از ۵۱۲ بیت بشود. هر راند از محاسبات، یک بلوک ۵۱۲ ـ بیتی از ورودی را برداشته و آن را کاملاً با یک بافر ۱۲۸ـ بیتیِ در حال کار، مخلوط می‌کند. در عملیات مخلوط کردن از یک جدول که از تابع سینوسی ساخته شده است، استفاده می‌شود. نکته‌ای که در استفاده از یک تابع شناخته شده وجود دارد آن است که از هرگونه بدگمانی در مورد طراح جلوگیری می‌شود تا گمان

نشود طرح در پشت یک در بسته قرار دارد و هیچ کسی جز خود طراح قادر به عبور از آن در نیست. این پروسه تا هنگامی‌که تمام بلوک‌های ورودی مصرف شوند، ادامه دارد. محتوای بافر ۱۲۸ـ بیتی، چکیده‌ی پیغام را تشکیل می‌دهد.

بعد از گذشت بیشتر از یک دهه استفاده‌ی مداوم از این روش و مطالعه بر روی آن، ضعف‌های MD5 منتهی شده‌اند به توانایی پیدا کردنِ تصادم‌ها، یعنی پیغام‌های متفاوتی که پیغام‌های درهم شده‌ی آن‌ها مثل هم هستند (Sotirov و همکاران، ۲۰۰۸). این به منزله‌ی ناقوس مرگ برای تابع چکیده است زیرا معنایش آن است که چکیده نمی‌تواند به صورتی امن برای نمایش پیغام مورد استفاده قرار گیرد. به این ترتیب کمیته‌ی امنیت تشخیص داد که MD5 شکسته می‌شود؛ بنابراین MD5 در هر جا که امکانش بود می‌بایستی جایگزین می‌شد و هیچ سیستم جدیدی نیز نبایستی از MD5 به عنوان بخشی از طرحش استفاده می‌کرد. با این وجود هنوز هم می‌توانید شاهد استفاده از MD5 در سیستم‌های موجود باشید.

۸-۴-۴ حمله‌ی روز تولد

در دنیای رمزها، هیچ چیزی همانی نیست که به نظر می‌رسد. ممکن است فکر کنید برای فروپاشی یک چکیده‌ی پیغامِ m ـ بیتی، عملیاتی از مرتبه‌ی 2^m لازم است. در حقیقت اغلب مواقع از **حمله‌ی روز تولد**[1] استفاده می‌کنند و $2^{m/2}$ عملیات کافی است. رویکردی که توسط یووال[2] در سال ۱۹۷۹ و در مقاله‌ای از او به نام "چگونه رابین[3] را فریب بدهیم" (که اکنون یک مقاله‌ی کلاسیک محسوب می‌شود) به چاپ رسید.

ایده‌ی این حمله از روشی نشأت گرفته که اساتید ریاضی غالباً در درس احتمالات از آن استفاده می‌کنند. پرسش این است: قبل از آن‌که احتمالِ داشتنِ دو فرد با تاریخ تولد یکسان از 1/2 بیشتر شود، چند دانشجو در یک کلاس باید حضور داشته باشند؟ بیشتر دانشجویان انتظار دارند که جواب چیزی بیش از ۱۰۰ باشد. در حقیقت، تئوری احتمال عدد ۲۳ را می‌دهد. بدون انجام یک تحلیل موشکافانه و به طور حسی می‌بینیم که اگر تعداد افراد ۲۳ باشد، با آن‌ها می‌توان $253 = 2 / (22 \times 23)$ زوجِ متفاوت تشکیل داد به طوری که احتمال اصابت برای هر یک از آن‌ها برابر با 1/365 باشد.

در حالت کلی اگر نگاشتی میان ورودی‌ها و خروجی‌ها وجود داشته باشد، با n ورودی (افراد، پیغام‌ها، و امثال آن‌ها) و k خروجیِ احتمالی (روز تولد، چکیده‌های پیغام، و امثال آن‌ها)، تعداد زوج‌های ورودی برابر با $n(n-1)/2$ خواهد بود. اگر $n(n-1)/2 > k$ باشد، شانسِ این‌که دست کم یک تطبیق (match) داشته باشیم خیلی زیاد است. بنابراین تقریباً احتمالِ یک تطبیق به ازای $n > \sqrt{k}$ وجود دارد. این نتیجه به این معناست که یک چکیده‌ی پیغامِ ۶۴ ـ بیتیِ احتمالاً می‌تواند با تولید 2^{32} پیغام و جستجو برای یافتنِ دو پیغامی که چکیده‌ی پیغامشان یکسان باشد، شکسته شود.

---

1. Birthday attack 2. Yuval (1979)

۳. Rabin cryptosystem: یک روش رمزنگاریِ نامتقارن است که همانند RSA مبتنی بر دشوار بودنِ مضرب‌گیری می‌باشد (مترجم).

اجازه دهید یک مثال عملی را بررسی کنیم. دپارتمان علوم کامپیوتر در دانشگاه ایالتی یک محل برای یک عضو هیئت علمی رسمی و دو نامزد برای تصدی این سمت به نام‌های تام[۱] و دیک دارد. تام دو سال قبل از دیک استخدام شده است و به همین دلیل ابتدا تام برای بررسی برگزیده می‌شود. اگر تام این پُست را بگیرد، شانسی برای دیک وجود نخواهد داشت. تام اطلاع دارد که رئیس دپارتمان، یعنی مریلین، خیلی در فکرِ کار اوست لذا از او درخواست می‌کند تا توصیه ـ نامه‌ای به دین بنویسد. دین تصمیم‌گیرنده‌ی اصلی است. تمام نامه‌ها محرمانه هستند.

مریلین به منشی‌اش، اِلن، می‌گوید نامه‌ای به دین بنویسد و خواسته‌ی مریلین را در آن مطرح کند. وقتی نامه حاضر شد، مریلین آن را مرور کرده و چکیده‌ی ۶۴ ـ بیتی را محاسبه و امضا می‌نماید، سپس نامه را به دین ارسال می‌کند. اِلن بعداً می‌تواند نامه را از طریق ایمیل ارسال نماید.

برای تام متأسفیم چون اِلن نظر مساعدی نسبت به دین دارد، ضمناً می‌خواهد کار مربوط به تام را نیز انجام دهد، بنابراین نامه‌ی زیر را با ۳۲ گزینه‌ی داخل کروشه، می‌نویسد[۲]:

Dear Dean Smith,

This [*letter* | *message*] is to give my [*honest* | *frank*] opinion of Prof. Tom Wilson, who is [*a candidate* | *up*] for tenure [*now* | *this year*]. I have [*known* | *worked with*] Prof. Wilson for [*about* | *almost*] six years. He is an [*outstanding* | *excellent*] researcher of great [*talent* | *ability*] known [*worldwide* | *internationally*] for his [*brilliant* | *creative*] insights into [*many* | *a wide variety of*] [*difficult* | *challenging*] problems.

He is also a [*highly* | *greatly*] [*respected* | *admired*] [*teacher* | *educator*]. Hisstudents give his [*classes* | *courses*] [*rave* | *spectacular*] reviews. He is [*our* | *the Department's*] [*most popular* | *best-loved*] [*teacher* | *instructor*].

[*In addition* | *Additionally*] Prof. Wilson is a [*gifted* | *effective*] fund raiser. His [*grants* | *contracts*] have brought a [*large* | *substantial*] amount of money into [*the* | *our*] Department. [*This money has* | *These funds have*] [*enabled* | *permitted*] us to [*pursue* | *carry out*] many [*special* | *important*] programs, [*such as* | *for example*] your State 2000 program. Without these funds we would [*be unable* | *notbe able*] to continue this program, which is so [*important* | *essential*] to both of us. I strongly urge you to grant him tenure.

متأسفانه (البته برای تام) اِلن به محض آن‌که نگارش و تایپ نامه را تمام می‌کند، یک نامه‌ی دوم نیز می‌نویسد:

---

۱. Tom، Dick، Marilyn، Dean ، Ellen (نام‌های به کار رفته در این پاراگراف)

۲. برای آن‌که مثال به طور دقیق ذکر شود، از برگرداندنِ این نامه و نامه‌ی بعدی به فارسی اجتناب گردیده (مترجم).

Dear Dean Smith,

This [*letter* | *message*] is to give my [*honest* | *frank*] opinion of Prof. Tom Wilson, who is [*a candidate* | *up*] for tenure [*now* | *this year*]. I have [*known* |*worked with*] Tom for [*about* | *almost*] six years. He is a [*poor* | *weak*] researcher not well known in his [*field* | *area*]. His research [*hardly ever* | *rarely*] shows [*insight in* | *understanding of*] the [*key* | *major*] problems of [*the* | *our*] day.

Furthermore, he is not a [*respected* | *admired*] [*teacher* | *educator*]. His students give his [*classes* | *courses*] [*poor* | *bad*] reviews. He is [*our* | *the Department's*] least popular [*teacher* | *instructor*], known [*mostly* | *primarily*] within [*the* | *our*] Department for his [*tendency* | *propensity*] to [*ridicule* | *embarrass*] students [*foolish* | *imprudent*] enough to ask questions in his classes.

[*In addition* | *Additionally*] Tom is a [*poor* | *marginal*] fund raiser. His [*grants* | *contracts*] have brought only a [*meager* | *insignificant*] amount of money into [*the* | *our*] Department. Unless new [*money is* | *funds are*] quickly located, we may have to cancel some essential programs, such as your State 2000 program. Unfortunately, under these [*conditions* | *circumstances*] I cannot in good [*conscience* | *faith*] recommend him to you for [*tenure* | *a permanent position*].

اکنون اِلن کامپیوترش را برنامه‌ریزی می‌کند تا 2^{32} چکیده‌ی پیغامِ مربوط به هر یک از دو نامه را شبانه محاسبه کند. شانس آورد که یک چکیده از نامه‌ی اول با یک چکیده از نامه‌ی دوم، با هم تطبیق دارند. اگر چنین نبود، اِلن می‌توانست چند موردِ بیشتر اضافه کند و مجدداً همین امشب شانسش را آزمایش کند. فرض کنید اِلن یک تطبیق پیدا کند. اسم نامه‌ی "خوب" را A و اسم نامه‌ی "بد" را B می‌گذاریم.

اکنون اِلن نامه‌ی A را جهت تأیید به مریلین ایمیل می‌کند. نامه‌ی B را سرّی نگه می‌دارد و آن را به هیچ کس نشان نمی‌دهد. البته مریلین هم نامه را تأیید می‌کند، چکیده‌ی پیغامِ ۶۴ـ بیتیِ خودش را محاسبه کرده، آن را امضا نموده، و چکیده‌ی امضا شده را به دین اسمیت ایمیل می‌کند. اِلن نیز جداگانه نامه‌ی B (و نه نامه‌ی A) را به دین ایمیل می‌کند.

دین پس از دریافت نامه و چکیده‌ی پیغامی که امضا شده است، الگوریتم چکیده‌ی پیغام را بر روی نامه‌ی B اجرا می‌کند و مشاهده می‌کند که با آنچه مریلین به او ارسال کرده همخوانی دارد و تام را اخراج می‌نماید. دین متوجه این واقعیت نیست که اِلن برنامه‌ریزی کرده تا دو نامه برای یک چکیده‌ی پیغامِ واحد، تولید شود و نامه‌ای که به او فرستاده با نامه‌ای که مریلین دیده و تأیید کرده، فرق دارد. (پایان اختیاریِ داستان: اِلن کاری که انجام داده را به دیک بازگو می‌کند. دیک برآشفته شده و موضوع را برملا می‌سازد. اِلن سراسیمه موضوع را نزد مریلین اعتراف می‌کند. مریلین با دین تماس می‌گیرد. بالاخره تام برای تصدی آن سمت انتخاب می‌شود.) حمله‌ی روز تولد با SHA-1 دشوار است زیرا حتی با سرعتِ گیج کننده‌ی ۱ تریلیون چکیده در ثانیه، ۳۲۰۰۰ سال طول خواهد کشید تا 2^{80}

چکیده‌ی دو نامه همراه با ۸۰ شکلِ متفاوتی که هر کدامشان دارند، محاسبه شوند، و حتی در صورتی که کار به پایان برسد، هیچ تضمینی برای پیدا شدنِ تطبیق وجود ندارد. اگر ابری از یک میلیون تراشه داشته باشیم که به صورت موازی کار کنند، ۳۲۰۰۰ سال به ۲ هفته تقلیل می‌یابند.

۸–۵ مدیریت کلیدهای عمومی

رمزنگاری کلید ـ عمومی این امکان را برای کسانی که از قبل، اشتراکی در یک کلیدِ مشترک ندارند فراهم می‌کند تا ارتباط امنی با هم برقرار کنند. همچنین امکان امضا کردنِ پیغام‌ها بدون حضور یک طرف سومِ قابل اعتماد، فراهم می‌گردد. و سرانجام آن‌که، چکیده‌های پیغامِ امضا شده این امکان را به دریافت کننده می‌دهند تا جامعیت پیغام‌های دریافتی را به آسانی و به صورتی امن، راستی‌آزمایی کند.

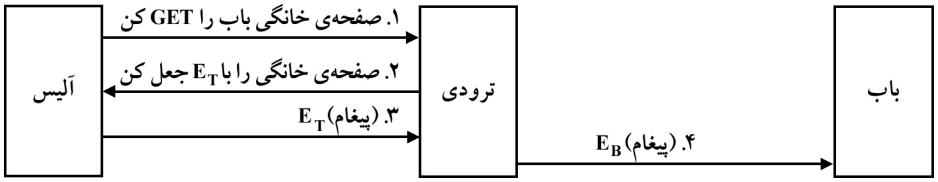
اما مسئله‌ای وجود دارد که آن را نادیده گرفته‌ایم: اگر آلیس و باب همدیگر را نمی‌شناسند، چگونه برای آغاز کردنِ پروسه‌ی برقراریِ ارتباط، کلیدهای عمومی‌شان را به هم می‌دهند؟ به دلیلی که خواهیم گفت، راه‌حلِ بدیهی، یعنی گذاشتنِ کلید عمومی بر روی سایت وبِ شخصی، عمل نمی‌کند. فرض کنید آلیس می‌خواهد کلید عمومی بابَ را در سایت وبِ باب جستجو کند. آلیس چگونه باید این کار را انجام دهد؟ آلیس با تایپ کردنِ آدرس URL باب، کارش را شروع می‌کند. سپس مرورگرِ آلیس آدرس DNS مربوط به صفحه‌ی خانگی باب را جستجو کرده و یک درخواست *GET* به آن ارسال می‌کند (شکلِ ۸–۲۳). متأسفانه ترودی درخواست را رهگیری کرده و با یک صفحه‌ی خانگیِ جعلی پاسخ می‌دهد، که احتمالاً کپی‌ای از صفحه‌ی خانگیِ باب است با این تفاوت که در این صفحه کلید عمومی باب با کلید عمومی ترودی جایگزین شده است. حالا هنگامی‌که آلیس اولین پیغامش را با استفاده از E_T رمزگذاری می‌کند، ترودی این پیغام را رمزبَرداری کرده، آن را خوانده، پیغام را با کلید عمومی باب مجدداً رمزگذاری کرده، و آن را به باب ارسال می‌نماید. باب به هیچ وجه تمایل ندارد ترودی پیغام‌هایی که به او می‌رسند را بخواند. بدتر آنکه، ترودی می‌تواند پیغام‌ها را قبل از آن‌که آن‌ها را مجدداً برای باب رمزگذاری کند، تغییر هم بدهد. روشن است که به مکانیزمی احتیاج داریم تا از امکان تبادل امنِ کلیدهای عمومی اطمینان حاصل کنیم.

۸–۵–۱ گواهینامه‌ها[1]

به عنوان اولین تلاش در توزیع امنِ کلیدهای عمومی، می‌توانیم یک **مرکز توزیع کلید**[2] به نام **KDC** را در نظر آوریم. این مرکز ۲۴ ساعته در دسترس است تا برحسب نیاز، کلید عمومی تأمین کند. یکی از اِشکالات متعددی که به این راه‌حل وارد می‌باشد این است که این طرح مقیاس‌پذیر نیست و مرکز توزیع کلید خیلی زود تبدیل به یک گلوگاه خواهد شد. همچنین اگر این مرکز از کار بیفتد، امنیت اینترنت به یک باره متوقف خواهد شد.

---

1. Certificate (اعتبارنامه) 2. Key Distribution Center



شکل ۸-۲۳ روشی برای تخریب رمزگذاری کلید ـ عمومی توسط ترودی.

به این دلایل راه‌حل دیگری توسعه داده شد، راه‌حلی که در آن نیازی به برخط بودنِ یک مرکز توزیع کلید به صورت همیشگی وجود نداشته باشد. بلکه متعلق بودنِ کلیدهای عمومیِ به افراد، شرکت‌ها، و سایر سازمان‌ها را گواهی نماید. سازمانی که کلیدهای عمومیِ را گواهی می‌کند، اکنون با نام **CA** (**مرجع صدور گواهی**[1]) شناخته می‌شود.

برای مثال فرض کنید باب مایل است به آلیس و افراد دیگری که آن‌ها را نمی‌شناسد اجازه دهد تا به صورتی امن با او ارتباط برقرار کنند. او می‌تواند با کلید عمومی‌اش (با همراه داشتنِ پاسپورت یا گواهینامه‌ی رانندگی) به CA رفته و درخواست گواهی نماید. سازمان CA یک گواهینامه شبیه آنچه در شکل ۸-۲۴ نشان داده شده، صادر می‌کند و پیغام درهمِ SHA-1 (SHA-1 hash) در آن را با کلید خصوصیِ CA امضا می‌کند. سپس باب هزینه‌ی CA را پرداخت کرده و یک CD-ROM می‌گیرد که حاوی گواهی و پیغام درهمِ امضا شده است.

وظیفه‌ی اصلی یک گواهی عبارت‌است‌از همبندسازی[2] یک کلید عمومی به یک موجودیت اصلی[3] (مثلاً یک شخص، یک شرکت و مانند آن). گواهینامه‌ها خودشان سرّی یا محافظت شده نیستند. مثلاً باب ممکن است تصمیم بگیرد گواهینامه‌ی جدیدش را در سایت وبش قرار دهد، همراه با یک پیوند در صفحه‌ی اصلی که می‌گوید: برای مشاهده‌ی گواهی کلید ـ عمومی من این‌جا را کلیک کنید. وقتی کلیک شود، هم گواهی و هم بلوک امضا (یعنی پیغام درهم شده (با SHA-1) و امضا شده‌ی گواهینامه) برگردانده می‌شود.

I hereby certify that the public key
19836A8B03030CF83737E3837837FC3s87092827262643FFA82710382828282A
belongs to
Robert John Smith
12345 University Avenue
Berkeley, CA 94702
Birthday: July 4, 1958
Email: bob@superdupernet.com

SHA-1 hash of the above certificate signed with the CA's private key

شکل ۸-۲۴ یک گواهی نمونه و پیغامِ درهمِ امضا شده‌ی آن.

1. Certification Authority 2. Bind 3. Principal

اکنون بیایید دوباره به سناریوی شکل ۸-۲۳ برگردیم. هنگامی‌که ترودی درخواست آلیس برای صفحه‌ی خانگی باب را رهگیری می‌کند، ترودی چه کاری می‌تواند انجام دهد؟ او می‌تواند گواهی و بلوک امضای خودش را در صفحه‌ی جعلی قرار دهد، اما وقتی آلیس محتوای گواهینامه را می‌خوانَد بلافاصله متوجه می‌شود که در حال صحبت با باب نیست چون اسم باب در آن قرار ندارد. ترودی می‌تواند فوراً و سر بزنگاه صفحه‌ی خانگی باب را ویرایش کرده و کلید عمومی باب را با مال خودش جایگزین کند. با این وجود هنگامی‌که آلیس الگوریتم‌های SHA-1 را بر روی گواهینامه اجرا کند، یک پیغام درهم‌سازی شده می‌گیرد که با آنچه از اِعمال کلید عمومیِ شناخته شده‌ی CA بر روی بلوکِ امضا به دست آورده تطبیق نمی‌کند. چون ترودی کلید خصوصیِ CA را ندارد لذا هیچ راهی برای تولید یک بلوک امضا که حاوی پیغام درهم‌سازی شده‌ی ناشی از صفحه‌ی وبِ ویرایش شده با کلید عمومی خودش است، ندارد. به این ترتیب آلیس می‌تواند اطمینان بیابد که کلید عمومی باب را در اختیار دارد، نه ترودی یا هر کس دیگری. و همان‌طور که وعده دادیم، این نظام نیازی ندارد که CA برای راستی‌آزمایی، برخط باشد و بنابراین یک گلوگاهِ بالقوه برطرف می‌گردد.

هرچند کارکرد استانداردِ یک گواهی عبارت‌است‌از همبندسازیِ یک کلید عمومی به یک طرف اصلی، ولی از گواهی می‌توان برای همبندسازیِ یک کلید عمومی به یک **خصوصیت**[1] نیز استفاده کرد. برای مثال، یک گواهی می‌تواند بگوید: "این کلید عمومی متعلق به یک فرد بالای ۱۸ سال است. " از این موارد می‌توان برای کنترل بعضی دسترسی‌ها استفاده نمود بدون آن‌که شناسه‌ی صاحب کلید نشان داده شود. معمولاً در این‌گونه موارد شخصی که گواهی را در اختیار دارد، آن را به سایت وب، طرفِ اصلی، یا پردازه‌ای خواهد فرستاد که مسئول کنترل اطلاعات است. در آن‌جا یک عدد تصادفی تولید شده و این عدد با استفاده از کلید عمومی‌ای که داخل گواهینامه قرار دارد، رمزگذاری می‌شود. اگر صاحب گواهینامه بتواند این عددِ رمزگذاری شده را رمزبَرداری کند و پس بفرستد، ثابت می‌شود که حقیقتاً خصوصیتِ تأکید شده در گواهینامه را دارد. گزینه‌ی دیگر آن است که می‌تواند از یک عدد تصادفی برای تولید یک کلید نشست برای تماس بعدی استفاده شود.

مثالی دیگر از موردی که گواهینامه ممکن است دربردارنده‌ی یک خصوصیت باشد عبارت است از سیستم توزیع شده‌ی شیءگرا[2]. هر شیء به طور معمول چندین متد دارد. مالک شیء می‌تواند به هر مشتری یک گواهینامه بدهد که نقشه‌ی بیتی (bit map) از این‌که آن مشتری اجازه‌ی احضار (invoke) کدام متدها را دارد، در آن باشد و آن نقشه‌ی بیتی را با استفاده از یک گواهینامه‌ی امضا شده، به یک کلید عمومی همبند نماید. در این‌جا هم اگر شخصی که گواهی را در اختیار دارد بتواند مالکیتِ کلید خصوصیِ متناظر را ثابت کند، مُجاز به اجرای متدهای موجود در نقشه‌ی بیتی خواهد بود. این رویکرد دارای این ویژگی است که نیازی به دانستنِ شناسه‌ی مالک وجود ندارد. این ویژگی در جاهایی که محرمانگی اهمیت دارد، سودمند می‌باشد.

1. Attribute 2. Object-oriented distributed system

X.509 ۲-۵-۸

اگر همه‌ی متقاضیان امضا، با داشتنِ انواع متفاوتی از گواهینامه، به سراغ CA می‌رفتند در این صورت مدیریت تمام این فرمت‌های گوناگون به زودی تبدیل به یک مسئله می‌شد. برای حل این مسئله یک استاندارد از طرف ITU برای گواهینامه‌ها ایجاد و تأیید شده است. نام این استاندارد X.509 است و به صورت گسترده در اینترنت مورد استفاده قرار دارد. از استاندارد ابتدایی که ایجاد شد تا به حال سه نگارش از آن ارائه شده و لذا ما در این‌جا V3 را بررسی خواهیم کرد.

استاندارد X.509 خیلی تحت تأثیر OSI بوده و بعضی از بدترین ویژگی‌های آن را به امانت گرفته است (از قبیل نام‌گذاری (naming) و کدگذاری). جالب این‌جاست که IETF با X.509 موافقت کرد، درحالی‌که در زمینه‌های دیگر، از آدرس‌های ماشین گرفته تا پروتکل‌های حمل برای فرمت‌های ایمیل، IETF عموماً OSI را نادیده گرفته و سعی می‌کرد درست عمل کند. نگارش IETF از X.509 در RFC 5280 شرح داده شده است.

استاندارد X.509 در اصل روشی برای توصیف گواهینامه‌هاست. فیلدهای اصلی در یک گواهینامه در شکل ۸-۲۵ فهرست شده‌اند. توضیحاتی که در شکل داده شده، بایستی ایده‌ی کلی از عملکرد فیلدها را نشان دهد. برای اطلاعات بیشتر لطفاً یا به خودِ استاندارد و یا به RFC 2459 مراجعه نمایید.

به طور مثال اگر باب در دپارتمان وام در بانکی به نام "پول" (Money Bank) مشغول به کار باشد، آدرس X.509 او احتمالاً به صورت زیر است:

/C=US/O=MoneyBank/OU=Loan/CN=Bob/

در این آدرس *C* نشان دهنده‌ی کشور (country)، *O* نشان دهنده‌ی سازمان (organization)، *OU* نشان دهنده‌ی واحد سازمانی (organizational unit)، و *CN* نشان دهنده‌ی نام متداول (common name) می‌باشد. سازمان‌های CA و سایر موجودیت‌ها به همین شیوه نام‌گذاری می‌شوند. یک مسئله‌ی مهم در ارتباط با نام‌های X.500 آن است که اگر آلیس سعی کند با *bob@moneybank.com* تماس بگیرد و گواهینامه‌ای با یک نام X.500 ارائه داده باشد، مطمئن نیست که آن گواهینامه او را به همان "باب" مورد نظر او ارجاع بدهد. خوشبختانه با شروع نگارش ۳، نام‌های DNS به جای نام‌های X.500 آمده‌اند و لذا این مسئله احتمالاً از بین رفته است.

گواهینامه‌ها با استفاده از **ASN.1** OSI[1] کد می‌شوند که شبیه ساختاری در زبان C است، ولی علامت نوشتاریِ آن‌ها بسیار خاص و طول و تفصیل‌دار است. اطلاعات بیشتر درباره‌ی X.509 توسط Ford و Baum (۲۰۰۰) ارائه شده است.

---

۱. Abstract Syntax Notation 1 : نشان‌گذاری گرامر به صورت انتزاعی-۱

نام فیلد	معنی
نگارش (Version)	شماره‌ی نگارش X.509
شماره‌ی سریال (Serial number)	این عدد به علاوه‌ی نام CA گواهینامه را به صورتی یکتا مشخص می‌کند.
الگوریتم امضا (Signature algorithm)	الگوریتم مورد استفاده برای امضای گواهینامه
صادرکننده (Issuer)	نام X.509 مربوط به CA
دوره‌ی اختیار (Validity period)	زمان شروع و پایان مربوط به دوره‌ی زمانیِ اعتبار
نام موضوع (Subject name)	موجودیتی که کلیدش دارد گواهی می‌شود
کلید عمومی (Public key)	کلید عمومیِ موضوع و ID الگوریتمی که از آن استفاده می‌کند
شناسه‌ی صادرکننده (Issuer ID)	یک ID اختیاری که به صورتی یکتا صادرکننده‌ی گواهینامه را شناسایی می‌کند
شناسه‌ی موضوع (Subject ID)	یک ID اختیاری که به صورتی یکتا موضوع گواهینامه را شناسایی می‌کند
گسترش‌ها (Extensions)	گسترش‌های متعددی تعریف شده‌اند
امضا (Signature)	امضای گواهینامه (امضا شده توسط کلید خصوصیِ CA)

شکل ۸-۲۵ فیلدهای اصلی در یک گواهینامه‌ی X.509 .

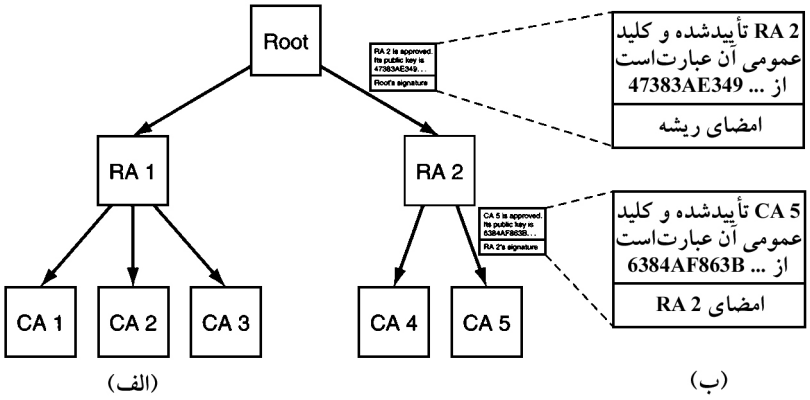
۸-۵-۳ زیرساخت‌های کلید عمومی

واضح است که داشتنِ یک CA منفرد برای صادر کردنِ تمام گواهینامه‌های دنیا، عملی نخواهد بود. این روش زیر بار متلاشی خواهد شد، ضمنِ آن‌که در این صورت CA یک نقطه‌ی خرابیِ مرکزی[1] خواهد بود. یک راه‌حلِ محتمل، می‌تواند عبارت باشد از داشتنِ چندین CA که همگی توسط یک سازمان اجرا شوند و برای امضای گواهینامه‌ها از یک کلید خصوصیِ یکسان استفاده کنند. هر چند این کار مشکل بار و خرابی را حل خواهد کرد ولی یک مسئله‌ی تازه ایجاد می‌کند: نشت کلید (key leakage). اگر چندین دوجین خدمتگزار وجود داشته باشند که در سرتاسر دنیا پخش شده‌اند، و همگی کلید خصوصیِ CA را داشته باشند، شانس به سرقت رفتنِ کلید خصوصی یا افشا شدنِ آن بسیار افزایش خواهد یافت. از آن‌جا که به خطر افتادنِ این کلید، زیرساخت امنیت الکترونیکیِ جهان را به نابودی خواهد کشاند لذا داشتنِ یک CA مرکزیِ منفرد از ریسک بالایی برخوردار است.

به‌علاوه، کدام سازمان CA را اداره خواهد کرد؟ تصوّرِ این‌که مرجعی وجود داشته باشد که به لحاظ مشروعیت و معتمد بودن در سراسر دنیا قابل پذیرش باشد، دشوار است. در بعضی کشورها مردم اصرار دارند که چنین مرجعی بایستی دولتی باشد درحالی‌که مردم در بقیه‌ی کشورها اصرار بر غیردولتی بودنِ آن دارند.

به این دلایل، روش متفاوتی برای گواهی کردنِ کلیدهای عمومی ایجاد شد. این روش تحت نام کلی **PKI** (**زیرساخت کلید عمومی**[2]) شناخته می‌شود. در این بخش نحوه‌ی کار آن را به طور خلاصه بررسی خواهیم کرد، ضمناً طرح‌های متعددی در این مورد ارائه شده‌اند بنابراین احتمالاً جزئیاتِ آن در طول زمان تکمیل خواهد شد.

1. Central point of failure 2. Public Key Infrastructure



شکل ۸-۲۶ (الف) یک PKI سلسله مراتبی. **(ب)** یک زنجیره از گواهینامه‌ها.

یک PKI چندین مؤلفه[1] دارد: کاربران، CAها، گواهینامه‌ها، و راهنماها[2]. آنچه PKI انجام می‌دهد عبارت‌است‌از ایجاد روشی جهت ساختاربندی این مؤلفه‌ها و تعریف استانداردهایی برای انواع اسناد و پروتکل‌ها. یک شکل کاملاً ساده از PKI عبارت‌است‌از یک سلسله مراتب از CA ها، یعنی آن طور که در شکل ۸-۲۶ نشان داده شده. در این مثال سه سطح را نشان داده‌ایم اما در عمل تعداد سطوح ممکن است کمتر یا بیشتر باشند. ریشه، یعنی CA سطح بالا، CA های سطح دوم را گواهی می‌کند که آن‌ها را در این‌جا **RA** ها (**مراجع ناحیه‌ای**[3]) می‌نامیم، زیرا می‌توانند یک ناحیه‌ی جغرافیایی (از قبیل یک کشور یا اقلیم) را پوشش دهند. این واژه استاندارد نیست، و در حقیقت هیچ واژه‌ای برای سطوح مختلف درخت، واقعاً استاندارد نیست. این RA ها به نوبه‌ی خود CAهای واقعی را گواهی می‌نمایند، یعنی CAهایی که گواهینامه‌های X.509 را برای سازمان‌ها و اشخاص صادر می‌کنند. هنگامی که ریشه یک RA جدید را تأیید می‌کند، یک گواهی X.509 تولید می‌کند که حاکی از آن است که RA را تأیید کرده است. کلید عمومی RA در این گواهی است. ریشه این گواهی را امضا می‌کند و آن را به RA می‌دهد. به همین ترتیب هنگامی‌که یک RA یک CA جدید را تأیید می‌کند، یک گواهینامه ایجاد و آن را امضا می‌کند که این کار حکایت از آن دارد که آن CA را تأیید کرده و گواهینامه دربردارنده‌ی کلید عمومی CA می‌باشد.

طرز کار PKI ما نیز به همین طریق است. فرض کنید آلیس برای برقراری ارتباط با باب به کلید عمومی او نیاز داشته باشد، بنابراین کلید عمومی باب را جستجو کرده و گواهینامه‌ای می‌یابد که حاوی این کلید بوده و توسط CA 5 امضا شده است. اما آلیس هرگز چیزی درباره‌ی CA 5 نشنیده است. همه‌ی آنچه که آلیس می‌داند آن است که شاید CA 5 دختر ۱۰ ساله‌ی باب باشد. آلیس بایستی به CA 5 برود و بگوید: "قانونی بودن خودت را اثبات کن." در پاسخ، CA 5 گواهینامه‌ای را ارائه

1. Component 2. Directory 3. Regional Authority

می‌دهد که از RA 2 دریافت کرده و حاوی کلید عمومی CA 5 است. حالا آلیس که مجهز به کلید عمومی CA 5 است می‌تواند راستی‌آزمایی کند که آیا حقیقتاً گواهینامه‌ی باب به وسیله‌ی CA 5 امضا شده یا خیر (یعنی این گواهینامه قانونی هست یا خیر).

مگر آن‌که RA 2 هم پسر ۱۲ ساله‌ی باب باشد. بنابراین مرحله‌ی بعدی برای آلیس عبارت است از درخواست از RA 2 برای اثبات قانونی بودنِ خویش. پاسخ به درخواست آلیس عبارت است از یک گواهینامه که به وسیله‌ی ریشه امضا شده است و دربردارنده‌ی کلید عمومی RA 2 می‌باشد. حالا آلیس مطمئن است که کلید عمومی باب را در اختیار دارد.

اما آلیس چگونه کلید عمومی ریشه را پیدا می‌کند؟ عجب. فرض بر این است که همه کلید عمومی ریشه را می‌دانند. برای مثال، مرورگرِ آلیس ممکن است همراه با کلید عمومی ریشه که به صورت توکار در آن قرار دارد، عرضه شده باشد.

باب انسان منصفی است و نمی‌خواهد باعث زحمت آلیس شود. باب می‌داند که آلیس ناچار است CA 5 و RA 2 را کنترل کند، بنابراین برای آن‌که به او کمک کند دو گواهینامه‌ی مورد نیاز را گردآوری کرده و به آلیس می‌دهد. اکنون آلیس می‌تواند از اطلاعاتِ خودش درباره‌ی کلید عمومی ریشه استفاده کرده و گواهینامه‌ی سطح بالا را راستی‌آزمایی کند و سپس از کلید عمومیِ داخل آن برای راستی‌آزماییِ دومین کلید استفاده نماید. آلیس برای انجام راستی‌آزمایی نیازی ندارد با کس دیگری تماس بگیرد. چون گواهینامه‌ها همگی امضا شده هستند، آلیس می‌تواند به سادگی هرگونه تلاش برای نفوذ و جعل را کشف نماید. زنجیره‌ای از گواهینامه‌ها که همانند این مثال تا ریشه به عقب برگردند، بعضی اوقات یک **زنجیره‌ی اعتماد**[1] یا یک **مسیر صدور گواهینامه**[2] نامیده می‌شود. از این تکنیک در عمل استفاده‌ی گسترده‌ای می‌شود.

البته اکنون با این پرسش مواجهیم که چه کسی ریشه را اجرا می‌کند؟ راه‌حل این است که یک ریشه‌ی منفرد نداشته باشیم، بلکه تعداد زیادی ریشه وجود داشته باشند که هر کدام RAها و CAهای خودشان را دارند. در حقیقت مرورگرهای امروزی زمانی که به بازار می‌آیند، کلید عمومیِ بیشتر از ۱۰۰ ریشه از قبل در آن‌ها بارگذاری شده است و لذا بعضی اوقات با عنوان **لنگرهای اعتماد**[3] به آن‌ها اشاره می‌شود. به این ترتیب می‌توان از داشتنِ یک **مرجعِ قابل اعتمادِ**[4] سراسری اجتناب نمود.

اما حالا با این موضوع مواجه هستیم که فروشنده‌ی مرورگر چگونه تصمیم بگیرد که کدام یک از لنگرهای اعتمادِ ادعایی، قابل اتکا بوده و کدام‌ها کاذب (sleazy) هستند. این کاملاً به کاربر بستگی دارد که به کدام یک از انتخاب‌های فروشنده‌ی مرورگر اعتماد کند که انتخاب‌های مدبرانه‌ای انجام داده و به آسانی با همه‌ی لنگرهای اعتماد و پرداخت هزینه‌ی آن‌ها موافقت نکند. اغلب مرورگرها به کاربران اجازه می‌دهند تا کلیدهای ریشه را بررسی کنند (که معمولاً به صورت گواهینامه‌های امضا شده به وسیله‌ی ریشه هستند) و هر موردی که به نظر مشکوک می‌رسند را حذف نمایند.

---

1. Chain of trust 2. Certification path 3. Trust anchor 4. Trusted authority

راهنماها

مبحث بعدی در ارتباط با PKI عبارت‌است‌از محل ذخیره‌ی گواهینامه‌ها (و زنجیره‌های آن‌ها در جهت عقب، تا جایی که به یک لنگر اعتمادِ شناخته شده برسند).یک احتمال عبارت‌است‌از این‌که هر کاربر را وادار کنیم تا گواهینامه‌های مربوط به خودش را ذخیره کند. هرچند این کار امن است (چون هیچ راهی برای کاربران وجود ندارد تا بدون آن‌که شناخته شوند، با گواهینامه‌های امضا شده نفوذ کنند)، ولی با دردسر نیز همراه می‌باشد. گزینه‌ی دیگری که ابداع شده، استفاده از DNS به عنوان یک راهنمای گواهینامه[1] است. آلیس قبل از تماس با باب احتمالاً باید با استفاده از DNS آدرس IP باب را پیدا کند. پس چرا DNS را وادار نکنیم تا کل زنجیره‌ی گواهینامه‌ی باب را نیز همراه با آدرس IP اش برگردانَد؟

بعضی‌ها فکر می‌کنند این روش قابل اجراست، اما سایرین خدمتگزارهای اختصاصیِ راهنما را ترجیح می‌دهند، خدمتگزارهایی که تنها وظیفه‌شان مدیریت گواهینامه‌های X.509 است. چنین راهنماهایی می‌توانند سرویس‌های جستجو را با استفاده از ویژگی‌های نام‌های X.509 فراهم کنند. برای مثال، به لحاظ نظری، چنین راهنمایی می‌تواند یک پرس‌وجو مانند این را پاسخ دهد: "فهرستی از تمام افرادی که نامشان آلیس است و در دپارتمان فروش در ایالات متحده یا کانادا هستند را به من بده."

ابطال

جهان واقعی مملو از گواهینامه‌هاست، از قبیل پاسپورت‌ها و گواهینامه‌های رانندگی. بعضی اوقات این گواهینامه‌ها قابل ابطال[2] هستند، به عنوان مثال گواهینامه‌های رانندگی به ازای بعضی تخلفات رانندگی می‌توانند باطل شوند. همین مسئله در دنیای دیجیتال نیز اتفاق می‌افتد: اعطا کننده‌ی یک گواهینامه ممکن است تصمیم به ابطالِ آن بگیرد زیرا شخص یا سازمانی که این گواهینامه را در اختیار دارد، به طریقی از آن سوء استفاده کرده است. همچنین اگر کلید خصوصیِ موضوعِ مورد بحثمان در معرض افشا قرار داشته باشد، یا بدتر از آن، کلید خصوصیِ CA مورد مصالحه قرار گرفته باشد، امکان ابطال گواهینامه وجود دارد. بنابراین ضرورت دارد که یک PKI با بحث ابطال نیز سروکار داشته باشد. احتمالِ ابطال سبب پیچیده شدنِ قضیه می‌گردد.

اولین گام در این راستا آن است که هر یک از CAها را وادار کنیم تا مرتباً یک **CRL** منتشر کنند (**فهرست ابطال گواهینامه**[3]). در این فهرست شماره‌ی سریال تمام گواهینامه‌هایی که باطل شده‌اند ارائه می‌شود. از آن‌جا که گواهینامه‌ها دربردارنده‌ی تاریخ انقضا هستند، لذا CRL تنها لازم است شامل شماره سریال گواهینامه‌هایی باشد که هنوز منقضی نشده‌اند. هنگامی‌که تاریخ انقضای یک گواهینامه سپری شود، به صورت خودکار نامعتبر می‌شود. بنابراین احتیاجی نیست که میان گواهینامه‌هایی که فقط مهلتشان تمام شده، با گواهینامه‌هایی که عملاً باطل شده‌اند، تمایزی قائل شویم. در هر دو مورد، این گواهینامه‌ها دیگر نمی‌توانند مورد استفاده قرار گیرند.

---

1. Certificate directory 2. Revocation 3. Certificate Revocation List

متاسفانه ارائه‌ی CRLها به این معناست که کاربری که در حال استفاده از یک گواهینامه می‌باشد، اکنون باید CRL را بررسی کند که آیا باطل شده یا خیر. اگر باطل شده باشد، دیگر نباید از آن استفاده کند. اما حتی اگر گواهینامه‌ای در فهرست نباشد، این امکان وجود دارد که درست بعد از انتشار فهرست باطل شده باشد. پس تنها راهی که واقعاً مطمئن شویم، پرسیدن از CA است. و در استفاده‌ی بعدی از همان گواهینامه، بایستی مجدداً از CA سوال شود چون ممکن است چند ثانیه قبلش گواهینامه باطل شده باشد.

پیچیدگی بعدی آن است که یک گواهینامه‌ی ابطال شده می‌تواند امکانِ بازگشتِ دوباره[1] داشته باشد. مثل وقتی که به دلیل عدم پرداخت هزینه‌ی یک سری خدمات گواهینامه باطل شده باشد ولی بعداً هزینه‌ها پرداخت شوند. این‌که مجبور باشیم با ابطال (و احتمالاً بازگشت از ابطال) درگیر باشیم باعث می‌شود که در یکی از بهترین خصوصیات گواهینامه‌ها (یعنی استفاده از گواهینامه‌ها بدون اجبار در تماس با یک CA) دچار محدودیت باشیم.

کجا بایستی CRLها ذخیره شوند؟یک محل خوب عبارت‌است‌از همان محلی که خود گواهینامه‌ها ذخیره می‌شوند. یک راهبُرد آن استکه CAها به صورت دوره‌ای CRLها را بازنگری کرده و راهنماها را وادار کنند تا گواهینامه‌های باطل شده را حذف کنند. اگر از راهنماها برای ذخیره‌ی گواهینامه‌ها استفاده نشود، CRLها می‌توانند در محل‌های مختلفی در سرتاسر شبکه درون حافظه‌های پنهان ذخیره شوند. از آنجا که CRL خودش یک سند امضا شده است، درصورتی‌که به آن نفوذ شود، این موضوع به سادگی قابل کشف است.

در صورتی‌که گواهینامه‌ها دوره‌ی عمر طولانی داشته باشند، CRLها نیز طول و دراز خواهند شد. برای مثال اگر کارت‌های اعتباری تا ۵ سال معتبر باشند، صف ابطالی‌هایی که تکلیفشان معین نشده بسیار طولانی‌تر از حالتی خواهد شد که کارت‌های جدید هر سه ماه یک بار صادر شوند. یک روش استاندارد برای برخورد با CRL های طولانی آن است که فهرست اصلی به دفعاتِ کمتر منتشر شود ولی به‌هنگام‌سازی‌ها به دفعات بیشتری منتشر گردند. با این کار پهنای باندِ مورد نیاز برای توزیع CRLها کاهش می‌یابد.

۸-۶ امنیت ارتباطات

ما اکنون مطالعه‌ی ابزار تجارت را به پایان رساندیم. اغلب روش‌ها و پروتکل‌های مهم پوشش داده شده‌اند. مابقی این فصل مربوط به این است که چگونه این روش‌ها در عمل اِعمال می‌گردند تا امنیت شبکه تأمین شود. ضمناً بعضی آرا در ارتباط با جنبه‌های اجتماعیِ امنیت در پایان فصل بررسی خواهند شد.

1. Reinstatement

در چهار بخشی که در پیش رو داریم، به امنیت ارتباطات توجه خواهیم کرد، یعنی چگونه بیت‌ها را به صورتی امن بگیریم بدون آن‌که از مبدأ تا مقصد ویرایشی بر روی آن‌ها شده باشد، و نیز چگونه بیت‌هایی را که نمی‌خواهیم، راه ندهیم. البته این‌ها تنها موارد امنیتیِ مطرح در شبکه‌بندی نیستند، اما مطمئناً از جمله مهم‌ترین موارد می‌باشند و لذا شروع خوبی برای مطالعه‌ی ما به حساب می‌آیند.

۸-۶-۱ IPsec

گروه IETF سال‌ها می‌دانسته است که امنیت در اینترنت مغفول مانده. اضافه کردنِ آن آسان نبود زیرا درباره‌ی محل استقرار آن، جدالی در جریان بود. اغلب کارشناسان شبکه عقیده داشتند برای یک امنیت واقعی، کنترل‌های رمزگذاری و جامعیت باید انتها ـ به ـ انتها باشند (مثلاً در لایه‌ی کاربرد). به این معنا که پردازه‌ی مبدأ، داده‌ها را رمزگذاری کرده و/یا جامعیتِ آن‌ها را حفظ می‌کند و آن‌ها را به پردازه‌ی مقصد ارسال می‌کند، یعنی جایی که داده‌ها رمزبَرداری شده و/ یا راستی‌آزمایی می‌شوند. بنابراین هرگونه نفوذ میان این دو پردازه، همچنین نفوذ در سیستم عامل آن‌ها، قابل تشخیص خواهد بود. مشکلی که در این رویکرد وجود دارد لزوم ایجاد تغییر در تمام کاربردهاست، به طوری که آگاه ـ به ـ امنیت[1] باشند. در همین دیدگاه، بهترین رویکردِ بعدی عبارت است از قرار دادنِ رمزگذاری در لایه‌ی حمل یا در یک لایه‌ی جدید مابین لایه‌ی کاربرد و لایه‌ی حمل. یعنی باز هم امنیت را انتها ـ به ـ انتها قرار دهیم اما نیازی به تغییر کاربردها نباشد.

دیدگاهِ متضاد به این صورت است که کاربران از امنیت سررشته‌ای ندارند و نخواهند توانست به درستی از آن استفاده کنند، و هیچ کس به هیچ ترتیبی تمایلی به ویرایش برنامه‌های موجود ندارد، بنابراین لایه‌ی شبکه بایستی بدون درگیر کردنِ کاربران، بسته‌ها را تصدیق هویت و/یا رمزگذاری کند. پس از سال‌ها جدال شدید، این دیدگاه حمایت کافی جذب کرد به طوری که یک استاندارد امنیت لایه‌ی شبکه تعریف گردید. استدلال تقریباً به این صورت بود که داشتنِ رمزگذاریِ لایه‌ی شبکه از این‌که کاربرانِ آگاه ـ به ـ امنیت این کار را درست انجام دهند، جلوگیری نمی‌کند و ضمناً تا حدی به کاربران ناآگاه ـ به ـ امنیت[2] نیز کمک می‌کند.

نتیجه‌ی این نبرد طرحی به نام **IPsec** (**امنیت IP**[3]) بود که در RFC های 2401، 2402، و 2406 شرح داده شده است. همه‌ی کاربران خواهان رمزگذاری نیستند (زیرا به لحاظ محاسباتی گران‌قیمت است) بلکه می‌خواهند این کار اختیاری باشد، بنابراین تصمیم گرفته شد که رمزگذاری همیشه ضروری باشد ولی اجازه‌ی استفاده از یک الگوریتم تهی (null algorithm) وجود داشته باشد. الگوریتم تهی در RFC 2410 شرح داده شده و به لحاظ سادگی، راحتی در پیاده‌سازی، و سرعتِ زیاد تحسین شده است.

1. Security-aware 2. Security-unaware 3. IP security

طراحی کامل IPsec چارچوبی است برای سرویس‌ها، الگوریتم‌ها، و دانه‌بندی‌ها. دلیلِ سرویس‌های متعدد آن است که همه مایل نیستند برای داشتنِ تمام سرویس‌ها به صورت دائمی، هزینه پرداخت کنند لذا سرویس‌ها به صورت جدا از هم در دسترس قرار می‌گیرند. سرویس‌های مهم عبارتند از رازپوشی (secrecy)، جامعیت داده (data integrity)، و محافظت در برابر حمله‌های بازنواخت (replay attack) (یعنی هنگامی‌که نفوذگر یک ارتباط را مجدداً راه‌اندازی می‌کند). همه‌ی این موارد مبتنی بر رمزنگاری کلید ـ متقارن هستند زیرا کاراییِ بالا بسیار حیاتی است.

دلیلِ داشتنِ چندین الگوریتم، آن است که الگوریتمی که امروز به نظر امن می‌رسد ممکن است در آینده شکسته شود. اگر الگوریتم IPsec مستقل باشد، حتی اگر یک الگوریتم خاصی شکسته شود، چارچوب کلی می‌تواند همچنان کارش را ادامه دهد.

دلیلِ داشتنِ چندین دانه‌بندی، آن است که علاوه بر امکاناتِ متداول، امکانِ محافظت از یک اتصال TCP منفرد، محافظت از کلِ ترافیک مابین یک جفت میزبان، یا محافظت از کلِ ترافیک مابین یک جفت مسیریابِ امن، نیز فراهم باشد. جنبه‌ای از IPsec که تا حدودی عجیب است آن است که هرچند در لایه‌ی IP قرار دارد ولی اتصال ـ گرا است. البته در واقع این امر چندان هم عجیب نیست زیرا برای داشتنِ امنیت، بایستی دست کم برای مدتی یک کلید ایجاد و استفاده شود ــ در اصل، نوعی اتصال با یک نام متفاوت. همچنین هزینه‌ی برپا شدنِ اتصال‌ها به ازای تعداد زیاد بسته‌ها، مستهلک می‌شود. در واژه‌شناسیِ IPsec کلمه‌ی "اتصال" ("connection") یک **SA** (**ارتباط امنیتی**[1]) نامیده می‌شود. یک SA عبارت‌است‌از یک اتصال یک‌سویه[2] مابین دو نقطه‌ی انتهایی. این اتصال یک شناسه‌ی امنیتی دارد که با آن اتصال مرتبط شده است. درصورتی‌که در هر دو جهت به ترافیک امن نیاز باشد، به دو SA نیز احتیاج است. شناسه‌های امنیتی در بسته‌هایی حمل می‌شوند که بر روی این اتصال‌های امن در حرکت هستند. وقتی یک بسته‌ی امن می‌رسد، از این شناسه‌ها برای جستجوی کلیدها و سایر اطلاعات مرتبط استفاده می‌شود.

به لحاظ فنی IPsec دو بخش اصلی دارد. بخش اول دو سرآیند جدید ارائه می‌دهد که می‌توانند برای حمل شناسه‌ی امنیتی، داده‌ی کنترلِ جامعیت، و اطلاعاتِ دیگر، به بسته‌ها اضافه شوند. بخش دیگر به نام **ISAKMP** (**پروتکل مدیریتِ کلید و ارتباط امنیتیِ اینترنت**[3]) است و مربوط به برقراریِ کلیدها می‌باشد. پروتکل ISAKMP یک چارچوب است. پروتکل اصلی برای انجام کار **IKE** (**تبادل کلید اینترنت**[4]) می‌باشد. همان‌طور که در RFC 4306 توضیح داده شده، و توسط Perlman و Kaufman (۲۰۰۰) مورد اشاره قرار گرفته، باید از نگارش شماره‌ی 2 از IKE استفاده شود زیرا نگارش قبلی دچار ایراد اساسی بوده است.

---

1. Security Association 2. Simplex
3. Internet Security Association and Key Management Protocol 4. Internet Key Exchange

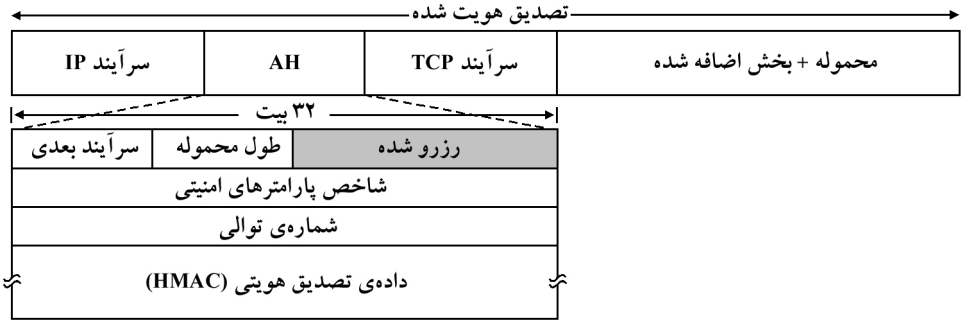
از IPsec می‌توان در یکی از دو مود استفاده کرد. در **مود حمل**[1] سرآیند IPsec درست بعد از سرآیند IP درج می‌شود. فیلد *Protocol* در سرآیند IP تغییر داده می‌شود به نحوی که تشخیص دهد یک سرآیند IPsec به دنبال سرآیند IP عادی (قبل از سرآیند TCP) قرار دارد. سرآیند IPsec شامل اطلاعات امنیتی و در درجه‌ی اول شاملِ شناسه‌ی SA، یک شماره‌ی توالی جدید، و احتمالاً یک کنترل جامعیتِ محموله می‌باشد.

در **مود تونل**[2] کل بسته‌ی IP، یعنی سرآیند و بقیه‌ی اجزای بسته، در بدنه‌ی یک بسته‌ی IP جدید محصورسازی می‌گردد، همراه با یک سرآیند IP کاملاً جدید. مود تونل هنگامی سودمند است که تونل در محلی غیر از مقصد نهایی خاتمه می‌یابد. بعضی مواقع پایان تونل یک ماشین است که نقش دروازه‌ی امنیتی[3] را دارد (به طور مثال، یک دیواره‌ی آتشِ شرکت است). این حالت در مورد VPN (شبکه‌ی خصوصیِ مَجازی[4]) رایج می‌باشد. در این مود، دروازه‌ی امنیتی بسته‌ها را در حین عبور از دروازه، محصورسازی کرده و یا از شکل محصورسازی شده خارج می‌کند. با خاتمه یافتنِ تونل در این ماشین امنیتی، ماشین‌هایی که بر روی LAN شرکت هستند اجباری به اطلاع از IPsec ندارند. فقط دروازه‌ی امنیتی باید از IPsec اطلاع داشته باشد.

مود تونل همچنین هنگامی سودمند است که یک بندیل از اتصالات TCP با یکدیگر جمع شده و به عنوان "یک" جریانِ رمزگذاری شده با آن‌ها رفتار می‌شود، زیرا اجازه نمی‌دهد یک نفوذگر بتواند ببیند چه کسی برای چه کسی، چه تعداد بسته ارسال می‌کند. بعضی اوقات حتی دانستنِ این‌که چه حجم ترافیکی به کجا می‌رود، اطلاع ارزشمندی است. به طور مثال اگر در حین یک بحران نظامی، حجم جریان ترافیک مابین پنتاگون[5] و کاخ سفید[6] به سرعت کاهش یافته ولی حجم ترافیک مابین پنتاگون و بعضی تأسیسات نظامی در اعماق کوهستان راکی در کلورادو[7] به همان میزان افزایش بیابد، یک نفوذگر ممکن است بتواند اطلاعات به درد بخوری را از این داده‌ها استنتاج کند. مطالعه‌ی الگوی جریان بسته‌ها، حتی اگر رمزگذاری شده باشند، **تحلیل ترافیک**[8] نامیده می‌شود. مود تونل تا حدی روشی برای خنثی کردنِ تحلیل ترافیک فراهم می‌سازد. اِشکال مود تونل عبارت‌استاز افزودنِ یک سرآیند IP اضافی، و در نتیجه افزایش قابل ملاحظه در اندازه‌ی بسته. برعکس، مود حمل تأثیر چندانی در اندازه‌یِ بسته ندارد.

اولین سرآیند جدید **AH** است (**سرآیند تصدیق هویت**[9]). این سرآیند، امکانِ کنترل جامعیت و تامین امنیتِ ضد بازنواخت[10] را تأمین می‌کند، ولی رازپوشی را خیر (به عبارت دیگر داده را رمزگذاری نمی‌کند). استفاده از AH در مود حمل در شکل ۸-۲۷ نشان داده شده است. در IPv4 ، این سرآیند میان سرآیند IP (شامل تمام گزینه‌ها) و سرآیند TCP جا داده می‌شود. در IPv6، AH فقط یک سرآیند

1. Transport mode 2. Tunnel mode 3. Security gateway machine 4. Virtual Private Network
5. Pentagon 6. White House 7. Colorado Rocky Mountains 8. Traffic analysis
9. Authentication Header 10. Antireplay security



شکل ۸-۲۷ سرآیند تصدیق هویتِ IPsec در مود حمل برای IPv4.

گسترشیِ دیگر تلقی شده و از این منظر با آن رفتار می‌شود. در حقیقت، این فرمت نزدیک به فرمت مربوط به سرآیند گسترشیِ IPv6 استاندارد است. همان‌طور که نشان داده شده، محموله می‌تواند به اندازه‌ی طول مشخصی که برای الگوریتم تصدیق هویت لازم است، اضافه شود.

اکنون سرآیند AH را بررسی می‌کنیم. فیلد *Next header* جهت ذخیره‌ی مقداری که فیلد IP *Protocol* قبل از جایگزین شدن با 51 داشته است (تا نشان دهد که یک سرآیند AH بعد از این سرآیند قرار دارد) به کار می‌رود. در اغلب موارد، کدِ مربوط به TCP (یعنی 6) در این محل می‌آید. فیلد *Payload length* تعداد کلمه‌های ۳۲ـ بیتی در سرآیند AH منهای عدد ۲ است.

فیلد *Security parameters index* شناسه‌ی اتصال است. این فیلد توسط ارسال‌کننده درج شده تا نشان دهنده‌ی رکورد مشخصی در پایگاه داده‌ی دریافت کننده باشد. این رکورد حاوی کلید اشتراکیِ به کار رفته بر روی این اتصال است و نیز اطلاعات دیگری درباره‌ی اتصال دارد. اگر این فیلد به جای IETF توسط ITU ابداع شده بود، به نام *Virtual circuit number* نامیده می‌شد.

فیلد *Sequence number* برای شماره‌گذاریِ تمام بسته‌های ارسالی بر روی یک SA استفاده می‌شود. هر بسته یک شماره‌ی یکتا می‌گیرد، حتی در ارسال‌های مجدد. به عبارت دیگر، ارسال مجددِ یک بسته در این‌جا یک شماره‌ی متفاوت از شماره‌ی اولیه‌اش می‌گیرد (گرچه شماره‌ی توالیِ TCP تفاوتی نکند). هدف از این فیلد، تشخیصِ حمله‌های بازنواخت می‌باشد. این شماره‌های توالی در صورتی‌که به انتها برسند نمی‌توانند مجدداً از ابتدا آغاز شوند[1]. اگر تمام 2^{32} شماره‌ی توالی مصرف شوند، بایستی برای ادامه‌ی ارتباط، یک SA جدید برقرار شود.

نهایتاً به فیلد *Authentication data* می‌رسیم که فیلدی با طول متغیر است و حاوی امضای دیجیتالی محموله می‌باشد. هنگامی‌که SA برقرار می‌شود، دو طرف در این باره که از چه الگوریتمی برای امضا استفاده کنند، مذاکره می‌کنند. معمولاً در این مواقع از رمزنگاری کلید ـ عمومی استفاده نمی‌شود زیرا بسته‌ها باید با سرعت فوق‌العاده زیادی پردازش شوند و همه‌ی الگوریتم‌های شناخته شده‌ی کلید

1. Wrap around

ـعمومی خیلی کُند هستند. از آن‌جا که IPsec مبتنی بر رمزنگاری کلید ـمتقارن است و ارسال‌کننده و دریافت‌کننده قبل از برپاسازیِ یک SA، بر روی یک کلید اشتراکی مذاکره می‌کنند، لذا از این کلید اشتراکی در محاسبه‌ی امضا استفاده می‌شود. یک راه آسان عبارت‌است‌از محاسبه‌ی متنِ درهم، بر روی بسته به علاوه‌ی کلید اشتراکی. البته کلید اشتراکی منتقل نمی‌گردد. روشی مشابه این، **HMAC** (**کد تصدیق هویتِ پیغامِ درهم‌سازی شده**[1]) نامیده می‌شود. محاسبه‌ی این کد بسیار سریع‌تر از آن است که ابتدا SHA-1 اجرا شود و سپس RSA بر روی حاصلِ آن اجرا شود.

سرآیند AH اجازه‌ی رمزگذاری داده را نمی‌دهد، لذا برای وقتی که کنترل جامعیت لازم است ولی نیازی به رازپوشی وجود ندارد، بسیار سودمند می‌باشد. یک ویژگی جالب توجه در AH آن است که کنترل جامعیت، بعضی فیلدها در سرآیند IP را می‌پوشانَد، یعنی آن فیلدهایی که در حین حرکت بسته از یک مسیریاب به مسیریابِ دیگر، تغییر نمی‌کنند. برای مثال، فیلد *Time to live* در هر پرش (hop) تغییر می‌کند، بنابراین نمی‌تواند در کنترل جامعیت به حساب بیاید. اما آدرس IP مبدأ در کنترل جامعیت لحاظ می‌شود و بدین ترتیب دستکاری در بسته‌ی اولیه را برای یک نفوذگر، غیرممکن می‌سازد.

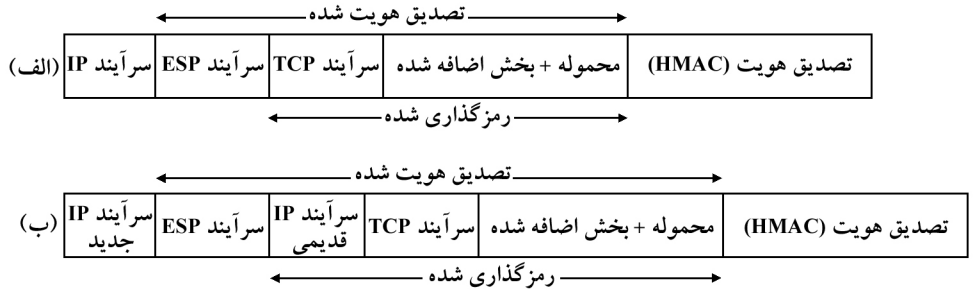
گزینه‌ی دیگر برای سرآیند IPsec عبارت‌است‌از **ESP** (**محصورسازیِ محموله‌ی امن**[2]). استفاده از آن در مود حمل و مود تونل در شکل ۸–۲۸ نشان داده شده است.

سرآیند ESP از دو کلمه‌ی ۳۲ـبیتی تشکیل شده است. این دو کلمه، فیلد *Security parameters index* و فیلد *sequencenumber* هستند که در AH آن‌ها را دیدیم. کلمه‌ی سومی که عمدتاً به دنبال این دو فیلد قرار دارد (اما به لحاظ فنی بخشی از سرآیند نیست) فیلد *Initialization vector* است که جهت رمزگذاری داده استفاده می‌شود مگر در رمزگذاریِ تهی (null) که در این حالت، این فیلد حذف می‌شود.

همچنین سرآیند ESP برای HMAC کنترل‌های جامعیت را نیز فراهم می‌کند، همان‌طور که AH این کار را انجام می‌دهد. اما به جای آن‌که این کنترل‌ها در سرآیند قرار گیرند، بعد از محموله می‌آیند که در شکل ۸–۲۸ نشان داده شده است. قرار دادنِ HMAC در انتها دارای یک مزیت هنگام پیاده‌سازی سخت‌افزاری است: HMAC می‌تواند در حینی که بیت‌ها از طریق رابط شبکه به بیرون می‌روند، محاسبه شده و به انتهای سرآیند اضافه گردند. به همین دلیل است که اترنت و LAN های دیگر CRCهایشان را به جای سرآیند، در پسایندِ (trailer) خود دارند. با AH، بسته ناچار باید بافر شود و قبل از آن‌که بسته بتواند ارسال گردد، بایستی امضا محاسبه شود. این کار به صورت بالقوه تعداد بسته‌هایی که در هر ثانیه می‌توانند ارسال شوند را کاهش می‌دهد.

با فرض این‌که ESP بتواند هر کاری که AH می‌کند را انجام دهد و بسیار کارآمدتر از AH هم باشد، پس چرا کلاً زحمت داشتنِ AH را تقبل کنیم؟ جواب این پرسش به گذشته برمی‌گردد. در ابتدا AH فقط جامعیت را ارائه می‌داد و ESP نیز فقط رازپوشی را. بعدها جامعیت به ESP اضافه شد اما

1. Hashed Message Authentication Code 2. Encapsulating Security Payload



شکل ۸-۲۸ **(الف)** ESP در مود حمل. **(ب)** ESP در مود تونل.

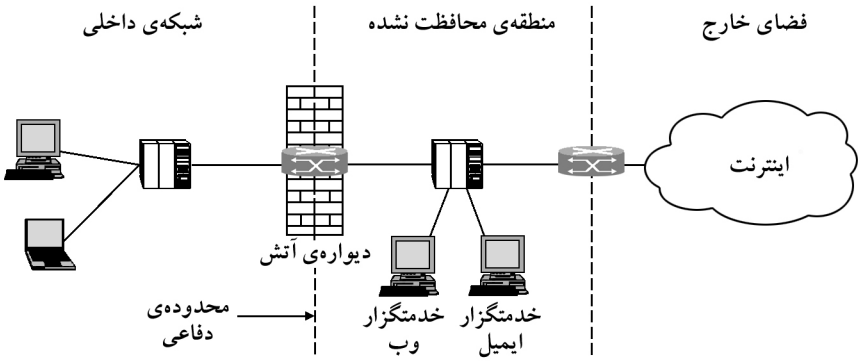
کسانی که AH را طراحی کرده بودند نمی‌خواستند اجازه دهند بعد از تمام کارهایی که کرده بودند، AH از بین برود. تنها استدلال واقعی آن‌ها این بود که AH بخشی از سرآیند IP را کنترل می‌کند، یعنی کاری که ESP انجام نمی‌دهد، با این تذکر که این موضوع در واقع یک استدلال ضعیف است. استدلال ضعیف دیگر این‌است‌که محصولی که از AH حمایت کند ولی از ESP حمایت نکند ممکن است دشواری کمتری در اخذ مجوز صادرات داشته باشد زیرا نمی‌تواند رمزگذاری را انجام دهد. این طور به نظر می‌رسد که در آینده ادامه‌ی کار AH متوقف شود.

۸-۶-۲ دیواره‌های آتش

امکان اتصال هر کامپیوتری، در هر جایی که باشد، به هر کامپیوتر دیگری، در هر جایی که باشد، یک نعمتِ مخلوط است. برای افرادی که در منزل هستند، گشت و گذار در اینترنت بسیار سرگرم‌کننده است. برای مدیران امنیتی شرکت‌ها، یک کابوس است. اغلبِ شرکت‌ها حجم زیادی اطلاعات محرمانه به صورت برخط دارند ــ اطلاعات محرمانه‌ی تجاری، نقشه‌های توسعه‌ی محصولات، راهبُردهای فروش، تحلیل‌های مالی، و امثال این‌ها. افشای این اطلاعات به یک رقیب می‌تواند نتایج وخیمی به همراه داشته باشد.

علاوه بر خطر درزکردنِ اطلاعات به بیرون، خطر درز کردنِ اطلاعات به داخل نیز وجود دارد. مشخصاً ویروس‌ها، کرم‌ها، و مزاحم‌های دیجیتالیِ دیگر می‌توانند امنیت را نقض کنند، داده‌های باارزش را تخریب کنند، و باعث شوند مدت زیادی از وقت سرپرستان بابت پاک‌کردنِ نابسامانی‌ای که این‌گونه مزاحمت‌ها از خود به جا گذاشته‌اند تلف گردد. غالباً این موارد توسط کارکنان بی‌دقت که می‌خواهند یک بازی جدید و جذاب را آزمایش کنند، وارد می‌شوند.

در نتیجه نیاز به مکانیزم‌هایی داریم تا بیت‌های "خوب" را نگه داشته و بیت‌های "بد" را بیرون کنیم. یک روش، استفاده از IPsec است. این رویکرد از داده هنگام گذار مابین سایت‌های امن، محافظت می‌کند. با این حال، IPsec برای حفاظت از ورود مزاحمانِ دیجیتالی و نفوذگرها به LAN شرکت، هیچ کاری نمی‌کند. برای آن‌که ببینیم این هدف چطور برآورده می‌شود باید دیواره‌های آتش را بررسی کنیم.



شکل ۸-۲۹ یک دیواره‌ی آتش در حال محافظت از یک شبکه‌ی داخلی.

دیواره‌های آتش[1] یک تطبیق امروزی از یک راه‌حل امنیتی قرون وسطایی است: یک خندق عمیق در اطراف زندان حفر کنید. این طراحی تمام افرادی که وارد یا خارج شوند را مجبور می‌کند از یک پل متحرکِ منفرد عبور کنند، جایی که می‌توانند توسط پلیس I/O بازرسی شوند. در مورد شبکه‌ها نیز همین ترفند امکان‌پذیر است: یک شرکت می‌تواند تعداد زیادی LAN داشته باشد که به روش‌های دلخواهی به هم متصل هستند، اما همان‌طور که در شکل ۸-۲۹ نشان داده شده، تمام ترافیک به یا از شرکت اجباراً از طریق یک پل متحرکِ الکترونیکی (دیواره‌ی آتش) عبور می‌کند. هیچ مسیر دیگری وجود ندارد.

دیواره‌ی آتش به عنوان یک **فیلتر بسته**[2] عمل می‌کند. دیواره‌ی آتش هر بسته‌ای که وارد یا خارج می‌شود را بازبینی می‌کند. بسته‌هایی که ضوابطِ تعیین شده در قوانینِ فرموله شده از سوی سرپرست شبکه را مراعات کنند، به صورت عادی پیش رانده می‌شوند. بسته‌هایی که در این آزمون شکست بخورند، قطعاً ساقط می‌شوند.

معیار فیلتر کردن معمولاً به صورت قوانین یا جدول‌هایی داده می‌شود که این اطلاعات در آن‌ها فهرست شده است: مبدأها و مقصدهای قابل پذیرش، مبدأها و مقصدهایی که مسدود شده‌اند، و یک سری قوانین پیش‌فرض درباره‌ی اقداماتی که در برابر بسته‌های وارده از ماشین‌های دیگر یا خارجه به ماشین‌های دیگر باید انجام داد. در حالت کلیِ مربوط به برپاسازیِ TCP/IP، یک مبدأ یا مقصد ممکن است از یک آدرس IP و یک پورت تشکیل شده باشد. پورت‌ها مشخص کننده‌ی سرویسِ مورد نظر هستند. برای مثال پورت TCP شماره‌ی ۲۵ برای ایمیل است، و پورت TCP شماره‌ی ۸۰ برای HTTP. بعضی پورت‌ها به سادگی می‌توانند مسدود شوند. به عنوان مثال یک شرکت می‌تواند بسته‌های رسیده به ازای تمام آدرس‌های IP که با پورت TCP شماره‌ی ۷۹ ترکیب شده‌اند را مسدود نماید. این کار یک زمانی در ارتباط با سرویس Finger (برای جستجوی آدرس ایمیل افراد) مرسوم بود ولی امروزه کمتر از آن استفاده می‌شود.

1. Firewall 2. Packet filter

سایر پورت‌ها به این سادگی مسدود نمی‌شوند. دشواری در این‌جاست که سرپرستان شبکه خواهان امنیت هستند ولی نمی‌توانند ارتباط با دنیای بیرون را قطع کنند. البته قطع ارتباط با دنیای بیرون از نظر امنیتی بسیار ساده‌تر و بهتر خواهد بود ولی اعتراضاتِ کاربران نسبت به آن، نهایتی ندارد. در چنین حالتی است که آنچه در شکل ۸-۲۹ با عنوان **DMZ** (**منطقه‌ی حافظت نشده**[1]) آمده، به درد می‌خورَد. بخش DMZ قسمتی از شبکه‌ی شرکت است که خارج از محدوده‌ی امنیتی قرار می‌گیرد. در این‌جا هر چیزی می‌تواند عبور کند. با قرار دادنِ یک ماشین، مثلاً یک خدمتگزار وب در DMZ، کامپیوترهایی که روی اینترنت هستند می‌توانند با آن تماس گرفته و سایت وبِ شرکت را مرور کنند. حالا دیواره‌ی آتش می‌تواند برای مسدود کردنِ ترافیک TCP ورودی به پورت ۸۰ به گونه‌ای پیکربندی شود که کامپیوترهایی که روی اینترنت هستند نتوانند از این پورت برای حمله به کامپیوترهای شبکه‌ی داخلی استفاده کنند. برای آن‌که خدمتگزار وب را بتوانیم مدیریت کنیم، دیواره‌ی آتش می‌تواند قانونی داشته باشد که اجازه‌ی اتصال میان ماشین‌های داخلی و خدمتگزار وب را بدهد.

دیواره‌های آتش به مرور زمان و در یک مسابقه‌ی مُچ‌اندازی با مهاجم‌ها، بسیار پیچیده‌تر شده‌اند. در ابتدا دیواره‌های آتش یک مجموعه‌ی قوانین را به صورت مستقل برای هر بسته اِعمال می‌کردند، اما ثابت شد که نوشتنِ قوانینی که عملکرد مناسب را اجازه بدهند ولی در عین حال کل ترافیکِ ناخواسته را نیز مسدود کنند، دشوار است. **دیواره‌های آتشِ حالت‌مند**[2] بسته‌ها را به اتصالات نگاشت کرده و از فیلدهای سرآیند TCP/IP برای محافظت از اتصالات استفاده می‌کنند. این کار بعضی قوانین مثل این را اجازه می‌دهد: به یک خدمتگزار وبِ خارجی اجازه بده بسته‌هایی را به یک میزبانِ داخلی ارسال کند، اما فقط درصورتی‌که میزبان داخلی ابتدا اتصالی را با خدمتگزار وبِ خارجی برقرار کرده باشد. چنین قانونی در طرح‌های غیرحالت‌مند[3] امکان‌پذیر نمی‌باشد. در این‌گونه طرح‌ها، بسته‌های خدمتگزار وبِ خارجی بایستی یا عبور کنند و یا ساقط شوند.

سطح دیگری از پیچیدگی ناشی از پردازش حالت‌مند، مربوط است به این‌که دیواره‌ی آتش **دروازه‌های سطح کاربرد**[4] را پیاده‌سازی کند. این پیاده‌سازی مستلزم آن است که دیواره‌ی آتش درون بسته‌ها را نگاه کند، حتی فراتر از سرآیند TCP را، تا ببیند آن کاربرد چه کاری انجام می‌دهد. با این قابلیت، این امکان وجود دارد که میان ترافیک HTTP که برای مرور وب استفاده می‌شود، با ترافیک HTTP که برای اشتراک فایلِ همتا ـ به ـ همتا به کار می‌رود، تمایز ایجاد شود. سرپرستان شبکه می‌توانند قوانینی بنویسند تا شرکت را از اشتراک فایل به صورت همتا ـ به ـ همتا معاف کنند ولی اجازه‌ی مرور وب (که برای کسب و کار، حیاتی است) را بدهند. برای تمام این شیوه‌ها، ترافیک خروجی درست همانند ترافیک ورودی می‌تواند مورد بازبینی قرار بگیرد، تا مثلاً از این‌که اسناد حساس به خارج از شرکت ایمیل شوند، جلوگیری گردد.

1. DeMilitarized Zone 2. Stateful firewall 3. Stateless 4. Application-level gateway

گرچه بحث بالا بایستی روشن باشد، ولی همچنان دیواره‌های آتش پروتکل‌های لایه‌بندیِ استاندارد را نقض می‌کنند. دیواره‌های آتش ابزار لایه‌ی شبکه هستند، اما برای انجامِ فیلترینگ، دزدکی به لایه‌های حمل و کاربرد هم نگاه می‌کنند. همین موضوع آن‌ها را شکننده می‌کند. برای نمونه، دیواره‌های آتش می‌خواهند به توافق‌های مربوط به شماره‌گذاریِ استاندارد پورت تکیه کنند تا نوع ترافیکی که در یک بسته حمل می‌شود را تعیین کنند. اغلب اوقات از پورت‌های استاندارد استفاده می‌شود، ولی نه توسط تمام کامپیوترها و نه حتی توسط تمام کاربردها. بعضی از کاربردهای همتا–به–همتا پورت‌ها را به صورت پویا انتخاب می‌کنند تا از این‌که به آسانی مورد هدف قرار گرفته (و مسدود شوند) جلوگیری کرده باشند. رمزگذاری با IPsec یا نظام‌های دیگر، اطلاعاتِ لایه‌ی بالاتر را از دیواره‌ی آتش پنهان می‌سازد. و مورد آخر این‌که، یک دیواره‌ی آتش نمی‌تواند به سرعت با کامپیوترهایی که از طریق این دیواره ارتباط برقرار می‌کنند، گفتگو کند تا بتواند به آن‌ها بگوید که چه سیاست‌هایی اِعمال می‌شود و این‌که چرا اتصال آن‌ها ساقط می‌شود. بنا به تمام این دلایل، معمارانِ اصیل شبکه‌بندی، دیواره‌های آتش را به مثابه‌ی نقص در معماری اینترنت در نظر می‌گیرند. اما اگر شما یک کامپیوتر باشید، اینترنت می‌تواند برایتان جای خطرناکی باشد. دیواره‌های آتش در حل این مسئله یاری می‌کنند لذا به نظر می‌رسد که ماندنی هستند.

حتی اگر دیواره‌ی آتش بسیار به دقت پیکربندی شود، هنوز مسائل امنیتیِ فراوانی وجود خواهند داشت. برای مثال اگر یک دیواره‌ی آتش به نحوی پیکربندی شود که اجازه‌ی ورود بسته‌ها از شبکه‌های خاصی داده شود (مثلاً از شعبه‌های یک شرکت)، نفوذگری که خارج از دیواره‌ی آتش است می‌تواند آدرس‌های مبدأ دروغینی قرار دهد تا از این کنترل عبور کند. اگر یکی از افرادِ خودی بخواهد اسناد محرمانه را خارج کند، می‌تواند آن‌ها را رمزگذاری کرده و یا حتی از آن‌ها عکس بگیرد و عکس‌ها را به شکل فایل‌های JPEG خارج کند، یعنی هر گونه فیلتری بر روی ایمیل را دور بزنَد. و البته هنوز این حقیقت را نگفته‌ایم که هرچند سه ـ چهارمِ تمام حمله‌ها از خارج از دیواره‌ی آتش شکل می‌گیرند، ولی حمله‌هایی که داخل دیواره‌ی آتش صورت می‌گیرند (مثلاً از طرف کارکنانی که دلخوری‌هایی دارند) بیشترین تخریب‌ها را به همراه دارند (Verizon، ۲۰۰۹).

یک مسئله‌ی متفاوت در ارتباط با دیواره‌های آتش این است که این ابزار یک دفاع پیرامونیِ منفرد را فراهم می‌کنند. اگر این دفاع شکسته شود، تمام تمهیدات از میان می‌روند. به همین دلیل دیواره‌های آتش غالباً در یک دفاع لایه‌ای استفاده می‌شوند. به عنوان مثال یک دیواره‌ی آتش می‌تواند ورود به شبکه‌ی داخلی را محافظت کند و هر کامپیوتر نیز دیواره‌ی آتشِ خودش را اجرا نماید. افرادی که در حال مطالعه‌ی این کتاب هستند درصورتی‌که فکر می‌کنند یک نقطه‌ی کنترلِ امنیتی کفایت می‌کند، حتماً اخیراً یک پرواز بین‌المللی در یکی از خطوط هواییِ برنامه‌ریزی شده، نداشته‌اند.

علاوه بر این، یک کلاس کاملاً جداگانه از حمله‌ها نیز وجود دارد که دیواره‌های آتش توانایی برخورد با آن را ندارند. ایده‌ی اصلیِ یک دیواره‌ی آتش این‌است‌که از ورود نفوذگرها و از خروج

داده‌های سرّی جلوگیری کند. متأسفانه افرادی وجود دارند که هیچ کاری را بهتر از تلاش برای ضربه زدن به سایت‌های خاص، انجام نمی‌دهند. آن‌ها این کار را با ارسال تعداد زیاد بسته‌های قانونی و موجه به مقصد انجام می‌دهند، تا آن‌که زیر بار زیاد، متلاشی شود. برای مثال، یک نفوذگر به منظور فلج نمودنِ یک سایت وب، می‌تواند یک بسته‌ی TCP *SYN* برای برقراری یک اتصال ارسال کند. در این صورتِ سایت یک جای خالی در جدول برای اتصال تخصیص می‌دهد و در پاسخ، یک بسته‌ی $SYN + ACK$ ارسال می‌کند. اگر نفوذگر پاسخ ندهد، جای خالی در جدول تا چند ثانیه باقی می‌مانَد تا مهلت زمانی سر برسد. اگر نفوذگر هزاران درخواست اتصال ارسال کند، تمام جاهای خالی در جدول اِشغال می‌شوند و هیچ اتصالِ قانونی‌ای نمی‌تواند پذیدفته شود. حمله‌هایی که در آن‌ها، نفوذگر به جای سرقت داده‌ها، قصد تعطیل کردنِ سایتِ هدف را داشته باشد، حمله‌های **DoS** (**محروم‌سازی از سرویس**[1]) نامیده می‌شوند. معمولاً بسته‌های درخواست، دارای آدرس‌های مبدأِ دروغین هستند و بنابراین نفوذگر به راحتی قابل پیگرد نمی‌باشد. حمله‌های DoS در برابر سایت‌های وبِ مهم، در اینترنت رایج هستند.

یک موردِ حتی بدتر، حمله‌ای است که که در آن، نفوذگر بر روی صدها کامپیوتر در همه جای دنیا تکه ـ تکه و پخش شده باشد و سپس به همه‌ی آن‌ها فرمان داده شود که در یک زمانِ واحد به یک هدفِ واحد حمله کنند. چنین روشی نه تنها قدرت آتشِ نفوذگر را افزایش می‌دهد، بلکه شانس کشف مورد را نیز کاهش می‌دهد زیرا بسته‌ها از طرف تعداد زیادی ماشین می‌آیند که این ماشین‌ها به کاربرانِ غیرمشکوک تعلق دارند. چنین حمله‌ای حمله‌ی **DDoS** (**محروم‌سازی از سرویس به صورت توزیع شده**[2]) نامیده می‌شود. دفاع در برابر این حمله کار دشواری است. حتی اگر ماشینی که مورد حمله قرار گرفته بتواند به سرعت دروغین بودنِ درخواست را تشخیص دهد، باز هم مدتی طول می‌کشد تا درخواست را پردازش کرده و آن را از بین ببرد، و اگر تعداد درخواست‌هایی که در هر ثانیه می‌رسند به قدر کافی باشند، CPU تمام وقتش را صرف برخورد با آن‌ها خواهد کرد.

۸-۶-۳ شبکه‌های خصوصیِ مَجازی

بسیاری از شرکت‌ها دفاتر و کارگاه‌هایی دارند که در شهرهای متعدد، و گاهی در چند کشور پراکنده هستند. در ایام قدیم، پیش از شبکه‌های داده‌ایِ عمومی، چنین شرکت‌هایی عموماً خطوطی را از شرکت تلفن اجاره می‌کردند که مابین بعضی از جفت ـ محل‌ها (pairs of locations) و یا مابین تمام جفت ـ محل‌ها قرار داشت. هنوز هم بعضی شرکت‌ها به همین شیوه عمل می‌کنند. شبکه‌ای که از کامپیوترهای شرکت و خطوط استیجاری تلفن تشکیل شده باشد، **شبکه‌ی خصوصی**[3] نامیده می‌شود.

شبکه‌های خصوصی به خوبی کار می‌کنند و بسیار امن هستند. اگر تنها خطوطِ قابل دسترسی، خطوط استیجاری باشند، هیچ ترافیکی نمی‌تواند به خارج از محل‌های مربوط به شرکت، نشت کند و

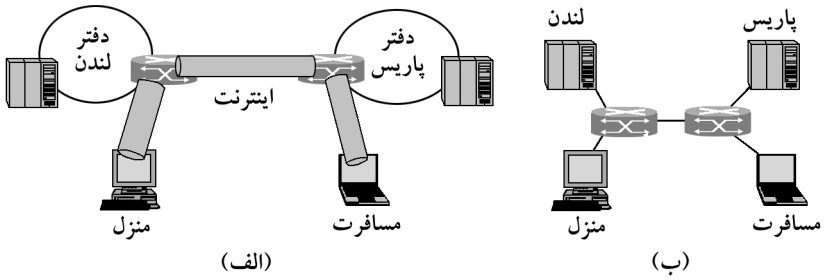
1. Denial of Service 2. Distributed Denial of Service 3. Private network

نفوذگران مجبور هستند با روش‌های فیزیکی خطوط تلفن را استراق سمع کنند که البته کار آسانی نیست. مشکلِ شبکه‌های خصوصی آن است که اجاره کردنِ یک خط اختصاصیِ T1 میان دو نقطه، هزاران دلار در ماه هزینه دارد، و خطوط T3 چندین برابر گران‌تر هستند. هنگامی‌که شبکه‌های داده‌ی عمومی، و بعدتر اینترنت، پدیدار شدند بسیاری از شرکت‌ها تمایل پیدا کردند که ترافیکِ داده‌هایشان (و احتمالاً ترافیک داده‌ی صوتی‌شان) به شبکه‌ی عمومی منتقل گردد، ولی البته بدون از دست دادنِ امنیتِ شبکه‌های خصوصی.

این نیازمندی خیلی زود منجر به ابداع **VPN**ها گردید (**شبکه‌های خصوصی مجازی**). این شبکه‌ها عبارتند از شبکه‌های هم‌پوشان[1] بر فراز شبکه‌های عمومی، اما همراه با اغلبِ ویژگی‌های شبکه‌های خصوصی. آن‌ها را "مَجازی" می‌نامیم زیرا این شبکه‌ها فقط یک اِنگاره[2] هستند، درست همانند مدارهای مجازی که مدارهای واقعی نیستند یا همانند حافظه‌ی مجازی که یک حافظه‌ی واقعی نیست.

یک رویکرد عامه‌پسند عبارت‌است‌از این‌که VPNها را مستقیماً بر فراز اینترنت بسازیم. طراحی رایج آن است که هر دفتر کاری را به یک دیواره‌ی آتش مجهز کنیم و تونل‌هایی از میان اینترنت مابین تمام جفت ـ دفاتر بسازیم (همان‌طور که در شکل ۸-۳۰(الف) نشان داده شده). مزیت دیگری که استفاده از اینترنت برای برقراری ارتباط دارد آن است که تونل‌ها می‌توانند بر اساس نیاز برپا شوند تا مثلاً شامل کامپیوترِ یک کارمند شوند که مثلاً در منزل یا در مسافرت است (مادام که آن شخص یک اتصال به اینترنت داشته باشد). به این ترتیب این انعطاف‌پذیری بسیار بیشتر از آن چیزی است که با خطوط استیجاری فراهم می‌شود. ولی همان‌طور که در شکل ۸-۳۰(ب) نشان داده شده، توپولوژیِ شبکه از دید کامپیوترهایی که بر روی VPN هستند مانند شبکه‌ی خصوصی به نظر می‌رسد. هنگامی‌که سیستم بالا می‌آید، هر جفت از دیواره‌های آتش بایستی درباره‌ی پارامترهای SA خود، مذاکره کنند. این پارامترها شامل سرویس‌ها، مودها، الگوریتم‌ها، و کلیدها هستند. اگر برای تونل زدن از IPsec استفاده شود، این امکان وجود دارد که تمام ترافیکی که مابین دو جفت دفتر برقرار می‌شوند، بر روی یک SA واحد که تصدیق هویت و رمزگذاری شده است، گردآوری شود. این امر امکانِ کنترل مجتمع، رازپوشی، و حتی مصونیتِ قابل توجه در برابر تحلیل ترافیک را فراهم می‌کند. بسیاری از دیواره‌های آتش دارای قابلیت‌های VPN به صورت توکار می‌باشند. بعضی مسیریاب‌های عادی نیز توان انجام این کار را دارند اما چون در اصل دیواره‌های آتش وظیفه‌ی امنیت را بر عهده دارند، داشتنِ تونل‌هایی که آغاز و خاتمه‌ی آن‌ها در دیواره‌های آتش باشد، امری طبیعی است تا بدین‌وسیله یک تمایز شفاف میان شرکت و اینترنت ایجاد شود. بنابراین دیواره‌های آتش، VPNها، و IPsec همراه با ESP در مود تونل، ترکیباتی طبیعی هستند که در عمل از آن‌ها استفاده‌ی زیادی می‌شود.

1. Overlay network 2. Illusion



شکل ۸-۳۰ (الف) یک شبکه‌ی خصوصیِ مجازی. (ب) توپولوژی شبکه به آن صورتی که از درون قابل مشاهده می‌شود.

با برقراری SAها، جریان ترافیک می‌تواند شروع شود. از نظر یک مسیریابِ درون اینترنت، یک بسته که در حال سفر در یک تونل VPN می‌باشد، فقط یک بسته‌ی معمولی است. تنها مورد غیرعادی درباره‌ی این بسته عبارت‌است‌از وجود یک سرآیند IPsec بعد از سرآیند IP، اما چون این سرآیندهای اضافی هیچ اثری بر پروسه‌ی پیش راندن ندارند، لذا مسیریاب‌ها به این سرآیند اضافی اهمیتی نمی‌دهند.

رویکرد دیگری که محبوبیتی به دست آورده است، برپاسازی VPN توسط ISP می‌باشد. با استفاده از MPLS (که در فصل ۵ شرح داده شده)، مسیرهای مربوط به ترافیک VPN می‌توانند در طول شبکه‌ی ISP، میان دفاتر شرکت برپا شوند. این مسیرها ترافیک VPN را جدا از بقیه‌ی ترافیک اینترنت نگه می‌دارند و می‌توانند مقدار مشخصی پهنای باند و یا موردی از کیفیت سرویس را ضمانت کنند.

یک مزیت کلیدیِ VPN آن است که برای تمام نرم‌افزارهای کاربر کاملا نامرئی[1] است. دیواره‌های آتش SAها را برپا و مدیریت می‌کنند. تنها فردی که دقیقاً از این برپاسازی مطلع می‌باشد، سرپرست سیستم است که باید دروازه‌های امنیتی را پیکربندی و مدیریت کند، یا سرپرست ISP که بایستی مسیرهای MPLS را پیکربندی کند. برای بقیه‌ی افراد، داستان مثل این است که باز هم یک شبکه‌ی خصوصی خطوط استیجاری داشته باشیم. برای اطلاعات بیشتر درباره‌ی VPNها به Lewis (۲۰۰۶) مراجعه نمایید.

۸-۶-۴ امنیت بی‌سیم

طراحی یک سیستم با استفاده از VPNها و دیواره‌های آتش که به لحاظ منطقی کاملاً امن باشد، به طرز شگفت‌انگیزی آسان است ولی این ایده در عمل نشتی‌هایی دارد! اگر بعضی ماشین‌ها بی‌سیم بوده و از ارتباطات رادیویی استفاده کنند، این ارتباطات درست از بالای دیواره‌ی آتش (آن هم در هر دو جهت) عبور می‌کنند. محدوده‌ی عمل شبکه‌های 802.11 غالباً چند صد متر است، بنابراین کسی که بخواهد از

1. Transparent

یک شرکت جاسوسی کند می‌تواند به راحتی در محل توقف اتومبیل‌های شرکت پارک کند و یک کامپیوتر نوت‌بوک را داخل اتومبیل به حال خود رها کند تا هر آنچه را که می‌شنود ضبط نماید. تا عصر، دیسک مملو از اطلاعات ارزشمند خواهد شد. به لحاظ نظری چنین نشتی حدس زده نمی‌شد.

ردّ بسیاری از مسائل امنیتی را می‌توان تا سازندگان ایستگاه‌های پایه‌ی بی‌سیم (یعنی همان نقاط دسترسی) دنبال نمود. سازندگانی که تلاش دارند تا محصولاتشان کاربرـ پسند باشند. معمولاً اگر کاربر دستگاه را از جعبه‌اش خارج کرده و آن را به پریز برق وصل کند، بلافاصله شروع به کار خواهد کرد ــ و تقریباً همیشه بدون رعایت هیچ امنیتی، تمام اسرار را به هر کسی که در محدوده‌ی رادیویی دستگاه باشد، می‌گوید. در این حالت اگر به یک اترنت وصل شود، به یک باره کل ترافیک اترنت در پارکینگ نیز نمایان می‌شود. بی‌سیم، تحقق یکی از رؤیاهای جاسوسان است: داده‌ی مجانی و بدون زحمت. پس نیازی به ذکر این موضوع نیست که امنیت در سیستم‌های بی‌سیم، حتی از سیستم‌های باسیم نیز مهم‌تر است. در این رابطه اطلاعات بیشتری توسط Nichols و Lekkas (۲۰۰۲) ارائه شده است.

امنیت در 802.11

قسمتی از استاندارد 802.11 که در ابتدا **802.11i** نامیده می‌شد، اختصاص دارد به توصیه درباره‌ی یک پروتکل امنیتی در سطح پیوند داده، جهت ممانعت از این‌که یک گره بی‌سیم، پیغام‌های ارسال شده میان جفت‌ـگره‌های بی‌سیم دیگر را بخواند یا مورد مداخله قرار دهد. نام آن **WPA2** (**دسترسی حافظت شده به WiFi ـ ۲**[1]) می‌باشد. مدل WPA ساده، یک مدل موقتی است که یک زیرمجموعه از 802.11i را پیاده‌سازی می‌کند. در برابر WPA2 بایستی از WPA صرفنظر شود.

به زودی 802.11i را بررسی خواهیم کرد اما ابتدا متذکر می‌شویم که 802.11i یک جایگزین برای **WEP** (**محرمانگی معادل باسیم**[2]) می‌باشد، یعنی نسل اول از پروتکل‌های امنیتی 802.11 . پروتکل WEP به وسیله‌ی یک کمیته‌ی استانداردهای شبکه‌بندی طراحی گردید. این پروسه مثلاً نسبت به روشی که NIST طراحی AES را برگزید، کاملاً متفاوت است. نتایج فاجعه‌آمیز بود. کجای کار اشتباه بود؟ آن‌گونه که بعداً معلوم شد، همه چیز اشکال داشت. برای مثال، WEP برای حفظ محرمانگی، داده را با XOR کردن آن با خروجی یک رمز جریانی رمزگذاری می‌کرد. متأسفانه آرایش‌های ضعیف در کلیدگذاری، به معنای آن بود که در اغلب اوقات، خروجی مورد استفاده‌ی مجدد قرار می‌گرفت. این امر باعث شد با روش‌های پیش پا افتاده شکسته شود. مثلاً یک اشکال دیگر این بود که کنترل جامعیت بر مبنای CRC ـ ۳۲ بیتی انجام می‌شد. این کد، یک کد کارآمد برای تشخیص خطاهای انتقال می‌باشد اما یک مکانیسم قوی رمزنگاری برای شکست دادن مهاجم‌ها نیست.

این موارد و کاستی‌های دیگری در طراحی، باعث شد رقابت با WEP آسان باشد. نخستین اثبات تجربی که WEP در آن شکست خورد، زمانی پیش آمد که آدام استابل فیلد[3] در AT&T دستیار

1. WiFi Protected Access 2 2. Wired Equivalent Privacy 3. Adam Stublefield

بود (Stubblefield و همکاران، ۲۰۰۲). آدام توانست حمله‌ای که توسط Fluhrer و همکاران (۲۰۰۱) طراحی شده بود را ظرف یک هفته کد کرده و آزمایش نماید. که البته از این مدتِ یک هفته‌ای، بیشترین زمان صرفِ این شد که مدیریت را متقاعد کند تا برایش یک عدد کارت WiFi بخرند تا در آزمایشاتش از آن استفاده کند. اکنون نرم‌افزاری که ظرف یک دقیقه گذرواژه‌های WEP را به صورت غیرقانونی باز می‌کند، به صورت رایگان در دسترس است و استفاده از WEP شدیداً منع می‌شود. هرچند WEP از دسترسی تفننی جلوگیری می‌کند، ولی یک امنیت حقیقی را تأمین نمی‌کند. هنگامی‌که روشن شد WEP جداً شکست خورده است، گروه 802.11i با دستپاچگی گرد هم آمدند. این گروه تا قبل از ژوئن ۲۰۰۴ یک استاندارد رسمی ارائه کرد.

اکنون 802.11i را شرح خواهیم داد که اگر به طرزی مناسب برپا شده و به کار گرفته شود، امنیت واقعی را فراهم خواهد کرد. دو سناریوی متداول برای استفاده از WPA2 وجود دارند. اولین سناریو مربوط به یک محیط شرکتی است. در این سناریو، یک شرکت دارای یک خدمتگزار مجزا برای تصدیق هویت است که یک پایگاه داده برای نام کاربری و گذرواژه دارد. از این خدمتگزار می‌توان برای تعیین این‌که آیا یک مشتریِ بی‌سیم اجازه‌ی دسترسی به شبکه را دارد یا خیر، استفاده نمود. مشتری‌ها در این محیط از پروتکل‌های استاندارد استفاده می‌کنند تا هویت خویش را به شبکه بشناسانند. استانداردهای اصلی عبارتند از **802.1X** که از طریق آن‌ها نقطه‌ی دسترسی به مشتری امکانِ اجرای یک دیالوگ با خدمتگزارِ تصدیق هویت، و مشاهده‌ی نتیجه‌ی کار را می‌دهد. و همچنین استاندارد **EAP** (**پروتکل تصدیق هویتِ گسترش‌پذیر**[1]) (RFC 3748) که چگونگی تعامل میان مشتری و خدمتگزارِ تصدیق هویت را مشخص می‌کند. در اصل، EAP یک چارچوب است و سایر استانداردها، پیغام‌های پروتکل را تعریف می‌کنند. ولی در این موضوعات، زیاد به کندوکاو جزئیات نمی‌پردازیم زیرا این مسائل چندان به مرور کلیِ ما مربوط نیست.

سناریوی دوم در یک محیط خانگی است، یعنی محیطی که در آن هیچ خدمتگزارِ تصدیق هویتی وجود ندارد. بلکه یک گذرواژه‌ی اشتراکیِ منفرد وجود دارد که مشتری‌ها برای دسترسی به شبکه‌ی بی‌سیم از آن استفاده می‌کنند. در این‌جا پیچیدگی کمتر از حالتی است که خدمتگزارِ تصدیق هویت داشتیم، به همین دلیل هم این روش در منزل و در کسب و کارهای کوچک استفاده می‌شود، اما امنیتِ آن هم کمتر است. تفاوت اصلی در این است که با یک خدمتگزارِ تصدیق هویت، هر مشتری یک کلید برای رمزگذاری ترافیک می‌گیرد که سایر مشتری‌ها از آن اطلاع ندارند. هنگامی‌که یک گذرواژه‌ی اشتراکیِ منفرد داشته باشیم، کلیدهای متفاوتی برای هر یک از کاربران به دست می‌آیند اما همه‌ی مشتری‌ها یک گذرواژه دارند و اگر بخواهند می‌توانند به کلیدهای همدیگر پی ببرند.

1. Extensible Authentication Protocol

کلیدهای مورد استفاده برای رمزگذاریِ ترافیک، به عنوان بخشی از یک عمل تصدیق هویت[1] (یا دست‌دادنِ طرفین جهت تصدیق هویت)، محسوب می‌شوند. عمل "دست دادن" زمانی به درستی انجام می‌شود که مشتری با یک شبکه‌ی بی‌سیم مرتبط گردد و توسط یک خدمتگزارِ تصدیق هویت (البته در صورت وجودِ چنین خدمتگزاری) تأیید شود. در آغازِ عملِ "دست دادن"، مشتری یا گذرواژه‌ی شبکه‌ی اشتراکی را در اختیار دارد یا گذرواژه‌ی خودش برای خدمتگزارِ تصدیق هویت را دارد. از این گذرواژه جهت به دست آوردنِ یک کلید اصلی (master key) استفاده می‌شود. با این وجود از کلید اصلی به طور مستقیم در رمزگذاری بسته‌ها استفاده نمی‌شود. رفتار استاندارد در رمزگذاری این است که برای هر دور استفاده، یک کلیدِ نشست به دست آورده شود، این کلید برای نشست‌های مختلف عوض شود، و کلید اصلی هر چقدر ممکن است کمتر در معرض دید قرار داده شود. این همان کلیدِ نشستی است که در عملِ "دست دادن"، محاسبه می‌شود.

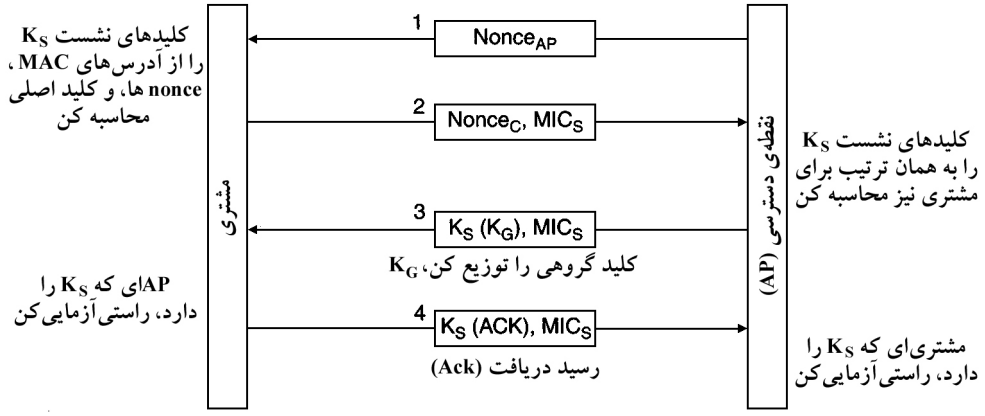
کلید نشست با یک "دست دادنِ" چهار- بسته‌ای، که در شکل ۸-۳۱ نشان داده شده، محاسبه می‌شود. ابتدا AP (نقطه‌ی دسترسی) یک عدد تصادفی برای شناسایی ارسال می‌کند. اعداد تصادفی‌ای که مانند این مورد، فقط یک بار در پروتکل‌های امنیتی به کار می‌روند **nonce** (نانس)[2] نامیده می‌شوند. می‌توان گفت که نانس مخفف "عددی که یک بار استفاده می‌شود" می‌باشد. مشتری نیز نانسِ خودش را انتخاب می‌کند. مشتری از نانس‌ها، آدرس MAC خودش و آدرس MAC مربوط به AP، و کلید اصلی استفاده می‌کند تا یک کلید نشست به نام K_s را محاسبه نماید. کلید نشست به چند قسمت تقسیم می‌شود و هر یک از قسمت‌ها برای منظورهای متفاوتی استفاده می‌شوند. از جزئیاتِ این موضوع صرفنظر می‌کنیم. اکنون مشتری کلیدهای نشست را در اختیار دارد، اما AP خیر. بنابراین مشتری نانسِ خودش را برای AP ارسال می‌کند، و AP همین محاسبات را انجام می‌دهد تا همین کلیدهای نشست را به دست آورَد. چون کلیدها بدون اطلاعات سرّیِ بیشتر، قابل استحصال نیستند لذا نانس‌ها می‌توانند به صورت آشکار ارسال شوند. پیغام ارسالی از طرف مشتری با انجام یک کنترلِ جامعیت به نام **MIC** (**کنترل جامعیتِ پیغام**[3]) که مبتنی بر کلید نشست می‌باشد، محافظت می‌شود. صحت MIC را AP می‌تواند کنترل کند، و لذا پیغام حقیقتاً باید از طرف مشتری آمده باشد، سپس AP کلیدهای نشست را محاسبه می‌کند. شبیه آنچه در HMAC داشتیم، یکMIC در واقع نام دیگری برای یک کد تصدیق هویتِ پیغام[4] است. از اصطلاح MIC در بیشتر مواقع به جای پروتکل‌های شبکه‌بندی استفاده می‌شود زیرا امکانِ اشتباه گرفتنِ آن با MAC (کنترل دسترسی به رسانه) وجود دارد.

---

1. Authentication handshake

۲. Nonce : در علم مهندسیِ امنیت، نانس عبارت است از یک عدد دلخواه که تنها یک بار در یک ارتباط رمزنگاری استفاده می‌شود. غالباً یک عدد تصادفی یا شبه‌تصادفی است و در پروتکل‌های تصدیق هویت صادر می‌شود تا مطمئن شویم ارتباط‌های قبلی نتوانند توسط حمله‌ی راه‌اندازیِ مجدد مورد استفاده قرار گیرند (مترجم).

3. Message Integrity Check 4. Message authentication code



شکل ۸-۳۱ عملیات دست دادن در برپاسازیِ کلید 802.11i .

در دو پیغامِ آخر، AP یک کلیدِ گروهیِ K_G را به سمت مشتری توزیع می‌کند، و مشتری این پیغام را ack می‌کند. دریافتِ این پیغام‌ها به مشتری این امکان را می‌دهد که AP را راستی‌آزمایی کند، یعنی آیا AP کلیدهای صحیحِ نشست را دارد، و مواردی از این قبیل. از کلیدِ گروهی برای ترافیک همه‌پخشی و چندپخشی بر روی 802.11 LAN استفاده می‌شود. چون حاصلِ عمل دست دادن، آن است که هر مشتری کلیدهای رمزگذاریِ مربوط به خودش را داشته باشد بنابراین هیچ یک از این کلیدها نمی‌توانند از طرف AP به منظور همه‌پخشی بسته‌ها به تمام مشتری‌های بی‌سیم استفاده شوند؛ به عبارت دیگر ضرورت دارد که با استفاده از این کلید، کپیِ مجزایی به هر مشتری ارسال گردد. به جای این کار، یک کلیدِ اشتراکی پخش می‌شود زیرا به این ترتیب ترافیک همه‌پخشی می‌تواند تنها یک بار ارسال شود و تمام مشتری‌ها می‌توانند آن را دریافت کنند. این کلید با تغییراتِ ناشی از ترک کردنِ مشتری‌ها و پیوستنِ مشتری‌ها به شبکه، بایستی به‌روزرسانی شود.

در انتهای بحث، به این موضوع می‌رسیم که کلیدها عملاً برای تأمین امنیت مورد استفاده قرار می‌گیرند. در 802.11i از دو پروتکل می‌توان برای تأمین محرمانگی، جامعیت، و تصدیق هویتِ پیغام استفاده نمود. یکی از این پروتکل‌ها به نام **TKIP** (**پروتکل جامعیتِ کلید موقت**[1])، مشابه آنچه در مورد WPA دیدیم، یک راه‌حل موقتی بود. علت طراحی این پروتکل آن بود که امنیت بر روی کارت‌های 802.11 قدیمی و کُند، بهبود یابَد به طوری که به لحاظ ارتقاءی ثابت‌افزار[2]، امنیت دست کم بهتر از امنیتی بشَود که توسط WEP می‌تواند تأمین گردد. با این حال، هم‌اکنون این پروتکل نیز شکسته شده است به طوری که بهتر است آن را با پروتکل دیگری که توصیه شده، یعنی **CCMP**، جایگزین نمایید. این حروف نشان دهنده‌ی چه عبارتی هستند؟ این حروف، اختصاری برای یک نام

1. Temporary Key Integrity Protocol

۲. Firmware : در **الکترونیک** و **کامپیوتر**، اغلب به برنامه‌های تقریباً ثابت و نسبتاً کوچک و یا ساختمان‌های داده‌ای که درون **سخت‌افزار** انواع دستگاه‌های الکترونیک است، گفته می‌شود (مترجم).

جالب است: **Counter Mode Cipher Block Chaining Message Authentication Code Protocol** (پروتکل کد تصدیق هویتِ پیغام با زنجیره‌ی بلوک رمزی به همراه مودِ شمارنده)، که آن را همان CCMP می‌نامیم. می‌توانید آن را به هر نامی که مایل هستید بنامید!

پروتکل CCMP با یک روش منصفانه و روشن عمل می‌کند. این پروتکل از رمزگذاری AES همراه با یک کلید ۱۲۸ـ بیتی و اندازه‌ی بلوک، استفاده می‌کند. کلید، از کلید نشست می‌آید. برای تأمین محرمانگی، پیغام‌ها با AES و به روش مودِ شمارنده (counter mode) رمزگذاری می‌شوند. مودهای رمزی را در بخش ۸-۲-۳ بررسی کرده‌ایم. این مودها از این‌که پیغام هر بار به همان مجموعه‌ی بیت رمزگذاری شود، جلوگیری می‌کنند. مودِ شمارنده، یک شمارنده را به داخل پیغامِ رمزگذاری شده، مخلوط می‌کند. به منظور تأمین جامعیت، پیغام (مشتمل بر فیلدهای سرآیند) همراه با مود زنجیره‌ایِ بلوک رمزی، رمزگذاری می‌شود و آخرین بلوک ۱۲۸ـ بیتی به عنوان MIC نگهداری می‌شود. سپس هم پیغام (که در مود شمارنده رمزگذاری شده است) و هم MIC ارسال می‌شوند. مشتری و AP هر کدامشان قادر به انجام این رمزگذاری هستند، یا می‌توانند هنگامی‌که یک بسته‌ی بی‌سیم دریافت می‌شود، این رمزگذاری را راستی ـ آزمایی کنند. برای پیغام‌های همه‌پخشی یا چندپخشی از همین پروسه همراه با کلیدِ گروهی استفاده می‌شود.

امنیت در بلوتوث

بلوتوث محدوده‌ای به مراتب کوتاه‌تر از 802.11 دارد، لذا به آسانی نمی‌تواند مثلاً از داخل محوطه‌ی پارکینگ مورد حمله قرار گیرد اما در این‌جا نیز امنیت مهم است. برای مثال تصور کنید که کامپیوتر آلیس مجهز به یک کیبوردِ بی‌سیم بلوتوث می‌باشد. در نبودِ امنیت، اگر ترودی در دفتر مجاور باشد، می‌تواند هر چه آلیس تایپ می‌کند را ببیند، این شامل تمام ایمیل‌های ارسالیِ آلیس هم می‌شود. همچنین ترودی می‌تواند هر چیزی که کامپیوتر آلیس به چاپگرِ بلوتوثی که نزدیک او قرار دارد، ارسال می‌کند را بگیرد (مثل ایمیل‌های دریافتی و گزارشات محرمانه). خوشبختانه بلوتوث یک نظام امنیتیِ دقیق برای بی‌اثرکردنِ ترودی‌های این جهان دارد. در این‌جا مشخصه‌های اصلی این نظام امنیتی را به اجمال بیان خواهیم کرد.

بلوتوث نگارش 2.1 و نسخه‌های بعد از آن چهار مود امنیتی دارند. از کنترل صفر تا رمزگذاری کاملِ داده و کنترل جامعیت. همانند 802.11، اگر امنیت از کار انداخته شود (پیش‌فرض برای دستگاه‌های قدیمی‌تر) هیچ امنیتی وجود نخواهد داشت. اغلبِ کاربران تا زمانی که یک نفوذ جدی روی ندهد، امنیت را خاموش نگه می‌دارند؛ سپس امنیت را روشن می‌کنند. در کشاورزی این رویکرد را چنین می‌شناسند: بستنِ در اصطبل بعد از فرار اسب!

بلوتوث در چند لایه، امنیت را فراهم می‌کند. در لایه‌ی فیزیکی، پرش فرکانس (frequency hopping) یک امنیت بسیار اندک را فراهم می‌کند اما چون هر دستگاه بلوتوثی که به درون یک پیکونت (piconet) می‌رود ناچار است توالی پرش فرکانسی‌اش (frequency hopping sequence) را اعلام

کند، بدیهی است که این توالی سرّی نخواهد ماند. امنیت واقعی هنگامی آغاز می‌شود که فرمانبَری (slave) که به تازگی رسیده است، به فرمانده (master) درخواست یک کانال را می‌دهد. تا قبل از بلوتوثِ 2.1 ، فرض می‌شد که برای دو دستگاه، از قبل یک کلید سرّی مشترک قرار داده شده است. در بعضی موارد دو دستگاه توسط سازنده، به هم سیم‌بندی (hardwired) می‌شوند (مثلاً تلفن موبایل و هِدسِتی که روی هم و به عنوان یک دستگاه واحد فروخته می‌شوند). در مواردی دیگر، یک دستگاه (مثلاً هِدسِت) یک کلید سیم‌بندی شده (ثابت) دارد و کاربر بایستی آن کلید را به دستگاه دیگر (مثلاً به تلفن موبایل) به صورت یک عدد دهدهی وارد کند. این کلیدهای اشتراکی، **کلید عبور**[1] نامیده می‌شوند. متأسفانه غالباً کلیدهای عبور به صورت سخت‌افزاری کد "۱۲۳۴" یا یک مقدارِ قابل پیش‌بینیِ دیگر دارند. این کد به هرحال چهار رقم دهدهی است و لذا تنها 10^4 انتخاب وجود دارد. در بلوتوث 2.1 با لحاظ کردنِ امنیت به شیوه‌ی جفتی و ساده، دستگاه‌ها یک کد از یک محدوده‌ی شش ـ رقمی انتخاب می‌کنند. این کار باعث می‌شود که کلید عبور کمتر قابل پیش‌بینی باشد ولی هنوز از امن بودن فاصله دارد.

برای برقراریِ یک کانال، هر کدام از فرمانبَر و فرمانده کنترل می‌کنند که آیا طرفِ دیگر کلید عبور را می‌داند یا خیر. اگر طرفِ دیگر کلید عبور را بلد باشد، فرمانبَر و فرمانده مذاکره می‌کنند که آیا کانال رمزگذاری شود، کنترلِ جامعیت روی آن اِعمال شود، یا هر دو. سپس آن‌ها یک کلید نشستِ ۱۲۸ـ بیتیِ تصادفی انتخاب می‌کنند که تعدادی از بیت‌های آن ممکن است عمومی باشند. نکته‌ای که سبب تضعیف این کلید می‌شود، پیروی کردن از محدودیت‌های حکومتی در کشورهای مختلف است که جهت منع صادرات طراحی می‌شوند، یا آن‌که استفاده از کلیدهایی است که طول آن‌ها بیش از آن باشد که دولت بتواند آن‌ها را بشکند.

رمزگذاری از یک رمزِ جریانی به نام **E0** استفاده می‌کند؛ کنترل جامعیت نیز از **SAFER+** استفاده می‌کند. هر دوی این‌ها، رمزهای بلوکیِ کلید ـ متقارنِ متعارف هستند. رمز SAFER+ به مسابقات AES فرستاده شد اما در همان راندِ اول از دور مسابقات خارج گردید چون کُندتر از سایر الگوریتم‌های شرکت کننده بود. قبل از آن‌که رمز AES انتخاب شود، بلوتوث نهایی شده بود؛ اگر چنین نمی‌شد، بسیار محتمل بود که از ریندال استفاده کند.

رمزگذاری واقعی با استفاده از رمز جریانی در شکل ۸-۱۴ نشان داده شده است، به‌طوری‌که برای تولید متن رمزی، متن آشکار با جریان کلید (keystream) XOR شده است. متأسفانه خودِ E_0 (نیز همانند RC4) ممکن است ضعف‌های مهلکی داشته باشد (Jakobsson و Wetzel، ۲۰۰۱). هرچند تا زمان نگارش این متن، این رمز شکسته نشده ولی شباهت‌های آن با رمز A5/1، که ناکامی شدید آن باعث عدم اطمینان به ترافیک تلفن GSM شده، ایجاد نگرانی می‌کند (Biryukov و همکاران، ۲۰۰۰).

1. Passkey

گاهی مردم (که البته شامل نویسندگان این کتاب نیز می‌شود) تعجب می‌کنند که در بازی همیشگیِ موش و گربه که مابین رمزنگاران و رمزیاب‌ها در جریان است، اغلب این رمزیاب‌ها هستند که در طرف برنده قرار دارند.

مورد بعدی در بحث امنیت آن است که بلوتوث تنها دستگاه‌ها را تصدیق هویت می‌کند، نه کاربران را، بنابراین سرقت یک دستگاهِ بلوتوثی می‌تواند امکان دسترسی به حساب‌های مالی یا سایر حساب‌های کاربر را بدهد. اما ضمناً بلوتوث امنیت را در لایه‌های بالاتر پیاده‌سازی می‌کند، به همین دلیل حتی اگر امنیت در لایه‌ی پیوند شکسته شود، بخشی از امنیت باقی می‌مانَد، به‌خصوص برای کاربردهایی که برای ورود دستی، لازم است از طریق کیبورد یا وسیله‌ای شبیه آن، چیزی شبیه کد PIN داده شود تا تراکنش تکمیل گردد.

۸–۷ پروتکل‌های تصدیق هویت

تصدیق هویت عبارت‌است از روشی که از طریق آن یک پردازه درباره‌ی این مطلب که طرفش همانی است که ادعا می‌کند، و یک کلاهبردار نیست، بتواند راستی‌آزمایی کند. راستی‌آزماییِ شناسه‌ی یک پردازه‌ی راه دور در عینِ حضور یک نفوذگر فعال و بدنهاد، بسیار دشوار است و نیاز به پروتکل‌های پیچیده‌ی مبتنی بر رمزنگاری دارد. در این بخش بعضی از پروتکل‌های متعددِ تصدیق هویت که برای شبکه‌های کامپیوتریِ ناامن استفاده می‌شوند را مطالعه خواهیم کرد.

به این نکته توجه داشته باشید که بعضی از افراد "تصدیق صلاحیت" (Authorization) را با "تصدیق هویت" (Authentication) اشتباه می‌گیرند. تصدیق هویت با این پرسش سروکار دارد که آیا شما واقعاً با پردازه‌ی مشخصی ارتباط برقرار کرده‌اید یا خیر. تصدیق صلاحیت به این موضوع می‌پردازد که آن پردازه اجازه‌ی انجام چه کارهایی را دارد. برای مثال، فرض کنید یک پردازه‌ی مشتری با یک خدمتگزار فایل تماس می‌گیرد و می‌گوید: "من پردازه‌ی اسکات[1] هستم و مایلم فایل *cookbook.old* را حذف کنم." از دید خدمتگزار فایل در اینجا بایستی به دو پرسش پاسخ داده شود.

۱. آیا این واقعاً پردازه‌ی اسکات است (تصدیق هویت)؟
۲. آیا اسکات اجازه‌ی حذف *cookbook.old* را دارد (تصدیق صلاحیت)؟

فقط درصورتی‌که به هر دوی این پرسش‌ها بدون هیچ ابهامی پاسخ مثبت داده شود، عملِ مورد درخواست می‌تواند انجام گیرد. پرسش کلیدی در واقع پرسش اولی است. وقتی خدمتگزار بداند که با چه کسی طرف صحبت است، کنترل تصدیق صلاحیت در واقع فقط عبارت‌است‌از جستجو در جداول یا پایگاه‌های داده‌ی محلی. به همین دلیل است که در این بخش بر روی تصدیق هویت تمرکز خواهیم کرد.

1. Scott

مود کلی که اساساً تمام پروتکل‌های تصدیق هویت از آن استفاده می‌کنند، به این صورت است: آلیس با ارسال یک پیغام به باب یا به یک **KDC**[1] قابل اعتماد که انتظار می‌رود صادق باشد، کار را آغاز می‌کند. پیغام‌های متعددِ دیگری در جهت‌های مختلف، کار را ادامه می‌دهند. همین‌طور که این پیغام‌ها ارسال می‌شوند ترودی ممکن است آن‌ها را رهگیری کرده، آن‌ها را ویرایش و یا بازنواخت کند تا آلیس و باب را فریب بدهد و یا آن‌که فقط از کارهای آن‌ها سر در بیاورد.

معهذا هنگامی‌که پروتکل تکمیل شود، آلیس مطمئن است که با باب صحبت می‌کند و باب نیز اطمینان دارد که در حال صحبت با آلیس است. علاوه بر این در اغلب پروتکل‌ها دو طرف همچنین یک **کلیدِ نشست**[2] سرّی نیز برای استفاده در محاورات بعدی ایجاد می‌کنند. در عمل به دلیل کارایی، تمام ترافیک داده با استفاده از رمزنگاریِ کلید ـ متقارن رمزگذاری می‌شود (معمولاً با AES یا DES سه‌گانه)، هر چند که از رمزنگاریِ کلید ـ عمومی برای خودِ پروتکل‌های تصدیق هویت و برای ایجادِ کلید نشست، استفاده‌ی گسترده‌ای می‌شود.

در این‌که به ازای هر اتصال جدید، از یک کلید نشستِ جدید (که به طور تصادفی انتخاب می‌شود) استفاده می‌شود، نکته‌ای وجود دارد: به حداقل رساندنِ حجم ترافیک ارسالیِ کاربرانِ کلیدهای سرّی یا کلیدهای عمومی، به منظور کاهش حجم متن رمزی‌ای که یک نفوذگر می‌تواند به دست بیاورد. و همچنین به حداقل رساندنِ خسارات در صورتی‌که پردازه دچار خرابی شده و اطلاعاتِ روگرفتِ حافظه[3] به دست افراد نااهل بیفتد. این امید می‌رود که در آن زمان تنها کلیدِ موجود، کلید نشست باشد. بعد از برقراریِ نشست، تمام کلیدهای دائمی بایستی به دقت صفر شوند.

۸-۷-۱ تصدیق هویتِ مبتنی‌بر یک کلید سرّی اشتراکی

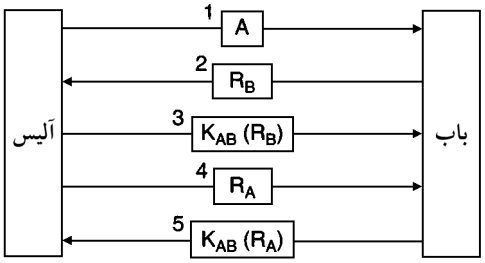
برای اولین پروتکل تصدیق هویتمان، فرض می‌کنیم آلیس و باب هم‌اکنون یک کلید سرّی مثل K_{AB} را به اشتراک گذاشته‌اند. این کلید اشتراکی می‌تواند از طریق تلفن یا حضوراً مورد توافق قرار گرفته باشد، ولی به هرحال از طریق شبکه نبوده است (یعنی توافق درباره‌ی کلید اشتراکی به روش ناامنی انجام نشده است).

این پروتکل مبتنی بر یک اصلِ موجود در بسیاری از پروتکل‌های تصدیق هویت می‌باشد: یک طرف، یک عدد تصادفی به طرف دیگر می‌فرستد. سپس طرف دیگر آن را به روش خاصی تغییر می‌دهد و حاصل را برمی‌گردانَد. این‌گونه پروتکل‌ها، پروتکل‌های **چالش ـ پاسخ**[4] نامیده می‌شوند. در این پروتکل و پروتکل‌های بعدیِ تصدیق هویت، این ملاحظات در نظر گرفته شده‌اند:

حروف A و B شناسه‌هایِ آلیس و باب هستند.

---

1. Key Distribution Center مرکز توزیع کلید 2. Session key 3. Core dump
4. Challenge-response



شکل ۸-۳۲ تصدیق هویتِ دو-طرفه با استفاده از پروتکل چالش ـ پاسخ.

R_i ها چالش‌ها هستند، به طوری که i مشخص‌کننده‌ی چالش‌کننده (challenger) است.

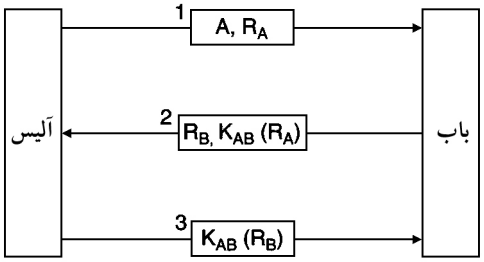
K_i ها کلیدها هستند، به طوری که i مالک کلید را مشخص می‌کند.

K_s کلید نشست است.

در شکل ۸-۳۲، توالی پیغام برای اولین پروتکل تصدیق هویت ما نشان داده شده است. در پیغام 1، آلیس شناسه‌ی خود یعنی A را به باب ارسال می‌کند، آن هم به روشی که باب بفهمد. البته باب هیچ راهی ندارد تا بفهمد آیا این پیغام از طرف آلیس است یا از طرف ترودی، لذا یک چالش انتخاب می‌کند، یک عدد تصادفیِ بزرگ به نام R_B و این عدد را به صورت متن آشکار و به عنوان پیغام 2 به "آلیس" برمی‌گردانَد. سپس آلیس این پیغام را با کلیدی که با باب مشترک است، رمزگذاری کرده و متن رمزیِ $K_{AB}(R_B)$ را در پیغام 3 پس می‌فرستد. هنگامی‌که باب این پیغام را می‌بیند، بلافاصله می‌فهمد این پیغام از طرف آلیس آمده زیرا ترودی K_{AB} را نمی‌داند و لذا نمی‌تواند این پیغام را تولید کند. به‌علاوه، از آن‌جا که R_B از یک فضای بزرگ (مثلاً از میان اعداد تصادفیِ ۱۲۸ـ بیتی) به صورت تصادفی انتخاب شده است، خیلی نامحتمل است که ترودی R_B و پاسخِ آن را از نشستِ اخیر دیده باشد. همچنین به همین اندازه هم نامحتمل است که ترودی بتواند پاسخ صحیح به هر چالشی را حدس بزند.

در این مرحله باب اطمینان پیدا می‌کند که در حال گفتگو با آلیس است، اما آلیس از هیچ چیزی مطمئن نیست. آلیس می‌داند که ترودی ممکن است پیغام 1 را رهگیری کرده و R_B را در پاسخ پس فرستاده باشد. ممکن است باب شب قبل درگذشته باشد. آلیس برای آن‌که پیدا کند طرف صحبتش کیست، یک عدد تصادفیِ R_A برداشته و آن را به صورت متن آشکار، در پیغام 4 به باب ارسال می‌کند. با پاسخِ باب همراه با $K_{AB}(R_A)$ ، آلیس حالا می‌داند در حال صحبت با باب است. اکنون اگر آن‌ها مایل به برقراریِ یک کلید نشست باشند، آلیس می‌تواند یک کلید نشستِ K_S انتخاب کرده و شکلِ رمزگذاری شده‌ی آن، با K_{AB} را به باب ارسال کند.

پروتکل شکل ۸-۳۲ حاوی ۵ پیغام است. بیایید ببینیم آیا می‌توانیم باهوش بوده و بعضی از آن‌ها را حذف کنیم؟ یک رویکرد در شکل ۸-۳۳ نشان داده شده. در این شکل آلیس به جای آن‌که منتظر باب بماند، خودش پروتکل چالش ـ پاسخ را راه‌اندازی می‌کند. مشابه قبل، باب در پاسخ به چالشِ آلیس، چالش خودش را ارسال می‌کند. کل پروتکل می‌تواند به ۳ پیغام (به جای ۵ پیغام) کاهش یابد.

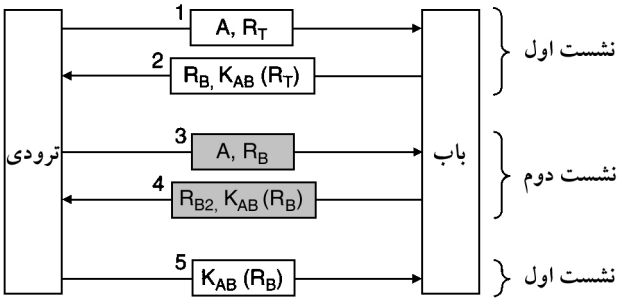


شکل ۸-۳۳ یک پروتکل تصدیق هویت دو- طرفه‌ی کوتاه‌تر.

آیا این پروتکل نسبت به نگارش اصلی، یک پیشرفت محسوب می‌شود؟ از یک نظر بله: کوتاه‌تر است. متأسفانه این پروتکل نیز اشتباه است. تحت شرایط خاص، ترودی می‌تواند با استفاده از **حمله‌ی بازتاب**[1] این پروتکل را شکست دهد. در عمل اگر ترودی بتواند به طور همزمان چندین نشست را با باب باز کند، می‌تواند این پروتکل را بشکند. مثلاً اگر باب یک بانک باشد و برای پذیرش اتصالاتِ متعددِ همزمان از طرف ماشین‌های متصدیان بانک مهیا شده باشد، شرایطِ مورد نظر ترودی اتفاق خواهد افتاد.

حمله‌ی بازتابِ ترودی در شکل ۸-۳۴ نشان داده شده. این حمله با ادعای ترودی مبنی بر این‌که آلیس است و با ارسال R_T ، آغاز می‌شود. باب طبق روال عادی با چالش خود، یعنی R_B ، پاسخ می‌دهد. حالا ترودی گیر می‌افتد.ترودی چکار می‌تواند بکند؟ او $K_{AB}(R_B)$ را نمی‌داند.

ترودی می‌تواند با پیغام 3، نشست دوم را باز کند و R_B را که به عنوان چالشِ خود از پیغام 2 برداشته است، تأمین کند. باب با آسودگیِ خاطر این پیغام را رمزگذاری کرده و $K_{AB}(R_B)$ را در پیغام 4 پس می‌فرستد. این پیغام‌ها را در نشست دوم با رنگ خاکستری نشان داده‌ایم تا مشخص‌تر باشند. حالا ترودی اطلاعاتِ مورد نیازش را دارد، لذا می‌تواند نشست اول را تکمیل کند و نشست دوم را بی‌نتیجه رها سازد. باب اینک متقاعد شده که ترودی، آلیس است بنابراین وقتی ترودی موجودی حساب بانکیِ آلیس را می‌خواهد، باب بدون سوال، پاسخ می‌دهد. بعد، وقتی ترودی از او می‌خواهد تمام این موجودی را به یک حساب بانکیِ سرّی در سوئیس منتقل کند، باب بدون لحظه‌ای تردید این کار را انجام می‌دهد.



شکل ۸-۳۴ حمله‌ی بازتاب.

1. Reflection attack

نتیجه‌ی اخلاقیِ این داستان آن است که:

طراحی یک پروتکل تصدیق هویتِ صحیح دشوارتر از آن چیزی است که به نظر می‌رسد.

چهار قانون کلی زیر به طراح کمک می‌کند تا از دام‌های رایج دوری نماید:

۱. آغازکننده مجبور باشد ثابت کند که کیست، قبل از آن‌که پاسخ‌دهنده مجبور به این کار باشد. این مورد باعث می‌شود باب قبل از آن‌که ترودی مجبور باشد مدرکی ارائه دهد که نشان دهد او کیست، از دادنِ اطلاعات ارزشمند به ترودی اجتناب کند.

۲. آغازکننده و پاسخ‌دهنده مجبور باشند از کلیدهای مختلفی برای اثبات خویش استفاده کنند، حتی اگر این موضوع به معنای داشتنِ دو کلید اشتراکیِ K_{AB} و K'_{AB} باشد.

۳. آغازکننده و پاسخ‌دهنده مجبور باشند چالش‌شان را از مجموعه‌های مختلفی بردارند. مثلاً لازم باشد آغازکننده از اعداد زوج و پاسخ‌دهنده از اعداد فرد استفاده کنند.

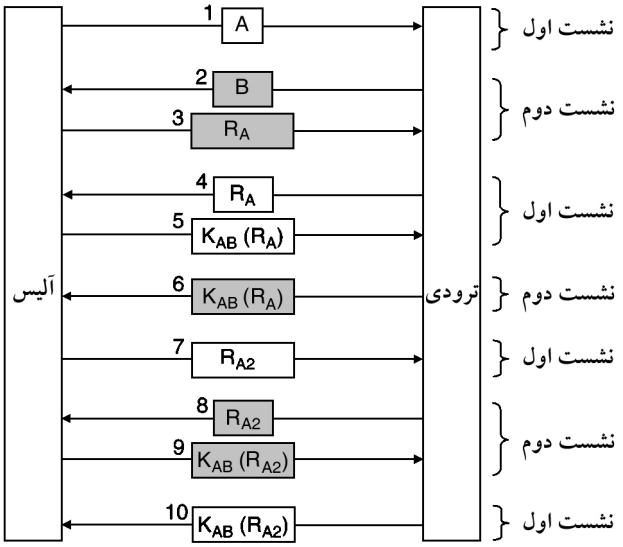
۴. مقاوم بودنِ پروتکل در برابر حمله‌ها، شامل نشست‌های موازی هم باشد، به طوری که اطلاعاتِ به دست آمده از یک نشست، در نشست دیگری که متفاوت از آن است، استفاده نشود.

اگر حتی یکی از این قوانین نقض شوند، پروتکل می‌تواند به دفعات شکسته شود. حالا اگر هر چهار قانون نقض شده باشند، نتایج مصیبت‌بار خواهند بود.

اکنون بیایید از نزدیک‌تر به شکل ۸-۳۲ نگاه کنیم. آیا مطمئن هستید این پروتکل موردِ حمله‌ی بازتاب قرار ندارد؟ احتمالش وجود دارد. این موضوعی بسیار ظریف است. ترودی می‌توانست با استفاده از حمله‌ی بازتاب، پروتکل ما را شکست دهد زیرا امکانِ باز کردنِ یک نشست دوم با باب و حقه‌زدن به او برای پاسخ‌دادن به پرسش‌های خودش، وجود داشت. اگر آلیس یک کامپیوتر همه‌منظوره بود که به جای یک کاربر در پشت یک کامپیوتر، چندین نشست را می‌پذیرفت، چه اتفاقی می‌افتاد؟ بیایید ببینیم ترودی چکار می‌توانست بکند.

برای آن‌که ببینیم حمله‌ی ترودی چگونه عمل می‌کند، شکل ۸-۳۵ را مشاهده کنید. آلیس با اعلام شناسه‌اش در پیغام 1 کار را آغاز می‌کند. ترودی این پیغام را رهگیری کرده و با این ادعا که باب است، نشست خودش را با پیغام 2 شروع می‌کند. اینجا هم مجدداً پیغام‌های شماره‌ی 2 را به رنگ خاکستری نشان داده‌ایم. آلیس به پیغام 2 پاسخ می‌دهد و در پیغام 3 می‌گوید: "آیا ادعا می‌کنی باب هستی؟ ثابت کن. " این‌جاست که ترودی گیر می‌افتد زیرا نمی‌تواند ثابت کند که باب است.

حالا ترودی چه می‌کند؟ او به نشست اول برمی‌گردد، جایی که نوبت اوست که یک چالش ارسال کند، و R_A ای که از پیغام 3 برداشته را ارسال می‌کند. آلیس در پیغام 5 به او پاسخ می‌دهد و ترودی را با اطلاعاتی که برای ارسال در پیغام 6 در نشست 2 به آن نیاز دارد، تأمین می‌کند. در این مرحله ترودی در واقع برنده است زیرا به چالشِ آلیس در نشست 2 با موفقیت پاسخ داده است. او حالا می‌تواند نشست 1 را لغو کند، و هر عددِ قدیمی‌ای را برای بقیه‌ی نشست 2 ارسال کند تا یک نشستِ تصدیق هویت شده با آلیس در نشست 2 داشته باشد.

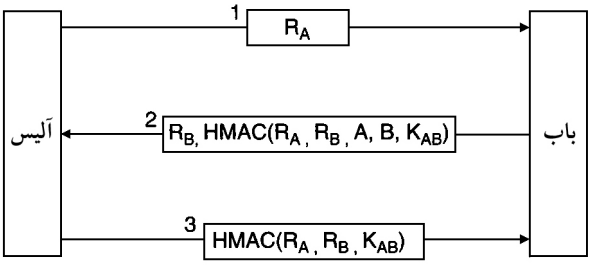


شکل ۸-۳۵ یک حمله‌ی بازتاب بر روی پروتکلِ شکل ۸-۳۲.

اما ترودی واقعاً نابکار است و به کم قانع نیست. بنابراین به جای ارسال هر گونه عدد قدیمی‌ای برای تکمیل نشست 2، منتظر می‌مانَد تا آلیس پیغام 7 را ارسال کند، یعنی چالشِ آلیس برای نشست 1. البته ترودی نمی‌داند چطور پاسخ دهد لذا دوباره از حمله‌ی بازتاب استفاده می‌کند و R_{A2} را به عنوان پیغام 8 ارسال می‌کند. آلیس به راحتی R_{A2} را در پیغام 9 رمزگذاری می‌کند. حالا ترودی به نشست 1 برمی‌گردد و عددی که آلیس می‌خواهد را در پیغام 10 برایش ارسال می‌کند، همان عددی که به راحتی از پیغام 9 که آلیس ارسال کرده، کپی کرده است. اکنون ترودی دو نشستِ کاملاً تصدیق هویت شده با آلیس دارد.

نتیجه‌ی این حمله کمی با نتیجه‌ی حمله به پروتکلِ سه ـ پیغامه که در شکل ۸-۳۴ دیدیم، متفاوت است. در این‌جا ترودی دو اتصال تصدیق هویت شده با آلیس دارد. در مثال قبل، ترودی یک اتصالِ تصدیق هویت شده با باب داشت. در این‌جا هم اگر تمام قوانین پروتکل تصدیق هویت که قبلاً بحَثشان را کرده‌ایم را اِعمال کرده بودیم، این حمله می‌توانست متوقف گردد. برای بررسی دقیق‌تر این نوع حمله‌ها و نحوه‌ی خنثی کردنِ آن‌ها به Bird و همکاران (۱۹۹۳) مراجعه نمایید. آن‌ها همچنین در بررسی‌هایشان نشان می‌دهند که اَمکانِ ساخت پروتکل‌های متقارنی که صحتِ آن‌ها قابل اثبات باشد، وجود دارد. معهذا ساده‌ترین مدل از این‌گونه پروتکل‌ها نیز کمی پیچیده است، لذا هم‌اینک کلاس متفاوتی از پروتکل‌ها را که نشان خواهیم داد که این کلاس نیز صحیح عمل می‌کند.

پروتکل تصدیق هویتِ جدید در شکل ۸-۳۶ نشان داده شده است (Bird و همکاران، ۱۹۹۳). این پروتکل از یک HMAC استفاده می‌کند (از نوعی که هنگام مطالعه‌ی IPsec مشاهده کردیم). آلیس کار را با ارسال یک نانس به باب، یعنی R_4 ، به عنوان پیغام 1 آغاز می‌کند. باب با انتخاب نانسِ خودش،



شکل ۸–۳۶ تصدیق هویت با استفاده از HMAC ها.

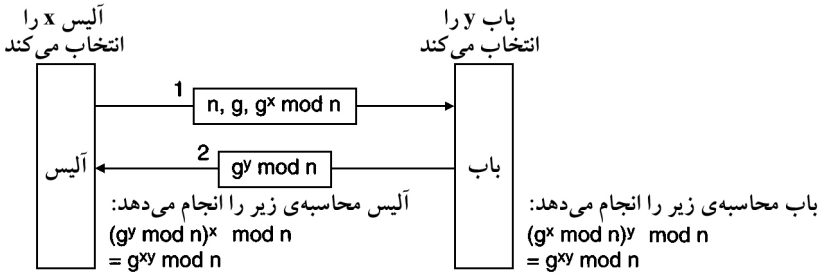
R_B ، پاسخ می‌دهد و آن را با یک HMAC برمی‌گرداند. کد HMAC از یک ساختار داده، مشتمل بر نانسِ آلیس، نانسِ باب، شناسه‌های آن‌ها، و کلید اشتراکیِ سرّی، یعنی R_{AB} ، تشکیل می‌شود. سپس این ساختار داده (مثلاً با استفاده از SHA-1) درون HMAC درهم‌سازی می‌شود. هنگامی‌که آلیس پیغام 2 را دریافت می‌کند، حالا یک R_A دارد (که خودش انتخاب کرده)، یک R_B دارد (که به صورت یک متن آشکار برایش رسیده)، دو شناسه دارد، و یک کلید سرّی R_{AB} دارد (که آلیس در تمام مدت آن را می‌دانسته)، بنابراین می‌تواند خودش HMAC را محاسبه کند. اگر این HMAC با آن HMAC ای که در پیغام است، یکی باشد آلیس می‌فهمد که در حال صحبت با باب است زیرا ترودی R_{AB} را نمی‌داند و لذا نمی‌تواند بفهمد که چه HMAC ای ارسال می‌شود. آلیس پاسخش را به باب ارسال می‌کند، با یک HMAC مشتمل بر فقط دو نانس.

آیا ترودی می‌تواند به طریقی این پروتکل را تخریب کند؟ خیر، زیرا نمی‌تواند مانند آنچه که در شکل ۸–۳۴ و ۸–۳۵ اتفاق افتاده بود، هیچ یک از دو طرف را وادار نماید تا مقداری که او انتخاب کرده را رمزگذاری یا درهم‌سازی کنند. هر دو HMAC ها شامل مقادیری هستند که توسط طرفِ ارسال کننده انتخاب شده‌اند، یعنی چیزی که تحت کنترل ترودی نیست.

استفاده از HMACها تنها روشِ استفاده از این ایده نیست. نظام دیگری که غالباً به جای محاسبه‌ی HMAC روی یک سری اقلام (item)، از آن استفاده می‌شود عبارت‌است‌از رمزگذاری اقلامِ مورد نظر به صورت ترتیبی و با استفاده از زنجیر کردنِ بلوکِ رمزی.

۸–۷–۲ برقراری یک کلید اشتراکی : تبادل کلید دیفی ـ هِلمن

تا این‌جا فرض کرده‌ایم که آلیس و باب یک کلید سرّی را به اشتراک دارند. فرض کنید این طور نباشد (زیرا تاکنون هیچ PKI پذیرفته شده‌ای برای امضا و توزیع گواهینامه‌ها، در سطح جهانی وجود ندارد). آلیس و باب چگونه می‌توانند یک کلید سرّی مشترک برقرار کنند؟ یک راه برای آلیس آن است که با باب تماس بگیرد و کلیدش را با تلفن در اختیار او قرار دهد. ولی ممکن است باب بگوید: "من از کجا بدانم شما آلیس هستید و ترودی نیستید؟" آن‌ها می‌توانند تلاش کنند تا جلسه‌ای را ترتیب دهند و هر کدام یک پاسپورت، یک گواهینامه‌ی رانندگی، و سه کارت اعتباریِ مهم با خود بیاورند. اما مردم



شکل ۸-۳۷ تبادل کلیدِ دیفی ـ هِلمن.

سرشان شلوغ است. آن‌ها ممکن است تا مدت‌ها نتوانند زمانی که برای هر دو طرف مناسب باشد را پیدا کنند. خوشبختانه، هر چند ممکن است باورنکردنی باشد ولی راهی وجود دارد که تمام افرادی که با هم بیگانه‌اند بتوانند در روزِ روشن یک کلید سرّیِ اشتراکی ایجاد کنند، حتی در شرایطی که ترودی به دقت هر پیغامی را ضبط می‌کند.

پروتکلی که به افراد بیگانه با هم اجازه‌ی برقراری یک کلید سرّیِ اشتراکی را می‌دهد، پروتکل **تبادل کلیدِ دیفی ـ هِلمن**[1] نامیده شده و به صورت زیر کار می‌کند (Diffie و Hellman ، ۱۹۷۶). آلیس و باب باید بر روی دو عدد بزرگ n و g توافق کنند. عدد n یک عدد اول است، $(n-1)/2$ نیز یک عدد اول است، و شرایط خاصی به g اِعمال می‌شوند. این اعداد می‌توانند عمومی باشند، لذا هر کدام از دو طرف فقط می‌توانند n و g را انتخاب کنند و به طور صریح به دیگری بگویند. اینک آلیس یک عدد بزرگ (مثلاً یک عدد ۱۰۲۴ـ بیتی) مثل x انتخاب می‌کند و آن را مخفی نگه می‌دارد. به همین ترتیب، باب نیز یک عدد بزرگِ سرّی مثل y انتخاب می‌کند.

همان‌طور که در شکل ۸-۳۷ نشان داده شده، آلیس با ارسال یک پیغام به باب شامل $(n, g, g^x \bmod n)$، پروتکلِ تبادل کلید را راه‌اندازی می‌کند. باب با ارسال یک پیغام که شامل $g^y \bmod n$ است، به آلیس پاسخ می‌دهد. حالا آلیس عددی که باب به او فرستاده را به توان x امِ پیمانه‌ی n می‌رساند تا به مقدار $(g^y \bmod n)^x \bmod n$ دست یابَد. باب عملیات مشابهی انجام می‌دهد تا به $g^{xy} \bmod n$ برسد. با استفاده از قوانین پیمانه‌ی ریاضی، هر دو محاسبات به $(g^x \bmod n)^y \bmod n$ منتهی می‌شوند. می‌بینید؟ به یک باره آلیس و باب صاحب یک کلید سرّی اشتراکی $g^{xy} \bmod n$ شدند.

البته ترودی هر دو پیغام را دیده است. او g و n را از پیغام 1 می‌داند. اگر ترودی می‌توانست x و y را حساب کند، کلید سرّی را می‌توانست بیابَد. مشکل این‌جاست که فقط با داشتنِ $g^x \bmod n$ نمی‌تواند x را پیدا کند. هیچ الگوریتمِ عملیاتی‌ای برای محاسبه‌ی لگاریتم‌های گسسته به پیمانه‌ی اعداد اولِ خیلی بزرگ، شناخته نشده است.

برای ملموس‌تر شدنِ این مثال، از مقادیر (کاملاً غیرواقعی) $n = 47$ و $g = 3$ استفاده خواهیم کرد. آلیس و باب به ترتیبِ مقادیر $x = 8$ و $y = 10$ را انتخاب می‌کنند. هر دوی این اعداد مخفی

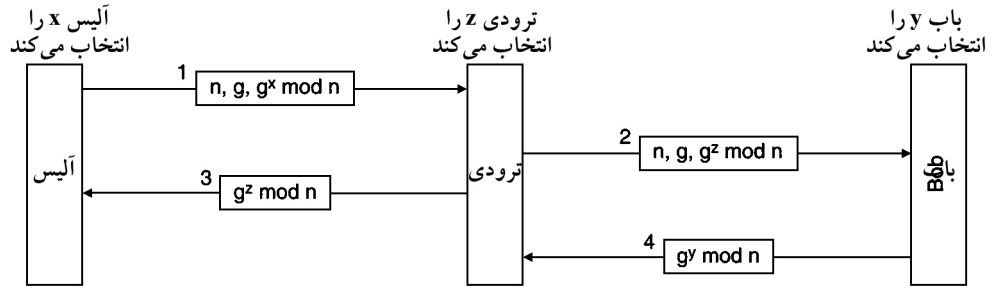
1. Diffie-Hellman key exchange

نگهداشته می‌شوند. پیغام آلیس به باب (47, 3, 28) است زیرا $3^8 \bmod 47$ برابر با 28 است. پیغام باب به آلیس (17) است. آلیس $17^8 \bmod 47$ را محاسبه می‌کند که حاصلش 4 است. باب $28^{10} \bmod 47$ را محاسبه می‌کند که حاصلش 4 است. اکنون آلیس و باب مستقلاً مشخص کردند که کلید سرّی 4 است. حالا ترودی برای پیدا کردنِ کلید مجبور است معادله‌ی $3^x \bmod 47 = 28$ را حل کند. حلِ این معادله با جستجوی جامع برای اعدادِ کوچکی نظیر این مثال می‌تواند انجام شود، ولی برای اعدادی که طولشان صدها بیت است، خیر. تمام الگوریتم‌های شناخته شده‌ی فعلی، حتی بر روی سوپرکامپیوترهایی با سرعتِ برق‌آسا و با موازی‌سازیِ انبوه، به زمانی بیش از حد طولانی نیاز دارند.

علی‌رغم ظرافت و زیباییِ الگوریتم دیفی ـ هِلمن، مشکلی وجود دارد: هنگامی‌که باب سه‌تایی (47, 3, 28) را دریافت می‌کند از کجا بداند از طرف آلیس است، نه ترودی. راهی وجود ندارد. متأسفانه همان‌طور که در شکل ۸–۳۸ نشان داده شده، ترودی می‌تواند از این حقیقت سوء استفاده کرده و هم آلیس و هم باب را گول بزنَد. در این‌جا، همزمان با آلیس و باب که به ترتیب x و y را انتخاب می‌کنند، ترودی نیز عدد تصادفیِ خودش، یعنی z را انتخاب می‌کند. آلیس پیغام 1 را به مقصد باب ارسال می‌کند. ترودی این پیغام را رهگیری می‌کند و با استفاده از g و n (که به هرحال عمومی هستند) پیغام 2 را به باب می‌فرستد، اما این پیغام را به جای x با z ارسال می‌کند. ترودی همچنین پیغام 3 را به آلیس پس می‌فرستد. بعداً باب پیغام 4 را به آلیس ارسال می‌کند، که دوباره ترودی آن پیغام را رهگیری کرده و از آنِ خود می‌کند.

اکنون همگی همنهشتیِ ریاضی را انجام می‌دهند. آلیس کلید سرّی را به صورت $g^{xy} \bmod n$ محاسبه می‌کند، ترودی نیز (برای پیغام‌های ارسالی به آلیس) همین عمل را انجام می‌دهد. باب $g^{xy} \bmod n$ را محاسبه می‌کند و ترودی نیز (برای پیغام‌های ارسالی به باب) همین عمل را انجام می‌دهد. آلیس تصور می‌کند در حال گفتگو با باب است، به همین دلیل یک کلیدِ نشست (با ترودی) برقرار می‌کند. باب نیز همین کار را انجام می‌دهد. هر پیغامی که آلیس بر روی این نشستِ رمزگذاری شده ارسال می‌کند، توسط ترودی گرفته شده، ذخیره شده، اگر ترودی بخواهد ویرایش می‌شود، و سپس به باب ارسال می‌شود (که این کار اختیاری است). در طرف دیگر نیز به همین ترتیب، ترودی همه چیز را می‌بیند و می‌تواند تمام پیغام‌ها را مطابق میلش ویرایش کند، آن هم زمانی که آلیس و باب در این خیال باطل به سر می‌برند که یک کانالِ امن به همدیگر دارند. به این دلیل، این حمله با عنوان **حمله‌ی فردی ـ در ـ میانه**[1] شناخته می‌شود. به این حمله، **حمله‌ی bucket brigade**[2] (صفی از انسان‌ها که سطل‌های آب را با یکدیگر دست به دست، رد می‌کنند ـ مترجم.) نیز گفته می‌شود زیرا یادآورِ آتش‌نشان‌های داوطلبی است که در قدیم سطل‌های آب را از مخزن آب تا محل آتش‌سوزی، دست به دست رد می‌کرده‌اند.

1. Man-in-the-middle attack 2. Bucket brigade attack



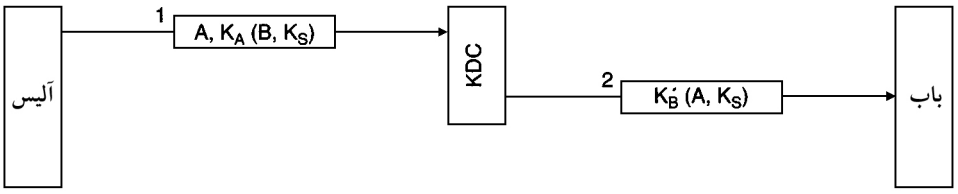
شکل ۸-۳۸ حمله‌ی فردی ـ در ـ میانه.

۸-۷-۳ تصدیق هویت با استفاده از یک مرکز توزیع کلید

برپاسازیِ یک کلید سرّیِ اشتراکی با یک فرد بیگانه، عمل می‌کند ولی نه کاملاً. از سوی دیگر، این کار از همان ابتدا هم ارزش انجام دادن نداشت. برای آن‌که با n نفر به این روش مرتبط شوید، به n کلید نیاز خواهید داشت. مدیریت کلید، برای فردی پُرمراجعه، کار دشواری است علی‌الخصوص اگر لازم باشد هر کلید بر روی یک کارت تراشه‌ی پلاستیکیِ جداگانه ذخیره شود.

یک رویکرد متفاوت عبارت‌استاز معرفی یک مرکز قابل اعتمادِ توزیع کلید. در این مدل، هر کاربر یک کلید اشتراکیِ منفرد با KDC دارد. تصدیق هویت و مدیریت کلیدِ نشست از طریق KDC انجام خواهد شد. همان‌طور که در شکل ۸-۳۹ نشان داده شده، ساده‌ترین پروتکل شناخته شده‌ی تصدیق هویتِ KDC، مستلزم وجود دو طرف و یک KDC قابل اعتماد می‌باشد.

ایده‌ای که در ورای این پروتکل قرار دارد ساده است: آلیس یک کلید نشست، مانند K_S انتخاب می‌کند و به KDC می‌گوید که با استفاده از K_S می‌خواهد با باب گفتگو کند. این پیغام با کلید سرّی‌ای که (فقط) میان آلیس با KDC به اشتراک گذاشته شده، یعنی K_A ، رمزگذاری می‌شود. این پیغام را KDC رمزبَرداری می‌کند، شناسه‌ی باب و کلید نشست را از آن استخراج می‌نماید. سپس KDC یک پیغام جدید می‌سازد که شامل شناسه‌ی آلیس و کلید نشست است و این پیغام را به باب ارسال می‌کند. این رمزگذاری با K_B انجام می‌شود، یعنی کلید سرّی‌ای که باب با KDC به اشتراک گذاشته است. هنگامی‌که باب این پیغام را رمزبَرداری می‌کند، می‌فهمد که آلیس تمایل به گفتگو با او دارد و متوجه می‌شود که آلیس از چه کلیدی می‌خواهد استفاده کند.



شکل ۸-۳۹ یک تلاش اولیه برای یک پروتکل تصدیق هویت که از KDC استفاده می‌کند.

عمل تصدیق هویت در این‌جا بدون هزینه انجام می‌شود. مرکز KDC می‌داند که پیغام 1 بایستی از طرف آلیس آمده باشد، زیرا هیچ کس دیگری قادر به رمزگذاری این پیغام با کلید سرّی آلیس نبوده است. به همین ترتیب، باب هم می‌داند که پیغام 2 بایستی از طرف KDC آمده باشد (که باب به آن اعتماد دارد) زیرا هیچ کس دیگری کلید سرّی باب را ندارد.

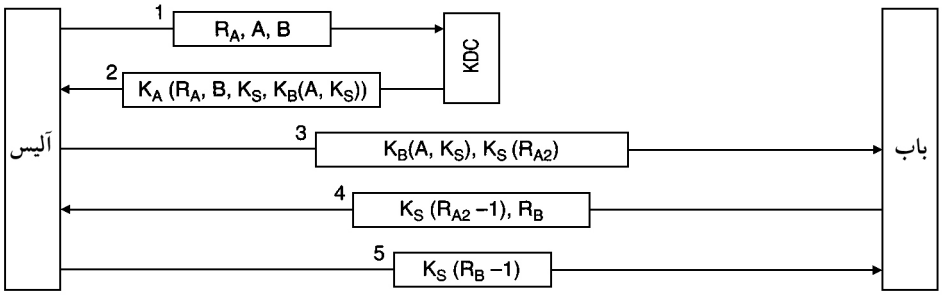
متأسفانه این پروتکل یک نقص جدی دارد. ترودی نیاز به پول دارد، بنابراین یک سرویس قانونی مورد نیاز آلیس را پیدا می‌کند و پیشنهاد جذابی به او می‌دهد، و این کار را به دست می‌آورَد. بعد از انجام این کار، ترودی خیلی مؤدبانه از آلیس تقاضای پرداخت هزینه از طریق انتقال بانکی می‌کند. سپس آلیس یک کلیدِ نشست با بانکدارش (باب) برقرار می‌کند. حالا آلیس یک پیغام درخواست انتقالِ وجه به شماره حساب ترودی، به باب ارسال می‌کند.

در همینِ حین، ترودی به شیوه‌های قبلی‌اش برمی‌گردد، یعنی جاسوسی در شبکه. او هم پیغام 2 در شکل ۸-۳۹، و هم پیغام مربوط به درخواست انتقالِ وجه که به دنبال پیغام 2 قرار دارد را کپی می‌کند. بعد، ترودی هر دو پیغام را به باب بازنواخت می‌کند. باب خیال می‌کند: "آلیس حتماً دوباره از ترودی سرویسی گرفته است. معلوم است ترودی کارش را خوب انجام می‌دهد." سپس باب باز هم همان مقدار پول را از حساب آلیس به حساب ترودی انتقال می‌دهد. بعد از پنجاهمین باری که جفت ـ پیغام‌ها ارسال شدند، باب به سرعت از دفترش خارج می‌شود تا ترودی را پیدا کرده و به او پیشنهاد پرداخت یک وام کلان را جهت توسعه‌ی کسب و کارش، که ظاهراً خیلی موفق است، بدهد. این مورد **حمله‌ی بازنواخت**[1] نام دارد.

چندین راه‌حل برای حمله‌ی بازنواخت وجود دارند. اولین راه، قرار دادنِ یک برچسب زمانی در هر پیغام است. به این ترتیب درصورت دریافت یک پیغام کهنه، می‌توان پیغام را از بین بُرد. مشکلِ این رویکرد آن است که کلاک‌های روی شبکه هرگز دقیقاً با هم همگام نیستند، لذا بایستی یک فاصله‌ی زمانی وجود داشته باشد که در خلال آن فاصله، یک برچسب زمانی دارای اعتبار باشد. ترودی می‌تواند در این فاصله‌ی زمانی، حمله‌ی بازنواخت را انجام داده و از مهلکه دور شود.

راه‌حل دوم عبارت است از قرار دادنِ یک نانس در هر پیغام. به این ترتیب هر یک از دو طرف باید تمام نانس‌های قبلی را به خاطر بسپرد و اگر پیغامی شامل یک نانسِ قبلاً استفاده شده است، آن را نپذیرد. اما نانس‌ها بایستی برای ابد نگه داشته شوند، مبادا ترودی بخواهد یک پیغام که مربوط به ۵ سال قبل است را بازنواخت کند. ضمناً، اگر ماشینی خراب شود و فهرست نانس‌هایش را از دست بدهد، باز هم در برابر حمله‌ی بازنواخت آسیب‌پذیر خواهد شد. برچسب‌های زمانی و نانس‌ها می‌توانند با هم ترکیب شوند تا مدت زمانی که نانس‌ها بایستی نگه داشته شوند، محدود گردد. اما واضح است که چنین پروتکلی بسیار پیچیده خواهد بود.

1. Replay attack



شکل ۸-۴۰ پروتکل تصدیق هویتِ نیدهام ـ شرودر.

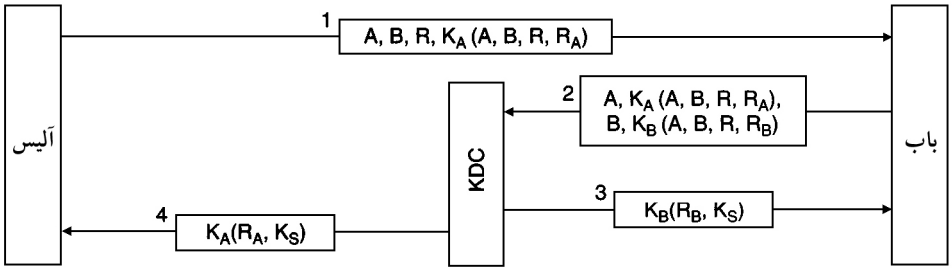
یک رویکرد پیشرفته‌تر برای تصدیق هویتِ متقابل[1] عبارت است از استفاده از یک پروتکل چالش ـ پاسخِ چند ـ مسیره[2]. یک مثال مشهور از چنین پروتکلی، پروتکل **تصدیق هویت نیدهام ـ شرودر**[3] است (Needham و Schroeder، ۱۹۷۸) که یکی از انواع آن در شکل ۸-۴۰ نشان داده شده است.

در آغازِ این پروتکل، آلیس به KDC می‌گوید که قصد دارد با باب گفتگو کند. این پیغام دربردارنده‌ی یک عدد تصادفی بزرگ، یعنی R_A ، به عنوان یک نانس است. مرکز KDC یک پیغام 2 پس می‌فرستد که دربردارنده‌ی عدد تصادفیِ آلیس، یک کلیدِ نشست، و یک بلیت (ticket) است که آلیس می‌تواند به باب ارسال کند. نکته‌ی مربوط به عدد تصادفیِ R_A این است که به آلیس اطمینان می‌دهد پیغام 2 تازه است و یک بازنواخت نیست. شناسه‌ی باب در برابر این وضعیت محصورسازی شده است: ممکن است ترودی چنین قصدِ خنده‌آوری داشته باشد که بخواهد B در پیغام 1 را با شناسه‌ی خودش جایگزین کند (تا با این جایگزین کردن، KDC در انتهای پیغام 2، بلیت را به جای K_B با K_T کدگذاری کند). بلیتِ کدگذاری شده با K_B ، درون پیغامِ کدگذاری شده قرار داده می‌شود تا جلوی ترودی گرفته شود و او نتواند بلیت را در راهِ برگشت به آلیس، با چیز دیگری جایگزین کند.

اکنون آلیس بلیت را همراه با یک عدد تصادفی جدید یعنی R_{A2} به باب ارسال می‌کند که با کلید نشست، یعنی K_S ، کدگذاری شده است. در پیغام 4، باب $K_S(R_{A2}-1)$ را پس می‌فرستد تا به آلیس اطمینان دهد در حال صحبت با بابِ حقیقی می‌باشد. برگرداندنِ $K_S(R_{A2})$ درست عمل نخواهد کرد زیرا ترودی می‌توانسته آن را از پیغام 3 کِش برود.

آلیس بعد از دریافت کردنِ پیغام 4، متقاعد می‌شود که در حال صحبت با باب است و هیچ بازنواختی در کار نیست. در واقع آلیس همین چند میلی‌ثانیه قبل R_{A2} را تولید کرده. منظور از پیغام 5 آن است که باب متقاعد گردد واقعاً در حال صحبت با آلیس است و هیچ بازنواختی در بین نیست. با وادار کردنِ هر دو طرفِ ارتباط به ساختنِ یک چالش و پاسخ به طرفِ دیگر، احتمالِ هر نوع حمله‌ی بازنواخت از میان می‌رود.

1. Mutual authentication 2. Multiway challenge-response protocol
3. Needham-Schroeder authentication



شکل ۸-۴۱ پروتکل اُتوی ـ ریز (کمی ساده شده).

هر چند این پروتکل به نظر خوب می‌رسد ولی یک نقص کوچک دارد. اگر ترودی ترتیبی بدهد که یک کلید نشست قدیمی از یک متن آشکار را به دست آورَد، می‌تواند یک نشست جدید را با باب راه‌اندازی کند. به این صورت که پیغام 3 که متناظر با کلیدِ مورد توافق طرفین است را بازنواخت می‌کند و باب را متقاعد می‌سازد که آلیس است (Denning و Sacco، ۱۹۸۱). در این زمان ترودی می‌تواند بدون انجام کنترل‌های دسترسی قانونی، حساب بانکی آلیس را غارت کند.

نیدهام و شرودر (۱۹۸۷) بعدها پروتکلی را منتشر کردند که این مشکل را تصحیح می‌کرد. در همان نشریه و همان شماره، اُتوی و ریز (۱۹۸۷)[1] نیز پروتکلی را منتشر کردند که این مشکل را با روش کوتاه‌تری حل می‌کرد. شکل ۸-۴۱ یک پروتکل اُتوی ـ ریز را با کمی ویرایش نشان می‌دهد.

در پروتکل اُتوی ـ ریز آلیس کار را با تولید یک جفت عدد تصادفی آغاز می‌کند: R که به عنوان یک شناسه‌ی عمومی استفاده خواهد شد، و R_A که آلیس از آن جهت چالش با باب استفاده خواهد کرد. هنگامی‌که باب این پیغام را می‌گیرد، با استفاده از بخش کدگذاری شده از پیغام آلیس، و یک بخشِ مشابه از پیغام خودش، یک پیغام جدید می‌سازد. هر دو بخشِ کدگذاری شده با K_A و K_B که آلیس و باب را تعیین هویت می‌کنند، دربردارنده‌ی شناسه‌ی عمومی، و دربردارنده‌ی یک چالش هستند.

مرکز KDC کنترل می‌کند که R در هر دو بخش یکسان باشد. درصورتی‌که ترودی از طریق R در پیغام 1 مداخله کرده باشد یا بخشی از پیغام 2 را تغییر داده باشد، آنگاه R در هر دو بخش یکسان نخواهند بود. اگر دو R_S با هم جور باشند، KDC باور می‌کند که پیغامِ درخواست که از باب رسیده است، معتبر است. در این زمان، KDC یک کلیدِ نشست تولید می‌کند و آن را دو بار کدگذاری می‌کند، یک بار برای آلیس و یک بار برای باب. هر پیغام دربردارنده‌ی عدد تصادفیِ دریافت‌کننده است تا گواهی باشد بر این‌که KDC آن را تولید کرده، نه ترودی. در این لحظه هم آلیس و هم باب مالک یک کلید نشستِ واحد هستند و می‌توانند ارتباط را آغاز کنند. اولین باری که آن‌ها پیغام‌های داده‌ای را با یکدیگر تبادل کنند، هر کدامشان می‌توانند مشاهده کنند که دیگری یک کپیِ همسان از K_S دارد، لذا تصدیق هویت تکمیل شده است.

1. Otway and Rees (1987)

۸-۷-۴ تصدیق هویت با استفاده از کربروس

کربروس[1] یک پروتکل تصدیق هویت است که در بسیاری از سیستم‌های واقعی از آن استفاده می‌شد (شامل ویندوز ۲۰۰۰ و نگارش‌های قبلی آن). این پروتکل مبتنی بر نوعی پروتکل نیدهام ـ شرودر است. نام این پروتکل از نام یک سگ چند ـ سر در اسطوره‌شناسی یونان گرفته شده است که نگهبان ورودیه‌ی هادس[2] بود (البته احتمالاً در برابرِ خروج از این سرزمین نگهبانی می‌داده، نه ورود به آن!). کربروس در دانشگاه M.I.T طراحی شد، با این هدف که کاربرانِ ایستگاه‌های کاری بتوانند به روشی ایمن به منابع شبکه دسترسی داشته باشند. بزرگ‌ترین تفاوت آن با نیدهام ـ شرودر این است که کربروس فرض می‌کند تمام کلاک‌ها به خوبی با هم همگام (synchronized) شده‌اند. این پروتکل نگارش‌های (iterations) متعددی دارد. یکی از آن‌ها V5 است که در صنعت خیلی مورد استفاده قرار گرفته و در RFC 4120 تعریف شده است. نگارش قبلیِ آن، یعنی V4، بعد از نقص‌های جدی‌ای که در آن پیدا شد، بالاخره از دور خارج گردید (Yu و همکاران، ۲۰۰۴). نگارش V5 نسبت به V4، تغییرات کوچکِ متعددی در پروتکل و نیز تعدادی ویژگی‌های توسعه‌ای دارد، از قبیلِ این حقیقت که: دیگر به DES که اکنون منسوخ شده است، وابسته نیست. برای اطلاعات بیشتر به Neuman و Ts'o (۱۹۹۴) مراجعه نمایید.

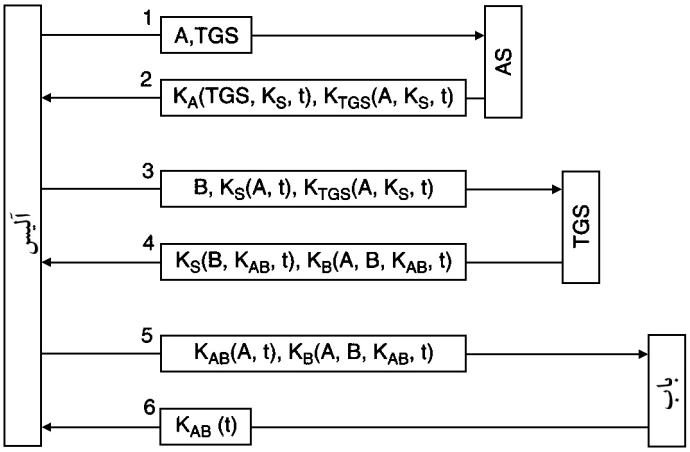
کربروس علاوه بر آلیس (که یک ایستگاه کاریِ مشتری است) با سه خدمتگزار دیگر نیز سروکار دارد:

۱. خدمتگزار تصدیق هویت (AS: Authentication Server): کاربران را در ورود به سیستم (login) راستی‌آزمایی می‌کند.
۲. خدمتگزار اعطا کننده‌ی بلیت (TGS: Ticket-Granting Server): "سند شناسایی بلیت‌ها" را صادر می‌کند.
۳. باب (به عنوان خدمتگزار): کاری که آلیس می‌خواهد را واقعاً انجام می‌دهد.

خدمتگزار AS شبیه یک KDC است، از این نظر که با هر کاربر در یک رمز عبورِ سرّی شریک است. وظیفه‌ی TGS صدور بلیت‌هایی است که بتوانند خدمتگزارهای واقعی را متقاعد کنند که دارنده‌ی بلیتِ TGSواقعاً همان کسی است که ادعا می‌کند.

به منظور آغاز یک نشست، آلیس در پشت یک ایستگاه کاری عمومیِ دلخواه می‌نشیند و نام خود را تایپ می‌کند. ایستگاه کاری، همان‌طور که در پیغام 1 در شکل ۸-۴۲ نشان داده شده، نام او و نام TGS را در یک متن آشکار به AS ارسال می‌کند. آنچه برمی‌گردد، یک کلید نشست و یک بلیت است که با $K_{TGS}(A, K_S, t)$ نشان داده می‌شود و برای TGS ایجاد شده است. کلید نشست با استفاده از

1. Kerberos 2. Hades در اسطوره‌های یونانی، به معنای "سرزمین مردگان (جهنم)" است. مترجم.



شکل ۸-۴۲ عملکرد کربروس V5 (نگارش ۵).

کلید سرّیِ آلیس رمزگذاری می‌شود، لذا تنها آلیس می‌تواند آن را رمزبَرداری کند. فقط هنگامی‌که پیغام 2 می‌رسد، ایستگاه کاری از آلیس رمز عبورش را می‌خواهد ــ نه قبل از آن. از این رمز عبور برای تولید K_A استفاده می‌شود تا پیغام 2 رمزبَرداری شده و کلید نشست به دست بیاید.

در این زمان، ایستگاه کاری رمز عبور آلیس را رونویسی (overwrite) می‌کند تا مطمئن شود که این رمز نهایتاً فقط برای چند میلی‌ثانیه درون ایستگاه کاری وجود دارد. اگر ترودی تلاش کند تا بعنوان آلیس login کند، رمز عبوری که تایپ می‌کند اشتباه بوده و ایستگاه کاری، این موضوع را تشخیص خواهد داد زیرا بخش استاندارد از پیغام 2 غیرصحیح خواهد بود.

آلیس بعد از login می‌تواند به ایستگاه کاری بگوید که می‌خواهد به خدمتگزار فایل (باب) متصل گردد. سپس ایستگاه کاری جهت درخواستِ یک بلیت برای استفاده جهت برقراری ارتباط با باب، پیغام 3 را به TGS ارسال می‌کند. عنصر کلیدی در این درخواست عبارت است از بلیت $K_{TGS}(A, K_S, t)$ که با کلید سرّیِ TGS رمزگذاری شده و برای اثباتِ این‌که ارسال کننده حقیقتاً آلیس است، از آن استفاده می‌شود. خدمتگزار TGS با تولید یک کلید نشست (K_{AB}) برای آلیس (تا آلیس هنگام برقراری ارتباط با باب، از آن استفاده کند) به پیغام 4 پاسخ می‌دهد. دو نسخه از آن برگردانده می‌شوند. اولی فقط با K_S رمزگذاری شده لذا آلیس می‌تواند آن را بخواند. دومی یک بلیت دیگر است که با کلید باب (K_B) رمزگذاری شده، بنابراین باب می‌تواند آن را بخواند.

ترودی می‌تواند پیغام 3 را کپی کند و تلاش نماید تا مجدداً از آن استفاده کند، اما توسط برچسب زمانیِ t که رمزگذاری شده و همراه پیغام ارسال می‌گردد، این تلاش دفع می‌شود. ترودی نمی‌تواند برچسب زمانی را با یک برچسب زمانیِ جدیدتر عوض کند زیرا K_S (کلید نشستی که آلیس از آن برای صحبت با TGS استفاده می‌کند) را نمی‌داند. حتی اگر ترودی سریعاً پیغام 3 را بازنواخت

کند، تمام آنچه که به دست می‌آورَد یک کپی دیگر از پیغام 4 است. ترودی نه در بار اول و نه در بار دوم نمی‌تواند این پیغام‌ها را رمزبَرداری کند.

اکنون آلیس می‌تواند K_{AB} را به وسیله‌ی بلیت جدید به باب ارسال کند تا یک نشست با او برقرار نماید (پیغام 5). این تبادل نیز برچسب زمانی خورده است. پاسخِ اختیاری (پیغام 6) به آلیس ثابت می‌کند واقعاً در حال صحبت با باب است، نه ترودی.

بعد از این سری از تبادل‌ها، آلیس می‌تواند تحت پوشش K_{AB} با باب ارتباط داشته باشد. اگر آلیس بعداً تصمیم به صحبت با خدمتگزار دیگری به نام کارول[1] بگیرد، کافی است پیغام 3 را به TGS تکرار کند، فقط این بار به جای B شناسه‌ی C را مشخص می‌کند. بلافاصله TGS با یک بلیتِ رمزگذاری شده با K_C پاسخ می‌دهد. آلیس می‌تواند آن را به کارول ارسال کند و کارول هم آن را به عنوان اثباتِ این‌که پیغام از طرف آلیس آمده، می‌پذیرد.

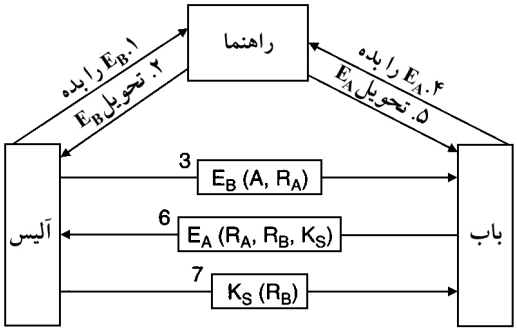
نکته‌ی تمام این کارها آن است که اکنون آلیس می‌تواند با روشی امن به تمام خدمتگزارها در سرتاسر شبکه دسترسی داشته باشد و رمز عبور او هرگز در شبکه منتقل نمی‌گردد. در حقیقت او فقط مجبور است برای چند میلی‌ثانیه در ایستگاه کاریِ خودش باشد. با این حال توجه داشته باشید که هر خدمتگزاری، تصدیق هویتِ خودش را انجام می‌دهد. هنگامی‌که آلیس بلیتش را به باب ارائه می‌دهد، این فقط به باب ثابت می‌کند که چه کسی بلیت را فرستاده است. درست توجه کنید. این‌که آلیس مُجاز به چه عملیاتی باشد، بستگی به باب دارد.

از آن‌جا که طراحانِ کربروس انتظار نداشته‌اند که کلِ دنیا کارشان را تنها به یک خدمتگزار تصدیق هویتِ منفرد بسپارند، لذا اجازه‌ی داشتنِ چندین **قلمرو**[2] را داده‌اند به طوری که هر قلمرو دارای AS و TGS مربوط به خود است. آلیس به منظور گرفتنِ بلیت برای یک خدمتگزار در یک قلمرو دور، از TGS خودش درخواست یک بلیت می‌کند که از طرف TGS واقع در قلمرو دور پذیرفته شود. اگر TGS دور، نزد TGS محلی ثبت شده باشد (با همان روشی که در مورد خدمتگزارانِ محلی داریم)، TGS محلی به آلیس یک بلیت خواهد داد که در TGS دور معتبر می‌باشد. آلیس به این ترتیب می‌تواند کارش را در آن مکان انجام دهد، از جمله گرفتنِ بلیت‌هایی برای خدمتگزارهای آن قلمرو. توجه داشته باشید که برای آن‌که طرف‌های واقع در دو قلمرو با هم کار کنند، هر طرف بایستی به TGS طرفِ دیگر اعتماد داشته باشد، وگرنه قادر به کار با یکدیگر نخواهند بود.

۸–۷–۵ تصدیق هویت با استفاده از رمزنگاریِ کلید ـ عمومی

تصدیق هویتِ متقابل با استفاده از رمزنگاریِ کلید ـ عمومی نیز می‌تواند انجام شود. در شروع کار، آلیس نیاز به گرفتنِ کلید عمومی باب دارد. همان‌طور که در شکل ۸–۴۳ در پیغام 3 مشاهده می‌شود، در صورت وجودِ یک PKI همراه با یک خدمتگزارِ راهنما[3] که گواهینامه‌های مربوط به کلیدهای عمومی

1. Carol 2. Realm 3. Directory server



شکل ۸-۴۳ تصدیق هویت متقابل با استفاده از رمزنگاری کلید ـ عمومی.

را در اختیار قرار می‌دهد، آلیس خواهد توانست کلید عمومیِ باب را درخواست کند. پاسخ (در پیغام 2) یک گواهینامه‌ی X.509 است که دربردارنده‌ی کلید عمومی باب می‌باشد. بعد از آن‌که آلیس صحت امضا را راستی‌آزمایی کرد، یک پیغام به باب ارسال می‌کند که شامل شناسه‌ی خودش و یک نانس است.

وقتی باب این پیغام را دریافت کند، نمی‌داند آیا پیغام از طرف آلیس است یا از طرف ترودی. اما او کار را ادامه می‌دهد و کلید عمومی آلیس را از خدمتگزار راهنما درخواست می‌کند (پیغام 4) و خیلی زود پیغام 5 را دریافت می‌کند. باب سپس پیغام 6 را به آلیس ارسال می‌کند. این پیغام شامل R_A آلیس، نانسِ مربوط به خود باب، R_B ، و کلید نشست (K_S) است.

هنگامی‌که آلیس پیغام 6 را می‌گیرد، با استفاده از کلید خصوصی‌اش آن را رمزبَرداری می‌کند. آلیس R_A را در آن مشاهده می‌کند که سبب دلگرمی‌اش می‌گردد. این پیغام بایستی از طرف باب آمده باشد چون ترودی هیچ راهی برای اطلاع یافتن از R_A ندارد. به‌علاوه این پیغام باید تازه باشد، و یک بازنواخت نیست زیرا آلیس همین حالا R_A را به باب ارسال کرده است. آلیس با فرستادنِ پیغام 7 موافقت خود را با نشست اعلام می‌کند. وقتی باب R_B را می‌بیند که با کلید نشستی که باب همین حالا ایجادش کرده، رمزگذاری شده است، متوجه می‌شود آلیس پیغام 6 را دریافت کرده و R_A را راستی‌آزمایی کرده است. باب حالا خوشحال است!

ترودی برای خرابکاری در این پروتکل چه کاری می‌تواند بکند؟ او می‌تواند در پیغام 3 دستکاری کند و باب را وادار به تحقیق از آلیس نماید، اما آلیس R_A ای را مشاهده خواهد کرد که خودش نفرستاده، پس دیگر ادامه نخواهد داد. ترودی نمی‌تواند پیغام 7 را جعل کرده و به باب برگردانَد زیرا R_B یا K_S را نمی‌داند و بدون کلید خصوصی آلیس نمی‌تواند آن‌ها را مشخص سازد. ترودی بدشانسی آورده!

۸-۸ امنیت ایمیل

هنگامی که یک پیغام ایمیل بین دو سایتی که از هم دور هستند ارسال می‌شود، در اصل در مسیرش از چند دوجین ماشین ترانزیت می‌شود. هر کدام از این ماشین‌ها می‌توانند پیغام را خوانده و برای مصرفِ

بعدی آن را ذخیره کنند. در عمل، علی‌رغم آنچه بسیاری از مردم خیال می‌کنند، چیزی به نام حفظ حریم خصوصی وجود ندارد. معهذا افراد متعددی وجود دارند که مایلند بتوانند ایمیلی ارسال کنند که غیر از گیرنده‌ی مورد نظر، هیچ شخص دیگری نتواند آن را بخواند: نه رئیس‌شان و نه حتی سیستم حکومتی‌شان. همین تمایل است که سبب شده افراد و گروه‌هایی مفاهیم رمزنگاری که پیشتر مطالعه کردیم را به ایمیل اِعمال کنند و یک ایمیل امن ایجاد نمایند. در بخش‌های بعدی یک سیستم ایمیل امن که بسیار مورد استفاده است، به نام PGP را مطالعه خواهیم کرد و به اجمال به سیستم دیگری به نام S/MIME نیز خواهیم پرداخت. برای اطلاعات بیشتر درباره‌ی ایمیلِ امن، Kaufman و همکاران (۲۰۰۲) و Schneier (۱۹۹۵) را ببینید.

۸-۸-۱ PGP ــ محرمانگیِ نسبتاً خوب

اولین مثال ما، **PGP** (**محرمانگیِ نسبتاً خوب**[1]) در اصل محصول اندیشه‌ی یک فرد است: فیل زیمرمان[2] (a۱۹۹۵، b۱۹۹۵). زیمرمان یک مدافع حفظ حریم خصوصی است که شعارش این است: "اگر حفظ حریم خصوصی قدغن شود، فقط قانون‌شکنان حریم خصوصی خواهند داشت." بسته‌ی PGP که در سال ۱۹۹۱ انتشار یافت، یک بسته‌ی کامل امنیتِ ایمیل است که محرمانگی، تصدیق هویت، امضاهای دیجیتالی، و فشرده‌سازی را تماماً در قالبی با استفاده‌ی آسان فراهم می‌کند. علاوه بر این، بسته‌ی کامل که دربردارنده‌ی همه‌ی کدِ source است، بدون نیاز به پرداخت هزینه و از طریق اینترنت توزیع می‌شود. به واسطه‌ی کیفیتِ آن، هزینه‌ی (صفر)، و راحتیِ دسترس‌پذیری بر روی platform های سیستم عاملِ یونیکس، لینوکس، ویندوز، و مَک (Mac OS)، امروزه به طور گسترده مورد استفاده می‌باشد.

بسته‌ی PGP داده را با به کار بردنِ یک رمز بلوکی به نام **IDEA** (**الگوریتم بین‌المللی رمزگذاری داده**[3]) که از کلیدهای ۱۲۸ـ بیتی استفاده می‌کند، رمزگذاری می‌کند. این رمز در سوئیس ساخته شد، آن هم زمانی که عیب‌هایی در DES مشاهده شده بود و AES هنوز ایجاد نشده بود. به لحاظ مفهومی، IDEA شبیه DES و AES است: طی یک سری راند، بیت‌ها را مخلوط می‌کند، اما توابعِ مخلوط کننده‌ی آن با DES و AES متفاوت هستند. مدیریت کلید از RSA ، و جامعیت داده از MD5 استفاده می‌کنند. این‌ها عناوینی هستند که تاکنون بررسی کرده‌ایم.

بسته‌ی PGP از روز اولی که آمد، ایجاد آشفتگی کرد (Levy، ۱۹۹۳). زیرا زیمرمان هیچ کاری در جهت منع مردم از قرار دادنِ PGP بر روی اینترنت انجام نمی‌داد، به طوری که جماعت در همه جای دنیا می‌توانستند آن را دریافت کنند. به همین خاطر دولت ایالات متحده ادعا کرد که زیمرمان قوانین ایالات متحده در ارتباط با ممنوعیتِ صادرات جنگ‌افزار را نقض کرده است. تحقیقات دولت ایالات متحده از زیمرمان تا ۵ سال ادامه یافت اما بالاخره قطع گردید، و احتمالاً به دو علت. اول آن‌که

1. Pretty Good Privacy 2. Phil Zimmermann 3. International Data Encryption Algorithm

زیمرمان خودش PGP را روی اینترنت نگذاشته بود، پس وکیل او ادعا کرد که *زیمرمان* هرگز چیزی را صادر نکرده است (و در این صورت کلاً ایجاد یک سایت وب برای صادرات، وجهی نداشت). دوم آن‌که سرانجام دولت به این درک رسید که برنده شدن دراین محاکمه به معنای آن است که هیئت منصفه را متقاعد کنند سایت وبی که شامل یک برنامه‌ی محرمانگیِ قابل پایین‌گذاری می‌باشد، با استفاده از قانون قاچاق سلاح که صادر کردنِ ادوات جنگی از قبیل تانک‌ها، زیردریایی‌ها، هواپیماهای نظامی، و جنگ‌افزارهای هسته‌ای را منع می‌کند، پوشش داده می‌شود. حتی سال‌ها تبلیغات منفی نیز احتمالاً کمک چندانی نکرد.

این مطلب را هم در حاشیه بگوییم که قوانین صادرات عجیب و غریب هستند، البته در حالت خوش‌بینانه. دولت عقیده داشت که قرار دادنِ کد در یک سایت وب، یک صادرات غیرقانونی است و پنج سالِ تمام زیمرمان را آزار داد. از سوی دیگر وقتی فردی پیدا شد که کدِ source کاملِ PGP را به زبان C و به صورت یک کتاب منتشر کرد (آن هم با فونت درشت و با قرار دادنِ مجموع مقابله‌ای (checksum) بر روی هر صفحه، تا بررسی آن راحت‌تر باشد) و سپس این کتاب را صادر نمود، این کار برای دولت خوشایند بود زیرا کتاب‌ها جزء ادوات جنگی رده‌بندی نشده‌اند. قدرت شمشیر بیشتر از قلم است، دست کم برای عمو سام!

مشکل دیگری نیز در ثبت حق انحصاریِ PGP پیش آمد. شرکت RSA Security که مالکیت ثبت RSA را دارد دعوایی اقامه کرد که بر اساسِ آن استفاده از الگوریتم RSA در PGP، حق مالکیت این شرکت را نقض کرده است. اما این مشکل با نگارش‌های از 2.6 به بعد برطرف گردید. به‌علاوه، PGP از الگوریتم ثبت شده‌ی دیگری به نام IDEA استفاده می‌کند، که البته آن هم در ابتدا مسائلی را باعث شد.

از آن‌جا که PGP کدِ باز است، لذا افراد و گروه‌های مختلفی آن را ویرایش کرده‌اند و انواع نگارش‌ها از آن ایجاد شده است. بعضی از آن‌ها طوری طراحی شده‌اند که از قوانین مربوط به جنگ‌افزارها خلاص شوند. تعدادی دیگر بر اجتناب از به‌کار بردنِ الگوریتم‌های دارای حق ثبتِ انحصاری تمرکز کرده‌اند. و تعدادی هم می‌خواهند آن را به یک محصولِ تجاریِ کد ـ بسته برگردانند. گرچه قوانین مربوط به جنگ‌افزارها تاحدودی لیبرال‌تر شده‌اند (در غیر این صورت محصولاتی که از AES استفاده می‌کنند هرگز نمی‌توانستند از ایالات متحده صادر شوند)، و حق ثبتِ انحصاری RSA نیز در سپتامبر ۲۰۰۰ زمان انقضایش سررسید، میراث به جا مانده از تمام این مسائل آن است که تعدادی نگارش‌های ناسازگار از PGP، با اسامی گوناگون، در جریان هستند. بررسی زیر بر PGP کلاسیک تمرکز دارد که قدیمی‌ترین و ساده‌ترین نگارش است. نگارش پُرطرفدار دیگری به نام Open PGP وجود دارد که در RFC 2440 توصیف شده است. یک نگارش دیگر نیز وجود دارد، به نام GNU Privacy Guard.

روش PGP عمداً به جای ابداع الگوریتم‌های رمزنگاری جدید، از الگوریتم‌های موجود استفاده می‌کند. روشPGP تا اندازه‌ی زیادی بر پایه‌ی الگوریتم‌هایی است که در برابر بازبینی‌های گسترده و موشکافانه دوام آورده‌اند و از طرف هیچ بنگاه دولتی که تلاش در تضعیف آن‌ها داشته باشند، طراحی نشده و از این بنگاه‌ها تأثیر نمی‌پذیرند. از دید افرادی که نسبت به دولت بدگمان هستند، این ویژگی یک امتیاز بزرگ است.

روش PGP از فشرده‌سازی متن، رازپوشی، و امضاهای دیجیتالی حمایت کرده و همچنین امکانات گسترده‌ای برای مدیریت کلید فراهم می‌کند، اما جای تعجب است که امکانی برای ایمیل ندارد. مثل پیش ـ پردازشگری[1] است که یک متن آشکار را به عنوان ورودی گرفته و یک متن رمزیِ امضا شده در base64 را به عنوان خروجی تولید می‌کند. البته این خروجی بعداً می‌تواند ایمیل شود. بعضی پیاده‌سازی‌های PGP به عنوان گام نهایی، برای آن‌که واقعاً پیغام را ارسال کنند، یک کارگزار کاربر را فراخوانی می‌کنند.

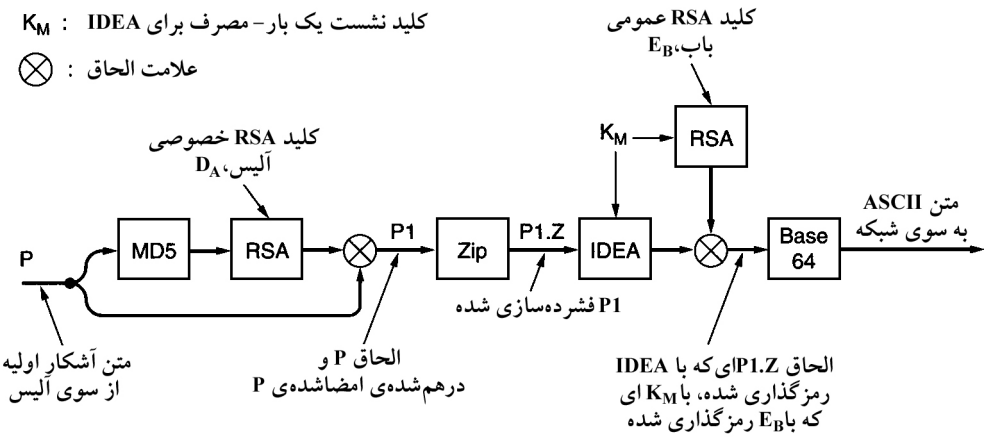
برای مشاهده‌ی نحوه‌ی عمل PGP، اجازه دهید مثال شکل ۸–۴۴ را در نظر بگیریم. در این مثال آلیس می‌خواهد یک پیغام متن آشکارِ امضا شده (P) را به روشی امن به باب ارسال کند. هم آلیس و هم باب دارای کلیدهای RSA خصوصی (D_X) و عمومی (E_X) می‌باشند. بیایید فرض کنیم که هر کدام از آن‌ها کلید عمومیِ نفر دیگر را می‌داند؛ به زودی مدیریت کلید PGP را پوشش خواهیم داد.

آلیس کار را با احضارِ برنامه‌ی PGP بر روی کامپیوترش آغاز می‌کند. برنامه‌ی PGP ابتدا پیغام آلیس (یعنی P) را با استفاده از MD5 درهم‌سازی می‌کند و سپس پیغامِ درهم‌سازی شده‌ی حاصل را با استفاده از کلید RSA خصوصیِ خودش (D_A) رمزگذاری می‌نماید. وقتی باب نهایتاً پیغام را می‌گیرد، می‌تواند پیغام درهم‌سازی شده را با کلید عمومی آلیس رمزبَرداری کرده و صحتِ این پیغامِ درهم‌سازی شده را راستی‌آزمایی کند. حتی اگر شخص دیگری (مانند ترودی) بتواند در این مرحله پیغام درهم‌سازی شده را به دست آورَد، قدرت MD5 ضمانت می‌کند که به لحاظ محاسبات کامپیوتری، ایجاد یک پیغام دیگر با همان درهم‌سازیِ MD5 نشدنی است.

اکنون پیغام درهم‌سازی شده‌ی رمزگذاری شده، و پیغام اصلی درون یک پیغامِ منفرد به نام $P1$ الحاق شده و با استفاده از برنامه‌ی ZIP فشرده‌سازی می‌شوند. برنامه‌ی ZIP از الگوریتم Ziv-Lempel استفاده می‌کند (Ziv و Lempel، ۱۹۷۷). خروجیِ این مرحله را $P1.Z$ بنامید.

سپس PGP با دادنِ یک پیام‌واره[2]، از آلیس یک ورودیِ تصادفی می‌خواهد. هم محتوا و هم سرعت تایپ کردن در تولیدِ یک کلید پیغام IDEA ۱۲۸ـ بیتی به نام K_M مورد استفاده قرار می‌گیرد (K_M کلید نشست نامیده می‌شود که در واقع یک نامِ بی‌مسمی است زیرا هنوز هیچ نشستی وجود ندارد). حالا از K_M برای رمزگذاریِ $P1.Z$ با IDEA در مودِ بازخوردِ رمز استفاده می‌شود. علاوه بر این، K_M با کلید عمومی باب، یعنی E_B، رمزگذاری می‌شود. سپس این دو مؤلفه به هم الحاق شده و

1. Preprocessor 2. Prompt



شکل ۸-۴۴ بهره‌برداری از PGP جهت ارسال یک پیغام.

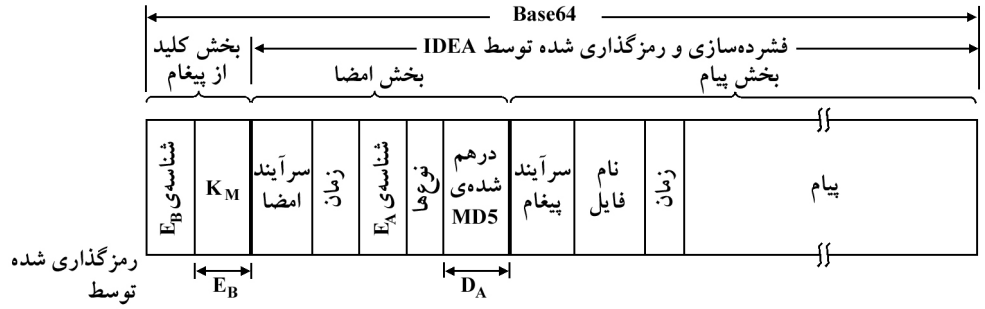
به base64 تبدیل می‌شوند (همان‌طور که در بخش MIME در فصل ۷ توضیح دادیم). پیغامِ حاصل فقط شامل حروف، ارقام، و علائم + و / و = است. معنایش آن است که این پیغام می‌تواند در یک بدنه‌ی RFC 822 قرار داده شود و انتظار داشته باشیم که بدون اصلاح و تعدیل برسد.

هنگامی‌که باب این پیغام را دریافت کند، کدگذاریِ base64 را معکوس کرده و کلید IDEA را با استفاده از کلید خصوصی RSA خود، رمزبَرداری می‌کند. با استفاده از این کلید، باب پیغام را رمزبَرداری می‌کند تا به *P1.Z* برسد. بعد از آن‌که *P1.Z* از حالت فشرده خارج شد، باب متن آشکار را از متن درهمِ رمزگذاری شده جدا می‌کند و متن درهم شده را با استفاده از کلید عمومی آلیس رمزبَرداری می‌کند. اگر متن آشکارِ درهم شده، با محاسبه‌ی MD5 ای که خودِ باب انجام داده سازگار باشد، در این صورت باب متوجه می‌شود که *P* یک پیغام صحیح است و این پیغام از طرف آلیس آمده.

توجه به این نکته اهمیت دارد که این‌جا RSA فقط در دو مورد به کار گرفته می‌شود: برای رمزگذاری کردنِ متن درهم شده‌ی MD5 ۱۲۸- بیتی و برای رمزگذاری کردنِ کلید IDEA ۱۲۸- بیتی. گرچه RSA کُند است، ولی با یک حجمِ داده‌ی بزرگ سروکار ندارد و فقط باید ۲۵۶ بیت را کدگذاری کند. علاوه بر این، تمام بیت‌های متن آشکار به طرز فوق‌العاده زیادی تصادفی هستند لذا ترودی نیاز به حجم کار قابل توجهی خواهد داشت تا فقط تعیین کند آیا کلیدی که حدس زده، صحیح است یا خیر. رمزگذاریِ اصلی توسط IDEA انجام می‌گیرد، که ده‌ها برابر سریع‌تر از RSA است. بنابراین PGP امنیت، فشرده‌سازی، و امضای دیجیتالی را تأمین می‌کند، آن هم بسیار کارآمدتر از نظامی که در شکل ۸-۱۹ نشان داده شده است.

روش PGP از چهار طول کلیدِ RSA حمایت می‌کند. بر عهده‌ی کاربر است که مناسب‌ترین مورد را انتخاب نماید. این طول‌ها عبارتند از:

۱. خودمانی (Casual) (به طول ۳۸۴ بیت): امروزه به راحتی قابل شکستن است.



شکل ۸–۴۵ یک پیغام PGP.

۲. تجاری (Commercial) (به طول ۵۱۲ بیت): توسط سازمان‌های سه ـ حرفی قابل شکستن است!!

۳. نظامی (Military) (به طول ۱۰۲۴ بیت): هیچکس بر روی کره‌ی زمین قادر به شکستنِ آن نیست.

۴. مربوط به موجودات بیگانه‌ی فضایی (Alien) (به طول ۲۰۴۸ بیت): هیچ کس، حتی در سایر کرات نیز قادر به شکستنِ آن نیست.

از آنجا که از RSA فقط برای دو محاسبه‌ی کوچک استفاده می‌شود لذا همه بایستی در تمام حالات از کلیدهایی با توانمندیِ مورد چهارم استفاده کنند.

فرمت یک پیغام کلاسیکِ PGP در شکل ۸–۴۵ نشان داده شده. فرمت‌های متعددِ دیگری نیز به کار می‌روند. پیغام دارای سه بخش است که به ترتیب عبارتند از کلید IDEA، امضا، و پیغام. بخش کلید نه تنها کلید را در خود دارد، بلکه یک شناسه‌ی کلید نیز دارد زیرا کاربران مُجاز به داشتنِ چندین کلید عمومی می‌باشند.

بخش امضا شامل یک سرآیند است که مورد نظر ما نمی‌باشد. به دنبال سرآیند این موارد قرار دارند: یک برچسب زمانی، شناسه‌ی کلید عمومیِ ارسال کننده (که می‌تواند برای رمزبَردار‌ی از امضای درهم‌سازی شده، به کار رود)، بعضی اطلاعاتِ مربوط به نوع (که الگوریتم‌های مورد استفاده را مشخص می‌سازند، تا MD6 و RSA2 بتوانند استفاده شوند)، و خودِ درهم‌سازیِ رمزگذاری شده.

همچنین بخش پیغام دربردارنده‌ی این موارد می‌باشد: یک سرآیند، یک اسم پیش‌فرض برای فایل (درصورتی‌که دریافت‌کننده، فایل را بر روی دیسک بنویسد، از این اسم پیش‌فرض برای فایل استفاده می‌شود)، یک برچسب زمانی مربوط به ایجاد پیغام، و بالاخره: خودِ پیغام.

در PGP توجه زیادی به مدیریت کلید شده است زیرا در تمام سیستم‌های امنیتی، مدیریت کلید پاشنه‌ی آشیل سیستم است. مدیریت کلید به ترتیبی که اینک می‌گوییم عمل می‌کند. هر کاربر دو ساختار داده را به طور محلی نگهداری می‌کند: یک حلقه‌ی کلید خصوصی و یک حلقه‌ی کلید عمومی. **حلقه‌ی کلید خصوصی**[1] شامل یک یا چند جفت کلید خصوصی/عمومیِ شخصی می‌باشد. علتِ آن‌که از چندین

1. Private key ring

جفت کلید به ازای هر کاربر حمایت می‌شود آن است که به کاربران اجازه داده شود تا به صورت دوره‌ای کلیدهای عمومی‌شان را عوض کنند. یا آن‌که هر زمان فکر کردند در خطرِ کشف رمز قرار گرفته‌اند، بتوانند کلیدهای عمومی‌شان را تغییر دهند، بدون آن‌که پیغام‌هایی که در راه هستند یا در حال آماده‌سازی می‌باشند، غیرمعتبر گردند. هر جفت کلید دارای یک شناسه است که با آن مرتبط شده تا ارسال کننده‌ی پیغام بتواند به گیرنده بگوید از کدام کلید عمومی برای رمزگذاریِ آن استفاده شده است. شناسه‌های پیغام از ۶۴ بیتِ با ارزش پایین در کلید عمومی تشکیل می‌شوند. مسئولیتِ اجتناب از تلاقی در شناسه‌های کلید ـ عمومی، با خودِ کاربران است. کلیدهای خصوصیِ روی دیسک، با استفاده از یک رمز عبورِ مخصوص (به طول دلخواه) رمزگذاری می‌شوند تا در برابر حمله‌های دزدکی[1] محافظت شوند.

حلقه‌ی کلید عمومی[2] دربردارنده‌ی کلیدهای عمومیِ مکاتبه‌کنندگان با این کاربر است. این کلیدهای عمومی جهت رمزگذاری کلیدهای پیغامِ مرتبط با هر پیغام، مورد نیاز می‌باشند. هر درایه‌ی موجود بر روی حلقه‌ی کلید عمومی دربردارنده‌ی کلید عمومی، شناسه‌ی ۶۴ ـ بیتیِ آن، و همچنین نشانه‌ای از اعتمادِ فراوان کاربر به کلید می‌باشد.

مشکلی که در این‌جا مهار شده به این ترتیب است: فرض کنید کلیدهای عمومی بر روی تابلوهای اعلانات نگهداری می‌شوند. یک راه برای ترودی جهت خواندن ایمیل سرّیِ باب آن است‌که به این تابلوی اعلانات حمله کرده و کلید عمومی باب را با کلیدِ مورد نظر خودش عوض کند. بعداً، هنگامی‌که آلیس کلیدی که ظاهراً متعلق به باب است را واکشی می‌کند، ترودی می‌تواند یک حمله‌ی bucket brigade را راه‌اندازی کند.

برای اجتناب از چنین حمله‌هایی، یا دست کم برای به حداقل رساندنِ پیامدهای این‌گونه حمله‌ها، آلیس باید بداند که چقدر می‌تواند به آنچه بر روی حلقه‌ی کلید عمومی‌اش (حلقه‌ی کلید عمومیِ آلیس) با عنوان "کلید باب" نامیده می‌شود، اعتماد کند. اگر آلیس بداند که باب شخصاً یک CD-ROM که دربردارنده‌ی کلید است را به دست او داده است، خواهد توانست میزان اعتماد را به بالاترین مقدار برساند. چنین رویکردِ غیرمتمرکزی در مدیریت کلید ـ عمومی که کنترل را به کاربر می‌سپارد، سبب می‌شود PGP از نظام‌های PKI متمرکز، فاصله بگیرد.

معهذا، مردم بعضی مواقع کلیدهای عمومی را با انجام پرس‌وجو از یک خدمتگزار کلیدِ قابل اعتماد، به دست می‌آورَند. به همین دلیل بعد از آن‌که X.509 استانداردسازی شد، PGP از این گواهینامه‌ها درست همانند مکانیسم حلقه‌ی کلید عمومیِ PGP سنتی حمایت کرد.

S / MIME ۲-۸-۸

سرمایه‌گذاری IETF در بحث امنیت ایمیل که **S/MIME** (**Secure/MIME**) نام دارد، در RFC های 2632 تا 2643 شرح داده شده است. این سیاست‌گذاری تصدیق هویت، جامعیت داده، رازپوشی، و

1. Sneak attack (حمله‌ی استتاری) 2. Public key ring

عدم انکار را تأمین می‌کند. همچنین کاملاً انعطاف‌پذیر است و انواع الگوریتم‌های رمزگونه را حمایت می‌کند. تعجبی ندارد که با نامی که انتخاب شده، S/MIME به خوبی با MIME قابل جمع است. این موضوع امکانِ حفاظت از تمام انواع پیغام‌ها را می‌دهد. انواعی از سرآیندهای MIME جدید تعریف می‌شوند، به طور مثال، برای نگهداشتنِ امضاهای دیجیتالی.

در S/MIME یک سلسله مراتبِ غیرقابل انعطاف که با یک ریشه‌ی منفرد شروع شود، وجود ندارد. وجود یک سلسله مراتبِ غیرقابلِ انعطاف، یکی از مشکلاتی بود که یک سیستمِ قدیمی‌تر به نام PEM (ایمیل پیشرفته‌ی خصوصی[1]) را محکوم به فنا کرد. به جای این حالت، کاربران می‌توانند چندین لنگرِ اعتماد داشته باشند. مادام که ردّ یک گواهینامه بتواند تا جایی گرفته شود که ما را به یک لنگرِ اعتماد برسانَد که کاربر به آن باور داشته باشد، آن گواهینامه معتبر فرض می‌شود. در S/MIME از الگوریتم‌های استاندارد و پروتکل‌هایی استفاده می‌شود که تاکنون آن‌ها را بررسی کرده‌ایم، لذا دیگر در این‌جا به بحث در مورد آن‌ها نمی‌پردازیم. لطفاً برای جزئیات به RFC ها مراجعه نمایید.

۸–۹ امنیت وب

تاکنون دو حوزه‌ی مهم را مطالعه کرده‌ایم که در بحث امنیت، ضروری می‌باشند: ارتباطات و ایمیل. این مباحث را می‌توانید به عنوان سوپ و پیش‌غذا در نظر بگیرید. اکنون زمانِ پرداختن به موضوع اصلی است: امنیتِ وب. وب محلی است که امروزه بیشترین تعداد ترودی‌ها به آن متوسل شده و عمل پلید خود را انجام می‌دهند. در بخش‌های بعد به بعضی مسائل و مباحث مرتبط با امنیتِ وب خواهیم پرداخت.

امنیتِ وب تقریباً می‌تواند به سه بخش تقسیم شود. اول، شیءها (objects) و منابع چگونه به صورتی امن نامیده می‌شوند؟ دوم، اتصالاتِ امن و تصدیق هویت شده، چطور می‌توانند برقرار شوند؟ سوم، هنگامی‌که یک سایت وب به یک مشتری، تکه‌ای از کدِ قابل اجرا ارسال می‌کند، چه روی می‌دهد؟ بعد از نگاهی به تهدیدها، تمام این موارد را بررسی خواهیم کرد.

۸–۹–۱ تهدیدها

تقریباً هر هفته مطالبی را درباره‌ی مشکلات مربوط به امنیت سایت‌های وب در روزنامه‌ها می‌خوانیم. این وضعیت واقعاً بسیار ترسناک است. بگذارید مثال‌هایی از آنچه تاکنون روی داده است را ببینیم. اول آن‌که، صفحات خانگیِ تعداد زیادی از سازمان‌ها مورد حمله قرار گرفته و با صفحه‌هایی که کرَکِرها (کرک‌کننده‌ها cracker) انتخاب کرده‌اند، عوض شده‌اند. (مطبوعاتِ رایج، افرادی که به کامپیوترها نفوذ می‌کنند را "هکر" ("hacker") می‌نامند ولی بسیاری از برنامه‌نویسان، این اصطلاح را برای برنامه‌نویسانِ

1. Privacy Enhanced Mail

بزرگ به کار می‌برند. ما در اینجا این افراد را "کِرَکِر" می‌نامیم.) سایت‌هایی که کرک شده‌اند عبارتند از سایت‌های متعلق به یاهو، ارتش ایالات متحده، CIA، ناسا، و روزنامه‌ی نیویورک تایمز. در بیشتر موارد، کرَکرها فقط متنی بامزه در سایت قرار می‌دهند که ظرف چند ساعت قابل ترمیم و اصلاح است.

حالا بیایید موارد خیلی جدّی‌تر را بررسی کنیم. تعداد قابل توجهی از سایت‌ها به وسیله‌ی حمله‌های محروم‌سازی ـ از ـ سرویس از کار می‌افتند. در این حمله‌ها، کرَکرها آن سایت را با ترافیکی که روانه‌اش می‌کنند، گرفتار سیلاب کرده و آن سایت را از پاسخگویی به پرس‌وجوهای درست و موجه، بازمی‌دارند. غالباً این حمله از سوی تعداد زیادی ماشین تدارک دیده می‌شود که قبلاً توسط کرَکر مورد نفوذ قرار گرفته‌اند (حمله‌های DDoS). این حمله‌ها آنقدر متداول شده‌اند که دیگر حتی به جهتِ خبری، مورد توجه قرار نمی‌گیرند اما سایت‌های مورد حمله را می‌توانند متحملِ هزاران دلار خسارت نمایند.

در سال ۱۹۹۹ یک کرَکر سوئدی به سایت هات‌میلِ مایکروسافت نفوذ کرده و یک سایت mirror ایجاد کرد که به همه امکان می‌داد تا نام یک کاربرِ هات‌میل را تایپ کنند و تمام ایمیلِ جاری و ایمیلِ آرشیو آن کاربر را بخوانند.

در یک مورد دیگر، یک کرَکر ۱۹ ساله‌ی روسی به نام ماکسیم[1] وارد یک سایت وبِ تجارت الکترونیکی شده و شماره‌های ۳۰۰,۰۰۰ کارت اعتباری را سرقت نمود. سپس او با مالکینِ سایت تجاری تماس گرفت و به آن‌ها گفت اگر ۱۰۰,۰۰۰ دلار به او پرداخت نکنند، تمام شماره‌ی کارت‌های اعتباری را در اینترنت اعلان خواهد کرد. آن‌ها به باج‌خواهیِ او جواب ندادند و ماکسیم نیز واقعاً شماره‌ی کارت‌های اعتباری را اعلان کرد و خسارت بزرگی را به تعداد زیادی قربانیانِ بی‌گناه تحمیل نمود.

در ماجرایی دیگر، یک دانشجوی کالیفرنیاییِ ۲۳ ساله، از طرف یکِ آژانس خبری یک خبر ساختگی را به یک نشریه ایمیلی ارسال کرد با این مضمون که: شرکت Emulex دچار یک زیان فصلیِ بزرگ شده و مدیر عامل آن بلافاصله مجبور به استعفا شده است. ظرف چند ساعت سهام شرکت ۶۰٪ سقوط کرد و در نتیجه، سهامداران بیش از ۲ بیلیون دلار ضرر کردند. کسی که مرتکب این جرم شده بود، با فروش سهام، کمی قبل از ارسال اطلاعیه، یک چهارمِ میلیون دلار صاحب شد. هر چند این مورد، نفوذ به سایت‌وب نبود ولی واضح است که قرار دادنِ چنین اطلاعیه‌ای بر روی صفحه‌ی خانگی هر شرکت بزرگی، تأثیری مشابهِ این خواهد داشت.

موارد متعددی از این دست را (متأسفانه) درباره‌ی صفحات خانگی می‌توانیم سراغ بگیریم. اما اینک وقتِ آن رسیده که بعضی شیوه‌های مرتبط با امنیت شبکه را بررسی کنیم. برای اطلاعات بیشتر درباره‌ی تمامی انواع مشکلات امنیتی، به Anderson (۲۰۰۸b)؛ Stuttard و Pinto (۲۰۰۷)؛ و Schneier (۲۰۰۴) مراجعه نمایید. جستجو در اینترنت نیز موارد خاصِ متعددی را نشان خواهد داد.

1. Maxim

۸-۹-۲ نام‌گذاریِ اَمن

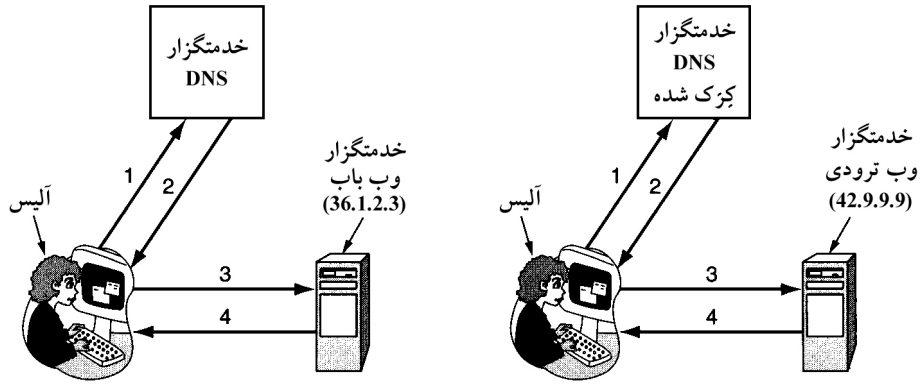
بیایید با موضوعی بسیار اساسی شروع کنیم: آلیس می‌خواهد سایت وب باب را بازدید کند. او URL باب را در مرورگرش تایپ می‌کند و چند ثانیه بعد، یک صفحه‌ی وب ظاهر می‌شود. اما آیا این بابَ است؟ ممکن است ترودی مجدداً به حقه‌های قدیمی‌اش دست زده باشد. برای مثال شاید ترودی تمام بسته‌های خروجیِ آلیس را رهگیری کرده و آن‌ها را بررسی می‌کند. هنگامی‌که ترودی یک درخواست *GET* در HTTP را به دست می‌آورَد که به مقصد سایت وبِ باب می‌رود، می‌تواند خودش به سایت باب رفته و صفحه را به دست آورَد، آن را مطابق میلش تغییر دهد و این صفحه‌ی جعلی را به آلیس برگردانَد. آلیس باخبر نخواهد شد. بدتر آن‌که، ترودی ممکن است قیمت‌ها در فروشگاه الکترونیکیِ باب را تخفیف دهد تا کاری کند که کالاهای باب جذاب‌تر شوند. او از این طریق به آلیس حقه می‌زنَد تا برای خرید کالاهای مورد نظرش، شماره‌ی کارت اعتباری‌اش را به "باب" ارسال کند.

یک اِشکال این حمله‌ی کلاسیکِ فردی ـ در ـ میانه، آن است که ترودی باید در موقعیتی باشد که ترافیک خروجی آلیس را رهگیری کرده و ترافیک ورودی او را جعل کند. در عمل ترودی بایستی یا خط تلفن آلیس و یا خط تلفن باب را استراق سمع کند زیرا استراق سمع از فیبر بدنه‌ی اصلیِ شبکه (backbone) واقعاً دشوار است. هرچند یقیناً استراق سمع از خطوط مخابراتی امکان‌پذیر است، ولی نیاز به کار زیادی دارد و هر چند ترودی باهوش است، ولی تنبل نیز هست. به‌علاوه، روش‌های آسان‌تری هم برای گول زدنِ آلیس وجود دارند.

تقلیدِ DNS

یکی از راه‌هایی که ترودی در اختیار دارد عبارت‌است‌از کرَک کردنِ سیستم DNS و یا شاید فقط کرَک کردنِ حافظه‌ی پنهانِ DNS در ISP آلیس، و جایگزین کردنِ آدرسِ IP باب (مثلاً آدرس 36.1.2.3) با آدرسِ IP ترودی (مثلاً آدرس 42.9.9.9). این کار به حملهَی بعدیِ منجر می‌شود. در شکل ۸-۴۶ (الف) روشی نشان داده شده که احتمالاً عمل خواهد کرد. در این‌جا آلیس (۱) از DNS آدرس IP باب را درخواست می‌کند، (۲) این آدرس را به دست می‌آورَد، (۳) از باب صفحه‌ی خانگی‌اش را درخواست می‌کند، و صفحه‌ی خانگی باب را نیز به دست می‌آورَد. ترودی رکورد DNS باب را ویرایش می‌کند تا آدرس IP خودش را (به جای آدرس IP باب) در آن قرار دهد، پس به وضعیتِ نشان داده شده در شکل ۸-۴۶ (ب) می‌رسیم. در این حالت هنگامی‌که آلیس آدرس IP باب را جستجو می‌کند، آدرس IP ترودی را به دست می‌آورَد و بنابراین همه‌ی ترافیکی که برایِ باب فرستاده است، به ترودی می‌رسد. اکنون ترودی می‌تواند بدونِ آن‌که مجبور باشد به خطوط تلفن نفوذ کند، حمله‌ی فردی ـ در ـ میانه را راه‌اندازی کند. ترودی به جای نفوذ به خطوط تلفن، بایستی به خدمتگزار DNS راه پیدا کرده و یک رکورد از آن را تغییر دهد، که کار آسان‌تری است.

ترودی چگونه می‌تواند DNS را فریب دهد؟ این کار نسبتاً آسان است. اجمالاً این‌که، ترودی می‌تواند خدمتگزار DNS در ISP آلیس را با حیله وادار به ارسالِ یک پرس‌وجو جهت جستجوی آدرس



۱. آدرس IP باب را به من بده.
۲. آدرس IP باب عبارت است از (36.1.2.3) .
۳. index.html را GET کن.
۴. صفحه‌ی خانگی باب.

(الف)

۱. آدرس IP باب را به من بده.
۲. آدرس IP ترودی عبارت است از (42.9.9.9) .
۳. index.html را GET کن.
۴. صفحه‌ی خانگی جعلی باب (توسط ترودی).

(ب)

شکل ۸-۴۶ (الف) وضعیت عادی. (ب) یک حمله‌ی مبتنی بر نفوذ به خدمتگزار DNS و ایجاد تغییر در رکورد باب.

باب بکند. متأسفانه چون DNS از UDP استفاده می‌کند، خدمتگزار DNS هیچ راهِ واقعی برای کنترلِ این‌که منبع جواب چه کسی است را ندارد. ترودی می‌تواند از این ویژگی بهره‌برداری کرده و پاسخِ مورد نظر را جعل کند و بدین ترتیب یک آدرس IP غلط به حافظه‌ی پنهانِ خدمتگزار DNS، تزریق نماید. برای سادگی فرض می‌کنیم که ISP آلیس در ابتدای کار هیچ درایه‌ای به سایت وبِ باب (یعنی *bob.com*) ندارد. اگر درایه‌ای داشته باشد، ترودی می‌تواند صبر کند تا مدتِ آن منقضی شده و بعداً سعی نماید (یا آن‌که از حقه‌های دیگری استفاده کند).

ترودی با ارسال یک درخواست جستجو به ISP آلیس و طلب کردنِ آدرس IP مربوط به *bob.com* ، حمله را آغاز می‌کند. از آن‌جا که هیچ درایه‌ای به ازای این نامِ DNS وجود ندارد لذا خدمتگزارِ حافظه‌ی پنهان از خدمتگزارِ سطحِ بالا پرس‌وجویی برای دامنه‌ی *com* انجام می‌دهد تا اطلاعات را به دست آورَد. اما ترودی بر خدمتگزار *com* غلبه می‌کند و یک پاسخ دروغ با این مضمون ارسال می‌کند: "آدرس *bob.com* عبارت است از 42.9.9.9" درحالی‌که این آدرس IP مربوط به خودش است. اگر پاسخ دروغی که ترودی فرستاده، زودتر به آلیس برسد، در حافظه‌ی پنهان ذخیره خواهد شد و پاسخ واقعی پذیرفته نخواهد شد زیرا به عنوان پاسخی ناخوانده که مربوط به یک پرس‌وجوی معوقه می‌باشد، در نظر گرفته می‌شود. نیرنگ زدن به یک خدمتگزار DNS در جهت نصب یک آدرس IP دروغین، **تقلید DNS**[1] نامیده می‌شود. یک حافظه‌ی پنهان که آدرس IP ای که مانندِ این حالت، عمداً دروغ است را نگه می‌دارد، **حافظه‌ی پنهانِ مسموم**[2] نامیده می‌شود.

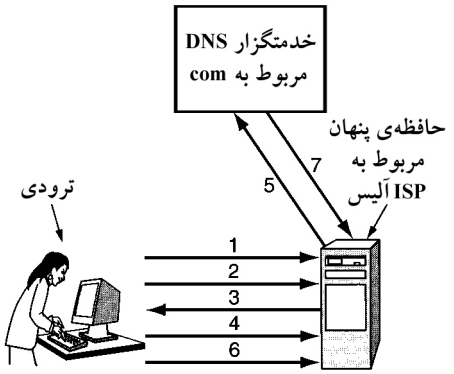
1. DNS spoofing 2. Poisoned cache

در اصل، قضایا به این سادگی‌ها نیستند. اول آن‌که ISP آلیس کنترل می‌کند که آیا پاسخ حاوی آدرس IP مبدأ صحیح از خدمتگزار سطح بالا هست یا خیر. اما چون ترودی می‌تواند هر چیزی که بخواهد را در فیلد IP قرار دهد، می‌تواند به راحتی این کنترل را خنثی سازد زیرا آدرس‌های IP خدمتگزارهای سطح بالا مجبورند عمومی باشند.

دوم آن‌که، خدمتگزارهای DNS برای آن‌که بتوانند بگویند کدام پاسخ مربوط به کدام درخواست است، تمام درخواست‌ها یک شماره‌ی توالی (sequence number) با خود حمل می‌کنند. به منظور تقلید ISP آلیس، ترودی ناچار است شماره‌ی توالی فعلی آن را بداند. آسان‌ترین راه برای ترودی جهت دانستن شماره‌ی توالی فعلی آن است که خودش یک دامنه ثبت کند، مثلاً با نام *trudy-the-intruder.com* . فرض کنید آدرس IP این دامنه نیز 42.9.9.9 باشد. ترودی همچنین یک خدمتگزار DNS برای دامنه‌ای که جدیداً درست کرده، ایجاد می‌کند: *dns.trudy-the-intruder.com*. این خدمتگزار نیز از آدرس IP ترودی یعنی 42.9.9.9 استفاده می‌کند زیرا ترودی فقط یک کامپیوتر دارد. حالا ترودی باید ISP آلیس را از خدمتگزار DNS خود باخبر سازد. انجام این کار آسان است. ترودی فقط باید از ISP آلیس درخواست *foobar.trudy-the-intruder.com* را بکند. این درخواست باعث می‌شود ISP آلیس با سوال از خدمتگزار سطح بالای *com* بفهمد که چه کسی دامنه‌ی جدید ترودی را سرویس می‌دهد.

با قرار گرفتن ایمن *dns.trudy-the-intruder.com* در حافظه‌ی پنهان در ISP آلیس، حمله‌ی واقعی می‌تواند شروع شود. اکنون ترودی پرس‌وجوی *www.trudy-the-intruder.com* را به ISP آلیس می‌دهد. طبیعتاً ISP یک پرس‌وجو به خدمتگزار DNS ترودی ارسال می‌کند تا از او سوال کند. این پرس‌وجو، آن شماره‌ی توالی‌ای که ترودی در جستجویش است را در خود دارد. ترودی فوراً از ISP آلیس می‌خواهد که باب را جستجو کند. ترودی بلافاصله با ارسال یک پاسخ جعلی به ISP، به پرسش خودش پاسخ می‌دهد البته با این ادعا که پاسخ از خدمتگزار *com* سطح بالا رسیده است، و می‌گوید: "آدرس *bob.com* عبارت است از 42.9.9.9". این پاسخ جعلی یک شماره‌ی توالی حمل می‌کند که یک واحد بالاتر از شماره‌ای است که هم‌اکنون دریافت کرده. تا زمانی‌که ترودی در این شرایط قرار دارد می‌تواند یک پاسخ جعلی دوم نیز با شماره‌ی توالی ۲ واحد بالاتر ارسال کند، و شاید یک دوجین پاسخ با شماره‌هایی که روند افزایشی دارند، ارسال نماید. یکی از آن‌ها جور درمی‌آید. بقیه دور انداخته خواهند شد. هنگامی‌که پاسخ جعلی به آلیس برسد، آن را در حافظه‌ی پنهان قرار می‌دهد؛ وقتی بعداً پاسخ واقعی برسد، مورد پذیرش قرار نمی‌گیرد زیرا هیچ پرس‌وجویی منتظرش نیست.

اکنون هنگامی‌که آلیس *bob.com* را جستجو می‌کند، به او گفته می‌شود از 42.9.9.9 استفاده کند که آدرس ترودی است. ترودی یک حمله‌ی موفق فردی ـ در ـ میانه را از اتاق نشیمن خودش اجرا کرده است. مراحل مختلف این حمله در شکل ۸–۴۷ نشان داده شده‌اند. این حمله‌ی به‌خصوص می‌تواند با وادار کردن خدمتگزارهای DNS به استفاده از ID های تصادفی در صف‌هایشان (به جای آن



۱. به دنبال foobar.trudy-the-intruder.com جستجو کن
(برای آن که به اجبار وارد حافظه‌ی پنهان ISP شود)
۲. به دنبال www.trudy-the-intruder.com جستجو کن
(برای به دست آوردن شماره‌ی توالی بعدی ISP)
۳. www.trudy-the-intruder.com را درخواست کن
(شماره‌ی توالی بعدی ISP، یعنی n را حمل می‌کند)
۴. فورا دنبال bob.com بگرد (برای آن که ISP وادار به انجام
پرس و جو از خدمتگزار com در گام ۵ گردد)
۵. پرس‌وجوی قانونی و درست از bob.com با seq = n + 1
۶. جواب جعلی تهیه شده توسط ترودی: bob در آدرس
42.9.9.9 است و seq = n + 1 می‌باشد
۷. جواب واقعی (پذیرفته نمی‌شود چون خیلی دیر رسیده)

شکل ۸–۴۷ ترودی ISP آلیس را تقلید می‌کند.

که فقط بشمرند) خنثی شود. اما به نظر می‌رسد هر زمان یک حفره را می‌گیریم، حفره‌ی دیگری باز می‌شود. مشخصاً در این مورد، ID ها فقط ۱۶ بیت هستند لذا مرور همه‌ی آن‌ها وقتی یک کامپیوتر در اختیار داشته باشیم که کارِ حدس زدن را انجام می‌دهد، آسان است.

DNS امن

مسئله‌ی اصلی آن است که DNS زمانی طراحی شده بوده که اینترنت یک امکان پژوهشی برای چند صد نفر از دانشگاهیان بود، نه آلیس، نه باب، نه ترودی به مهمانی دعوت نشده بودند. در آن زمان امنیت مورد بحث نبود؛ موضوعِ مهم فقط این بود که اینترنت کار کند. با گذشت سالیان، محیط به شدت تغییر کرده است و به همین دلیل در سال ۱۹۹۴ سازمان IETF یک گروه کاری برپا نمود تا DNS را به صورتی بنیادین، امن سازد. این پروژه (ی در حال پیشرفت) با عنوان **DNSsec** (**امنیت DNS**[1]) شناخته می‌شود؛ اولین خروجی آن در RFC 2535 ارائه شده است. متأسفانه DNSsec هنوز به شکل کامل برقرار نشده است بنابراین تعداد بسیار زیادی از خدمتگزارهای DNS هنوز هم در برابر حملات تقلید آسیب‌پذیر هستند.

به لحاظ مفهومی DNSsec بینهایت آسان است. این مفهوم مبتنی بر رمزنگاری کلید ـ عمومی می‌باشد. هر منطقه‌ی DNS ("DNS zone"، بر اساس طرحی که در شکل ۷–۵ آمده) یک جفت کلید عمومی/خصوصی دارد. تمام اطلاعاتِ ارسالی توسط خدمتگزار DNS با کلید خصوصیِ منطقه‌ی آغاز کننده[2] امضا می‌شود بنابراین دریافت کننده می‌تواند هویتِ آن را راستی‌آزمایی کند.

سه سرویس اصلی توسط DNSsec ارائه می‌شوند:

۱. اثباتِ محلی که داده از آن‌جا سرچشمه گرفته است.
۲. توزیع کلید عمومی.

1. DNS security 2. Originating zone

۳. تصدیق هویتِ تراکنش و تصدیق هویتِ درخواست.

سرویس اصلی همان سرویس اول است. این سرویس راستی‌آزمایی می‌کند که داده‌ای که برگردانده شده، از طرف جایی که منطقه به آن تعلق دارد (zone's owner)، تأیید شده باشد. دومین مورد برای ذخیره و بازیابی کلیدهای عمومی به شیوه‌ای امن، سودمند است. سومین مورد برای محافظت در برابر حمله‌های بازنواخت و تقلید، ضروری است. توجه داشته باشید که رازپوشی مورد نظر نیست زیرا تمام اطلاعاتِ DNS باید عمومی باشند. از آن‌جا که انتظار داریم مرحله‌بندی کردنِ عملیات در DNSsec چندین سال طول بکشد، لذا توانایی خدمتگزارهای آگاه ـ به ـ امنیت[1] برای همکاری کردن با خدمتگزارهای بی‌اعتنا ـ به ـ امنیت[2]، موضوع مهمی است، که دلالت بر این دارد که پروتکل نمی‌تواند تغییر کند. اکنون بیایید بعضی جزئیات را بررسی کنیم.

رکوردهای DNS درون مجموعه‌هایی به نام **RRSet** (**مجموعه‌های رکورد منبع**[3]) گروه‌بندی می‌شوند به طوری که همه‌ی رکوردهایی که دارای نام (name)، کلاس (class)، و نوع (type) یکسان هستند با هم در یک مجموعه جمع شده‌اند. برای مثال درصورتی‌که یک نام DNS به یک آدرس IP اصلی[4] و یک آدرس IP ثانویه[5] تفکیک شود، یک RRSet می‌تواند شامل چندین رکورد *A* باشد. مجموعه‌های RRSet با اضافه شدنِ چندین نوعِ رکوردِ جدید (که در زیر شرح داده شده) گسترش یافته‌اند. هر RRSet از نظر رمزنگاری، درهم‌سازی می‌شود (مثلاً با استفاده از SHA-1). حاصلِ درهم‌سازی شده، به وسیله‌ی کلید خصوصیِ منطقه امضا می‌شود (مثلاً با استفاده از RSA). واحد انتقال به مشتری‌ها عبارت است از RRSet امضا شده. با رسیدنِ یک RRSet امضا شده، مشتری می‌تواند راستی‌آزمایی کند که آیا این RRSet به وسیله‌ی کلید خصوصیِ منطقه‌ی آغازکننده امضا شده یا خیر. اگر امضا تأیید شود، داده پذیرفته می‌گردد. از آنجا که هر RRSet دربردارنده‌ی امضای مخصوص به خود است، لذا RRSet ها می‌توانند در هر جایی در حافظه‌ی پنهان ذخیره شوند، حتی در خدمتگزارهای غیرقابل اعتماد (بدونِ آن‌که از به مخاطره افتادنِ امنیت نگران باشیم).

چند نوع رکورد جدید توسط DNSsec معرفی می‌شوند. اولین مورد، رکورد *KEY* است. این رکوردها کلید خصوصی یک منطقه، کاربر، میزبان، یا هرموجودیت اصلیِ (principal) دیگر را نگهداری می‌کنند، همچنین الگوریتم رمزنگاری‌ای که جهت امضا به کار می‌رود، پروتکلی که برای انتقال مورد استفاده قرار می‌گیرد، و چند بیت دیگر در این رکوردها نگهداری می‌شوند. کلید عمومی بدون پوشش و به صورت بی‌دفاع ذخیره می‌گردد. از گواهینامه‌های X.509 استفاده نمی‌شود چون حجم زیادی دارند. فیلد الگوریتم به ازای امضاهای MD5/RSA یک عدد 1 (گزینه‌ی توصیه شده)، و به ازای سایر ترکیب‌ها، مقادیر دیگری را نگه می‌دارد. فیلد پروتکل می‌تواند استفاده از IPsec یا سایر پروتکل‌های امنیتی را (در صورت وجود) نشان دهد.

---

1. Security-aware 2. Security-ignorant 3. Resource Record Set 4. Primary IP address
5. Secondary IP address

دومین نوع جدید برای رکورد، رکورد *SIG* است. این رکورد بر اساس الگوریتمی که در رکورد *KEY* مشخص شده است، درهم‌سازیِ امضا شده را نگهداری می‌کند. این امضا به تمام رکوردها در RRSet (شامل هر رکوردِ *KEY* ای که موجود است، به غیر از خودش) اِعمال می‌گردد. همچنین زمان‌های مربوط به شروع دوره‌ی اعتبار و خاتمه‌ی آن را نیز نگهداری می‌کند. به‌علاوه‌ی نام امضا کننده و تعدادی اقلام دیگر.

طراحی DNSsec به نحوی است که یک کلید خصوصی منطقه می‌تواند offline بمانَد. یک یا دو بار در روز، محتویات پایگاه اطلاعاتیِ منطقه می‌تواند به صورت دستی (مثلاً با استفاده از CD-ROM) به یک ماشینِ جداگانه و غیرمتصل که کلید خصوصی برای آن ماشین، حُکم محلی (local) را دارد، حمل شود. همه‌ی RRSet ها می‌توانند در آن‌جا امضا شوند و سپس رکوردهای *SIG* می‌توانند بر روی CD-ROM به خدمتگزار اصلیِ منطقه برگردانده شوند. به این ترتیب کلید خصوصی می‌تواند بر روی یک CD-ROM قفل شده ذخیره شود و در امان بمانَد، غیر از مواقعی که جهت امضا کردنِ RRSet های جدیدِ روزانه، در یک ماشین جداگانه و غیرمتصل قرار داده می‌شود. بعد از آن‌که امضا کردن به پایان رسید، تمام کپی‌های کلید از حافظه پاک می‌شوند و دیسک و CD-ROM به محل امنِ خودشان برگردانده می‌شوند. این رویه امنیت الکترونیکی را به امنیتِ فیزیکی تقلیل می‌دهد، یعنی چیزی که مردم نحوه‌ی کار با آن را بلد هستند.

این روشِ پیش ـ امضای RRSet[1] ها سرعت پروسه‌ی پاسخگویی به پرس‌وجوها را خیلی بالا می‌بَرَد زیرا هیچ رمزنگاری‌ای مجبور نیست فی‌الفور انجام شود. در این‌جا مصالحه بر سرِ این است که حجم بزرگی از فضای دیسک لازم است تا تمام کلیدها و امضاها در پایگاه‌های اطلاعاتیِ DNS ذخیره شوند. بعضی رکوردها به واسطه‌ی امضاها تا ۱۰ برابر اندازه‌شان افزایش خواهد یافت.

هنگامی‌که یک پردازه‌ی مشتری یک RRSet امضا شده را بگیرد، بایستی کلید عمومیِ منطقه‌ی آغاز کننده را به آن اِعمال کند تا متن درهم‌سازی شده را رمزبَرداری کند. از سوی دیگر باید خودش نیز متن درهم‌سازی شده را محاسبه کند، و سپس این دو متن را با یکدیگر مقایسه نماید. اگر هر دو متن مثل هم باشند، در این صورت داده را معتبر در نظر می‌گیرد. اما این رویه پرسشی را مطرح می‌سازد: مشتری چگونه کلید عمومیِ منطقه را به دست می‌آورَد؟ یک راه عبارت است از به دست آوردنِ آن از یک خدمتگزار قابل اعتماد و با استفاده از یک اتصال امن (مثلاً با استفاده از IPsec).

اما در عمل فرض می‌شود که مشتری‌ها با کلیدهای عمومیِ تمام دامنه‌های سطح بالا، پیش ـ پیکربندی[2] خواهند شد. اکنون اگر آلیس بخواهد سایت وب بابَ را بازدید کند، می‌تواند RRSet مربوط به *bob.com* را از DNS درخواست نماید، که دربردارنده‌ی آدرس IP و یک رکورد *KEY* است که کلید عمومی باب را در خود دارد. این RRSet به وسیله‌ی دامنه‌ی سطح بالای *com* امضا خواهد شد، بنابراین آلیس به راحتی می‌تواند اعتبار آن را راستی‌آزمایی کند. مثالی از محتویاتی که این RRSet می‌تواند داشته باشد، در شکل ۸–۴۸ نشان داده شده است.

1. Presigning 2. Preconfigured

نام دامنه	مدت زندگی	کلاس	نوع	مقدار
Bob.com.	86400	IN	A	36.1.2.3
Bob.com.	86400	IN	KEY	3682793A7B73F731029CE2737D ...
Bob.com.	86400	IN	SIG	86947503A8B848F5272E53930C ...

شکل ۸-۴۸ یک RRSet نمونه برای *bob.com*. رکورد *KEY* کلید عمومی باب است. رکورد *SIG* متن درهم‌سازیِ شده و امضا شده‌ی خدمتگزار سطح بالای *com* از رکوردهای *A* و *KEY* است، جهت راستی‌آزمایی هویتِ آن‌ها.

اکنون آلیس که به یک کپی راستی‌آزمایی شده از کلید عمومی باب مجهز شده است، می‌تواند از خدمتگزار DNS باب (که توسط باب اجرا می‌شود)، آدرس IP مربوط به *www.bob.com* را درخواست کند. این RRSet توسط کلید خصوصی باب امضا خواهد شد، لذا آلیس می‌تواند در RRSet ای که باب باز می‌گرداند، امضا را راستی‌آزمایی نماید. اگر ترودی بتواند به طریقی یک RRset اشتباه را در حافظه‌ی پنهان تزریق نماید، آلیس می‌تواند به راحتی عدم تصدیق هویت در آن را تشخیص دهد زیرا رکورد *SIG* آن نادرست خواهد شد.

با این وجود، DNSsec یک مکانیسم رمزگونه فراهم می‌کند تا یک پاسخ را با یک پرس‌وجوی مشخص، به هم همبند سازد و به این ترتیب از نوعی تقلید که ترودی توانست انجام دهد (شکل ۸-۴۷) جلوگیری نماید. این تمهید (اختیاری) ضد تقلید، یک متن درهم‌سازی شده از پیغام پرس‌وجو (که با کلید خصوصیِ پاسخ دهنده امضا شده است) را به پاسخ اضافه می‌کند. از آنجا که ترودی کلید خصوصیِ خدمتگزار سطح بالای *com* را نمی‌داند، لذا نمی‌تواند به جای پاسخ به پرس‌وجویی که ISP آلیس به آن خدمتگزار ارسال کرده است، یک پاسخ جعلی ارسال کند. البته او می‌تواند پاسخ خود را زودتر به دست بیاورَد ولی این پاسخ مردود خواهد شد زیرا امضای موجود در پرس‌وجوی درهم‌سازی شده، نامعتبر است.

همچنین DNSsec از تعدادی نوع رکوردِ دیگر نیز حمایت می‌کند. به طور مثال، از رکورد *CERT* می‌توان برای ذخیره کردنِ گواهینامه‌ها (مثل X.509) استفاده نمود. علتِ وجود این رکورد آن است که بعضی از مردم می‌خواهند DNS را به PKI تبدیل کنند. آیا این اتفاق رخ می‌دهد یا نه؟ باید به انتظار نشست و دید. بحث‌مان درباره‌ی DNSsec را در این‌جا متوقف خواهیم کرد. برای جزئیاتِ بیشتر لطفاً به RFC 2535 مراجعه نمایید.

۸-۹-۳ SSL ــ لایه‌ی سوکت‌های اَمن

نام‌گذاریِ اَمن، شروع خوبی برای بحث امنیت است ولی مطالب بسیار بیشتری درباره‌ی امنیت وب وجود دارد. شکل ۶-۳۶ طرح یک قطعه‌ی TCP را نشان می‌دهد. اکنون چگونگی دستیابی به اتصالات امن را بررسی خواهیم کرد. هر آنچه که به امنیت مربوط باشد، راحت به دست نمی‌آید، و این مورد نیز همین‌طور است.

وقتی وب در معرض استفاده‌ی عموم قرار گرفت، ابتدا فقط برای توزیع کردنِ صفحه‌های ایستا از آن استفاده می‌شد. اما خیلی زود بعضی از شرکت‌ها ایده‌ی استفاده از وب برای تراکنش‌های مالی را مطرح کردند، از قبیل خرید کالا توسط کارت اعتباری، بانکداریِ برخط، و تجارت الکترونیکیِ کالاها. این کاربردها باعث نیاز به اتصالات امن شدند. در سال ۱۹۹۵، شرکت اتصالات Netscape[1] که در آن زمان فروشنده‌ی برترِ مرورگر بود، با معرفی یک بسته‌ی امنیتی به نام **SSL** (**لایه‌ی سوکت‌های امن**[2]) به این نیاز پاسخ داد. این نرم‌افزار و پروتکلِ آن اکنون به صورت گسترده مورد استفاده قرار دارند، به طور مثال توسط Firefox ، Safari، و اینترنت اکسپلورر. به همین دلیل ارزش دارد که بعضی جزئیاتِ آن بررسی شوند.

بسته‌ی SSL یک اتصال امن مابین دو سوکت می‌سازد که شامل موارد زیر است:

۱. مذاکره درباره‌ی پارامتر(parameter negotiation) مابین مشتری و خدمتگزار.
۲. تصدیق هویتِ خدمتگزار از سوی مشتری.
۳. اتصال سرّی.
۴. محافظت از جامعیتِ داده.

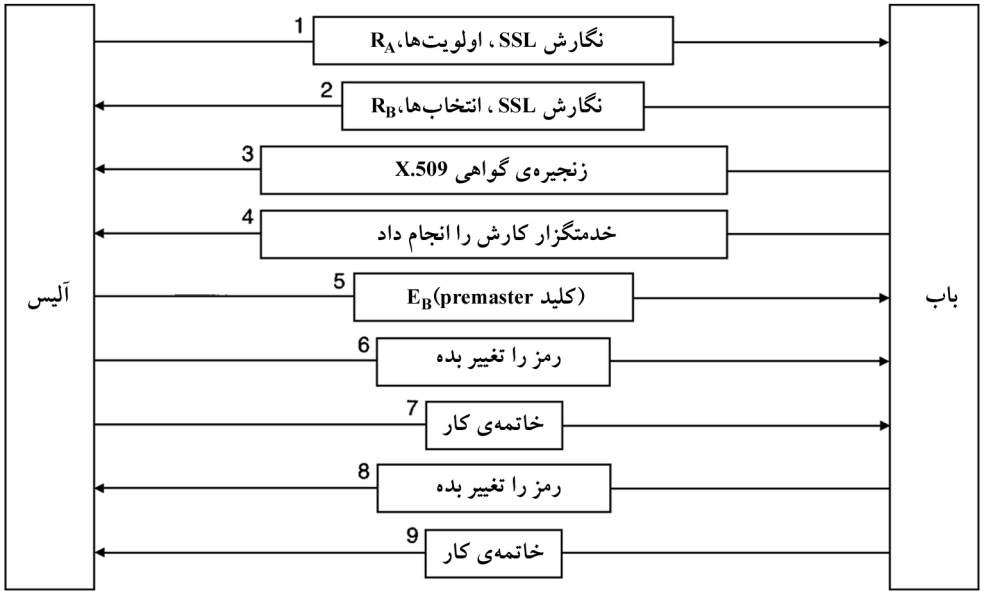
ما قبلاً این موارد را دیده‌ایم لذا نیازی نیست درباره‌ی آن‌ها وارد جزئیات شویم.

محل استقرار SSL در پُشته‌ی پروتکلِ مرسوم، در شکل ۸-۴۹ نشان داده شده است. در حقیقت SSL یک لایه‌ی جدید است که مابین لایه‌ی کاربرد و لایه‌ی حمل قرار داده شده است. این لایه، درخواست‌ها را از مرورگر پذیرفته و آن‌ها را به TCP (در لایه‌ی پایین) جهت انتقال به خدمتگزار ارسال می‌کند. با برقرار شدنِ اتصال امن، کار اصلی SSL عبارت خواهد بود از اداره کردنِ عملیات فشرده‌سازی و رمزگذاری. هنگامی که از HTTP بر فراز SSL استفاده شود، به آن **HTTPS** (**HTTP امن**[3]) گفته می‌شود، هرچند که پروتکل HTTP استاندارد است. بعضی اوقات SSL به جای پورت ۸۰، در یک پورت جدید (به شماره‌ی ۴۴۳) قابل دسترسی می‌باشد. البته SSL محدود به مرورگرهای وب نیست، اما رایج‌ترین کاربردِ آن همین است. لایه‌ی SSL تصدیق هویتِ متقابل[4] را نیز می‌تواند فراهم کند.

کاربرد (Application)(HTTP)
امنیت (Security)(SSL)
حمل (Transport)(TCP)
شبکه (Network)(IP)
پیوند داده (Data link)(PPP)
فیزیکی (Physical) (مودم، ADSL ، تلویزیون کابلی)

شکل ۸-۴۹ لایه‌ها (و پروتکل‌ها) برای یک مرورگر کاربرِ خانگی، با SSL.

1. Netscape Communications Corp. 2. Secure Sockets Layer 3. Secure HTTP
4. Mutual authentication



شکل ۸-۵۰ یک نگارشِ ساده شده از زیرـ پروتکل مربوط به برقراریِ اتصال SSL.

پروتکل SSL چندین نگارش داشته است که در این‌جا فقط نگارش ۳ را بررسی خواهیم کرد که بیشتر از بقیه مورد استفاده قرار گرفته. پروتکل SSL انواع مختلف گزینه‌ها را حمایت می‌کند. این گزینه‌ها عبارتند از انجام یا عدم انجام فشرده‌سازی، الگوریتم‌های رمزنگاریِ مورد استفاده، و بعضی موارد در ارتباط با انتقال اطلاعات راجع به محدودیت‌هایی بر روی رمزنگاری. آخرین مورد در شکل، در اصل برای اطمینان از این است که رمزنگاریِ مهم و جدّی فقط هنگامی استفاده شود که هر دو طرفِ اتصال در ایالات متحده باشند. در سایر موارد، کلیدها در حدِّ ۴۰ بیت هستند که برای رمزنگاران حُکم شوخی را دارد. شرکت Netscape وادار به لحاظ کردنِ چنین محدودیتی شده تا مجوز صدور را از دولت ایالات متحده کسب کند.

پروتکل SSL از دو زیرـ پروتکل تشکیل می‌شود، یکی برای برقراریِ یک اتصال امن و یکی برای استفاده از این اتصال. بیایید کارمان را با بررسی چگونگی برقراریِ اتصالات امن شروع کنیم. زیرـ پروتکلِ برقراریِ اتصال در شکل ۸-۵۰ نشان داده شده. این زیرـ پروتکل با پیغام 1 آغاز می‌کند، یعنی هنگامی‌که آلیس یک درخواست برای برقراری اتصال به باب ارسال می‌کند. این درخواست، نگارش SSL ای که آلیس در اختیار دارد، و همچنین اولویت‌های آلیس در ارتباط با فشرده‌سازی و الگوریتم‌های رمزنگاری را مشخص می‌کند. و نیز حاوی یک نانس است (R_A) که بعداً از آن استفاده می‌شود.

حالا نوبت باب است. در پیغام 2، باب از میان الگوریتم‌های مختلفی که آلیس می‌تواند حمایت کند، یکی را انتخاب کرده و نانس خودش (یعنی R_B) را ارسال می‌کند. باب سپس در پیغام 3 یک گواهینامه ارسال می‌کند که حاوی کلید عمومی‌اش است. اگر این گواهینامه از طرف یک مرجع

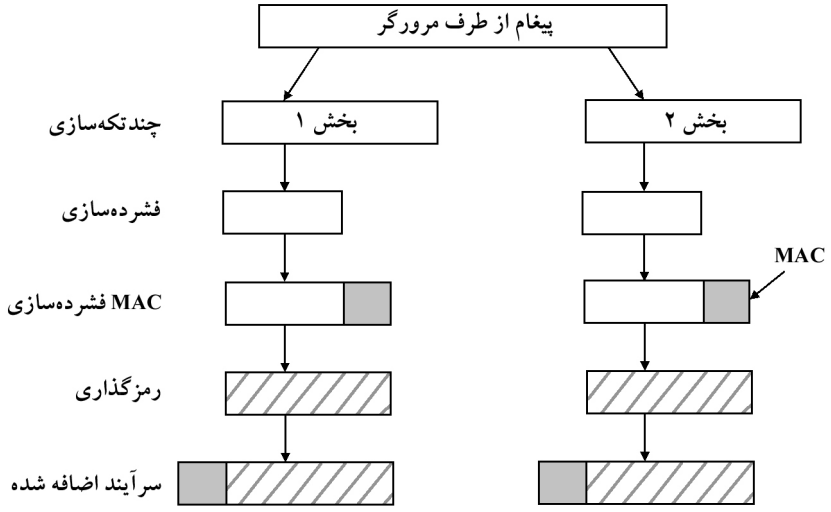
صلاحیتدارِ شناخته شده امضا نشده باشد، باب یک زنجیره از گواهینامه‌ها را نیز ارسال خواهد کرد که بتوانند یکی بعد از دیگری یکدیگر را تأیید کنند. تمام مرورگرها، از جمله مرورگرِ آلیس، از قبل با حدودِ ۱۰۰ کلید عمومی بارگذاری می‌شوند بنابراین باب می‌تواند زنجیره‌ای را برقرار کند که به یکی از این کلیدها لنگر داشته باشد. آلیس قادر به راستی‌آزماییِ کلید عمومی باب می‌باشد. در این مرحله باب ممکن است پیغام‌های دیگری را نیز ارسال کند (از قبیل یک درخواست بابت گواهینامه‌ی کلید عمومی آلیس). وقتی کار باب تمام شود، پیغام 4 را ارسال می‌کند. این پیغام به آلیس می‌گوید که حالا نوبت آلیس است.

آلیس با انتخاب یک **کلید premaster**[1] تصادفی ۳۸۴ـ بیتی و ارسالِ آن به باب، در حالی‌که با کلید عمومی باب رمزگذاری شده است، پاسخ می‌دهد (پیغام 5). کلید نشستِ اصلی که برای رمزگذاری داده به کار می‌رود، از کلید premaster استخراج می‌شود و به روشی پیچیده با هر دو تا نانس که داشتیم، ترکیب می‌شود. بعد از رسیدنِ پیغام 5، هم آلیس و هم باب قادر به محاسبه‌ی کلید نشست خواهند بود. به همین دلیل، آلیس به باب می‌گوید که به رمز جدید سوئیچ کند (پیغام 6) و همچنین به باب می‌گوید که برقراریِ زیرـ پروتکل را به اتمام رسانده است (پیغام 7). سپس باب، آلیس را ack می‌کند (پیغام 8 و پیغام 9).

با این حال، گرچه آلیس می‌داند باب کیست، ولی باب نمی‌داند آلیس کیست (مگر آن‌که آلیس یک کلید عمومی و یک گواهینامه‌ی متناظر با آن کلید داشته باشد، چیزی که برای یک شخص غیرمحتمل است). بنابراین اولین پیغام باب ممکن است درخواستِ login از آلیس باشد (با استفاده از نام و رمز عبوری که قبلاً مقرر شده است). اما پروتکلِ login خارج از محدوده‌ی SSL است. یک بار که این عمل (به هر طریقی) به انجام برسد، حمل داده می‌تواند آغاز گردد.

همان‌گونه که در بالا اشاره شد، SSL قادر به حمایت از چندین الگوریتم رمزگونه می‌باشد. قوی‌ترینِ آن‌ها، از DES سه‌گانه استفاده می‌کند همراه با سه کلید مجزا برای رمزگذاری و همراه با SHA-1 برای جامعیتِ پیغام. این ترکیب نسبتاً کُند است به همین دلیل غالباً برای بانکداری و سایر کاربردهایی که در آن‌ها به بالاترین حد از امنیت احتیاج است، استفاده می‌شود. برای کاردهای عادیِ تجارت الکترونیک از RC4 با یک کلید ۱۲۸ـ بیتی برای رمزگذاری و از MD5 برای تصدیق هویتِ پیغام، استفاده می‌شود. پروتکل RC4، کلید ۱۲۸ـ بیتی را به عنوان بذر[2] گرفته و برای مصرف داخلی‌اش، آن را به یک عدد بسیار بزرگ‌تر بسط می‌دهد. سپس از این عدد داخلی برای تولید یک جریان کلید[3] استفاده می‌کند. این جریان کلید با متن آشکار XOR می‌شود تا یک رمز جریانیِ کلاسیک را تهیه کند (همان‌طور که در شکل ۸-۱۴ دیدیم). نگارش‌های صادراتی نیز از RC4 با کلیدهای ۱۲۸ـ بیتی استفاده می‌کنند، اما ۸۸ تا از بیت‌ها عمومی می‌شوند تا شکستنِ رمز آسان گردد.

1. Premaster key 2. Seed 3. Keystream



شکل ۸-۵۱ انتقال داده با استفاده از SSL.

همانطور که در شکل ۸-۵۱ نشان داده شده برای انجام حملِ واقعی، یک زیرـ پروتکل دوم نیز مورد نیاز است. پیغام‌های رسیده از مرورگر ابتدا به واحدهایی تا 16 KB شکسته می‌شوند. اگر امکان فشرده‌سازی وجود داشته باشد، هر واحد به طور جداگانه فشرده‌سازی می‌شود. بعد از آن، یک کلید سرّی که از دو تا نانس و کلید premaster به دست آمده است، به متنِ فشرده‌سازی شده الحاق می‌شود و حاصلِ کار با استفاده از الگوریتم درهم‌سازی‌ای که در موردش توافق شده است (معمولاً الگوریتم MD5) درهم‌سازی می‌شود. این متنِ درهم‌سازی شده، به عنوان MAC (کد تصدیق هویتِ پیغام[1]) به هر تکه[2] اضافه می‌شود. سپس تکه‌ی فشرده‌سازی شده به‌علاوه‌ی MAC، با الگوریتمِ رمزگذاریِ متقارنی که در موردش توافق شده است، رمزگذاری می‌شود (که معمولاً با XOR کردنِ آن با جریانِ کلیدِ RC4 است). نهایتاً یک سرآیندِ تکه به آن الصاق شده و تکه از طریق اتصال TCP منتقل می‌گردد.

اما احتیاط شرط عقل است. از آن‌جا که نشان داده شده RC4 کلیدهای ضعیفی دارد که به راحتی می‌توانند رمزیابی شوند، امنیتِ SSL ای که از RC4 استفاده می‌کند، شکننده است (Fluhrer و همکاران، ۲۰۰۱). مرورگرهایی که به کاربران اجازه‌ی انتخاب از میان مجموعه‌ی رمز را می‌دهند، باید به نحوی پیکربندی شوند که همواره از DES سه‌گانه با کلیدهای ۱۶۸ـ بیتی و SHA-1 استفاده کنند، حتی اگر این ترکیب از RC4 و MD5 کُندتر باشد. یا حتی بهتر است کاربران وادار به ارتقای مرورگرهایشان به مرورگرهایی شوند که از آنچه بعدها به جای SSL می‌آید (که به طور خلاصه شرح خواهیم داد) حمایت کنند.

یک مشکل در ارتباط با SSL آن است که ممکن است موجودیت‌های اصلی (the principals) گواهینامه نداشته باشند، و حتی اگر هم داشته باشند همواره راستی‌آزمایی نمی‌کنند که آیا کلیدهایی که استفاده می‌شوند، با آن‌ها هماهنگ هستند یا خیر.

1. Message Authentication Code 2. Fragment

شرکت ارتباطات Netscape در سال ۱۹۹۶، SSL را به IETF برگرداند تا آن را استاندارد کند. حاصلِ کار عبارت بود از: **TLS** (**امنیت لایه‌ی حمل**[1]) که در RFC 5246 شرح داده شده.

پروتکل TLS بر روی SSL نگارش ۳ ساخته می‌شود. تغییراتی که بر SSL اِعمال شدند نسبتاً کوچکند، اما همین تغییرات هم کافیست تا SSL نگارش ۳ و TLS نتوانند با هم کار کنند. به طور مثال، روش به دست آوردنِ کلید نشست از کلید premaster و نانس‌ها تغییر کرد تا کلیدِ قوی‌تری حاصل شود (یعنی رمزیابیِ آن دشوارتر گردد). به دلیل این ناسازگاری، اغلبِ مرورگرها هر دو پروتکل را پیاده‌سازی می‌کنند به طوری که اگر ضرورت اقتضا کند، در حین مذاکره، TLS به SSL عدول کند. به این موضوع با عنوان SSL/TLS اشاره می‌شود. اولین پیاده‌سازی از TLS در سال ۱۹۹۹ ارائه شد و نگارش 1.2 در آگوست ۲۰۰۸ تعریف گردید. این نگارش شامل حمایت از مجموعه رمزهای قوی‌تر (به خصوص AES) می‌باشد. هنوز هم در بازار، SSL قدرتمند است هر چند که TLS احتمالاً رفته رفته جای آن را خواهد گرفت.

۸-۹-۴ امنیت کدِ در حال حرکت

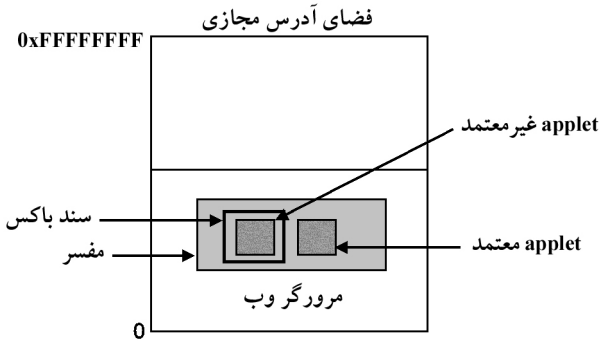
نام‌گذاری و اتصالات، دو محدوده‌ی مرتبط با امنیت وب هستند. اما موارد بیشتری هم مطرح می‌باشند. در روزهای آغازین، هنگامی‌که صفحات وب فقط فایل‌های HTML ایستا بودند، کدهای قابل اجرا در آن وجود نداشتند. اکنون این صفحات غالباً برنامه‌های کوچکی در خود دارند که شامل applet های جاوا، کنترل‌های ActiveX، و اسکریپت‌های جاوا می‌باشند. روشن است که پایین‌گذاری و اجرای چنین **کدِ در حال حرکتی**[2]، ریسک امنیتی زیادی دارد، بنابراین انواع روش‌ها برای کاهش این ریسک ابداع شده‌اند. در این‌جا نگاه سریعی خواهیم داشت به بعضی مواردی که از کدِ در حال حرکت و بعضی رویکردهای مرتبط با آن ناشی می‌شوند.

امنیت Applet جاوا

در اصل، applet های جاوا عبارتند از برنامه‌های جاوای کوچک و کامپایل شده به یک زبان ماشینِ پُشته ـ گرا[3] به نام **JVM** (**ماشین مَجازی جاوا**[4]). این برنامه‌ها می‌توانند بر روی یک صفحه‌ی وب قرار داده شوند تا همراه با آن صفحه، پایین‌گذاری شوند. همان‌طور که شکل ۸-۵۲ نشان می‌دهد بعد از بار شدنِ صفحه، applet ها در یک مفسّر JVM که درون مرورگر است ثبت می‌شوند.

مزیت اجرا کردنِ کد مفسّر بر فراز کد کامپایل شده آن است که هر یک از دستورالعمل‌ها (instruction) قبل از آن‌که اجرا شوند، توسط مفسّر بررسی می‌شوند. این کار به مفسّر فرصت می‌دهد تا کنترل کند آیا آدرسِ دستورالعمل معتبر هست یا خیر. به‌علاوه فراخوانی‌های سیستمی نیز پیدا شده و تفسیر می‌شوند. نحوه‌ی اداره شدنِ این فراخوانی‌ها، موضوع سیاست امنیت است. برای مثال، اگر یک

1. Transport Layer Security 2. Mobile code 3. Stack-oriented machine language
4. Java Virtual Machine



شکل ۸-۵۲ applet ها می‌توانند توسط یک مرورگر وب تفسیر شوند.

applet قابل اعتماد باشد (مثلاً از دیسکِ محلی آمده باشد) فراخوانی‌های سیستمیِ این applet می‌توانند بدون هیچ پرسشی به انجام برسند. اما اگر یک applet قابل اعتماد نباشد (مثلاً از طریق اینترنت رسیده باشد) می‌تواند در چیزی به نام **سَند باکس**[1] محصورسازی شود تا رفتار آن محدود شده و تلاش‌هایش جهت استفاده از منابع سیستمی به تله بیفتند.

هنگامی‌که یک applet تلاش کند از یکی از منابع سیستمی استفاده نماید، فراخوانی او (جهت تأیید و موافقت) به یک ناظرِ امنیتی فرستاده می‌شود. ناظر، این فراخوانی را با در نظر گرفتنِ سیاست امنیتیِ محلی بررسی کرده و بر این اساس تصمیم می‌گیرد که به این فراخوانی اجازه دهد و یا آن را نپذیرد. به این ترتیب می‌توان اجازه‌ی دسترسی به بعضی از منابع (ولی نه تمام منابع) را به appletها داد. متأسفانه حقیقتِ ماجرا آن است که مدل امنیت بد عمل می‌کند و خطاها همواره به چشم می‌خورند.

ActiveX

کنترل‌های ActiveX برنامه‌های دودویی x86 هستند که می‌توانند در صفحات وب، توکار شوند. هنگام مواجهه با یکی از این برنامه‌ها، کنترل می‌شود که آیا این برنامه باید اجرا شود یا خیر. اگر برنامه از این کنترل عبور کند، اجرا می‌شود. این برنامه تفسیر نمی‌شود و سندباکس هم نمی‌شود، لذا به اندازه‌ی سایر برنامه‌های کاربر توان داشته و توانِ بالقوه‌ی زیادی برای ایجاد خسارت دارد. بنابراین کلِ امنیتی که داریم، در این است که تصمیم گرفته شود آیا کنترلِ ActiveX اجرا شود یا خیر. در بازنگری مشخص می‌شود که کلِ ایده، یک حفره‌ی امنیتی بزرگ است.

روشِ انتخابی مایکروسافت جهت اتخاذ این تصمیم، بر مبنای ایده‌ی **امضا کردنِ کد**[2] است. هر کنترل ActiveX، توسط یک امضای دیجیتال همراهی می‌شود ــ یک شکل درهم شده از کد که توسط ایجاد کننده‌ی آن امضا شده است (با استفاده از رمزنگاری کلید_عمومی). هنگامی‌که یک کنترل ActiveX می‌رسد، مرورگر ابتدا امضا را راستی‌آزمایی می‌کند تا مطمئن شود در حین حمل، چیزی با آن درنیامیخته باشد. اگر امضا صحیح باشد، مرورگر در وهله‌ی بعد جداول داخلی آن را کنترل می‌کند

1. Sandbox 2. Code signing

تا ببیند آیا تولید کننده‌ی برنامه قابل اعتماد است یا خیر (یا آن‌که زنجیره‌ای وجود دارد که به یک تولید کننده‌ی قابل اعتماد می‌رسد). درصورتی‌که تولیدکننده قابل اعتماد باشد، برنامه اجرا می‌شود؛ در غیر این صورت برنامه اجرا نمی‌شود. سیستمی که مایکروسافت برای راستی‌آزماییِ کنترل‌های ActiveX دارد **Authenticode** نامیده می‌شود.

مقایسه میان رویکردهای جاوا و ActiveX سودمند است. در رویکرد جاوا هیچ تلاشی نمی‌شود تا معلوم گردد چه کسی applet را نوشته است. در عوض، یک مفسّرِ زمان اجرا باید اطمینان یابد آن دسته از کارهایی که مالک ماشین مشخص کرده appletها نمی‌توانند انجام دهند را انجام ندهد. در مقابل، در امضا کردنِ کد هیچ تلاشی نمی‌شود تا رفتار زمان اجرای کدِ در حال حرکت، پایش گردد. اگر کد از یک منبع قابل اعتماد آمده باشد و در حینِ عبور ویرایش نشده باشد، عیناً اجرا می‌شود. هیچ تلاشی نمی‌شود تا معلوم گردد آیا این کد بَدنهاد است یا خیر. اگر برنامه‌نویس اولیه‌ی آن، *قصدش این بوده* که دیسک سخت را فرمت کرده و سپس flash ROM را پاک کند تا کامپیوتر دیگر نتواند مجدداً بوت شود، و اگر گواهی شود که برنامه‌نویس معتمد است، در این صورت کد اجرا خواهد شد و کامپیوتر را به کلی تخریب خواهد کرد (مگر آن‌که کنترل‌های ActiveX در مرورگر فعال نشده باشند).

بسیاری از مردم احساس می‌کنند که اعتماد کردن به یک شرکت نرم‌افزاریِ ناشناخته، کار ترسناکی است. جهت اثبات این مسئله، یک برنامه‌نویس در سیاتل شرکتی کامپیوتری راه انداخت و گواهی اعتماد دریافت کرد (که کار آسانی است). سپس این برنامه‌نویس یک کنترلِ ActiveX نوشت که یک shutdown بی‌نقص و تمیز انجام می‌داد. او کنترلِ ActiveX اش را توزیعِ عمومی کرد. این برنامه ماشین‌های متعددی را shutdown کرد درحالی‌که این ماشین‌ها می‌توانستند دوباره بوت شوند، بنابراین هیچ ضرری پیش نیامد. او فقط سعی کرده بود این مسئله را به دنیا اطلاع دهد. واکنش رسمی به این مورد، عبارت بود از ابطال گواهینامه‌ی مربوط به این کنترلِ ActiveX به‌خصوص، و این نقطه‌ی پایانی بود بر یک پرده‌ی کوتاه از یک بحران. اما مسئله‌ای که از آن ناشی شده همچنان برقرار است، یعنی یک برنامه‌نویس بدذات می‌تواند از این موضوع بهره‌برداری کند (Garfinkel همراه با Spafford، ۲۰۰۲). از آن‌جا که هیچ راهی برای کنترل هزاران شرکت نرم‌افزاری (که احتمال دارد کدِ در حال حرکت بنویسند) وجود ندارد لذا روش امضا کردنِ کد، مانند بلایی است که منتظر است تا گریبانمان را بگیرد.

اسکریپتِ جاوا

اسکریپتِ جاوا هیچ مدل امنیتیِ رسمی‌ای ندارد، اما تاریخچه‌ای طولانی از پیاده‌سازی‌های همراه با نشتی دارد. هر فروشنده‌ای امنیت را به طریق متفاوتی اداره می‌کند. برای مثال، Netscape Navigator در نگارش ۲ از روشی مشابه مدل جاوا استفاده می‌کرد اما در نگارش ۴ از این روش دست کشیده و به مدل امضا کردنِ کد روی آورده است.

مسئله‌ی اصلی این‌است‌که وقتی به یک کد بیگانه اجازه دهیم روی کامپیوترمان اجرا شود، مانند این است که از دردسر دعوت کنیم. از دید امنیتی، مثل این است که سارقی را به منزلمان دعوت کنیم و سپس به دقت مواظبش باشیم تا از آشپزخانه به اتاق نشیمن نرود. اگر اتفاقِ غیر منتظره‌ای روی دهد و برای یک دقیقه حواستان پرت شود، اتفاقات بدی می‌توانند روی دهند. موضوع تنش‌زا در این‌جا آن است که کدِ در حال حرکت امکانِ گرافیک‌های جالب و چشمگیر و تعامل سریع را می‌دهد، و بسیاری از طراحانِ سایت‌های وب گمان می‌کنند که اهمیت این موضوع از امنیت بسیار بیشتر است، مخصوصاً وقتی کامپیوتری که در معرض خطر است مربوط به شخص دیگری غیر از خودشان باشد.

گسترش‌های مرورگر

همان‌طور که صفحات وب توسعه یافته‌اند، بازار پُررونقی هم در **گسترش‌های مرورگر**[1]، **add-on‌ها**، و **plug-in‌ها** وجود دارد. این‌ها برنامه‌هایی کامپیوتری هستند که عملکرد مرورگرهای وب را توسعه می‌دهند. اغلب اوقات plug-in ها توانایی تفسیر یا نمایش یک نوع محتوای خاص از قبیل PDF ها یا پویانمایی‌هایی از نوع Flash animation را فراهم می‌کنند. گسترش‌ها و add-on‌ها ویژگی‌های جدیدی برای مرورگر فراهم می‌کنند، از قبیل مدیریت بهترِ رمز عبور، یا روش‌هایی برای تعامل با صفحه‌ها (مثلاً امکان خرید آسان برای برخی اقلام مشخص شده).

نصب یک گسترش، add-on، یا plug-in خیلی آسان است، درست مانند دنبال کردنِ یک پیوند در مرورگر جهت نصب یک برنامه. این عمل منجر به پایین‌گذاری کد از طریق اینترنت و نصب آن در مرورگر می‌شود. همه‌ی این برنامه‌ها داخل چارچوب‌هایی نوشته می‌شوند که برحسب مرورگر، تفاوت‌هایی با هم دارند. اما می‌توان گفت همگی بخشی از ساختار مورد اعتماد مرورگر می‌شوند. به این معنا که اگر کدی که نصب شده دارای خطا باشد، کل مرورگر مشکل‌دار می‌شود.

دو مود خرابیِ واضح دیگر هم وجود دارند. اول آن‌که ممکن است برنامه رفتار بدخواهانه‌ای داشته باشد، مثلاً از طریق گردآوری اطلاعات پرسنلی و ارسال این اطلاعات به یک خدمتگزارِ راه دور. از دید خدمتگزار، کاربر این گسترش را دقیقاً به همین منظور نصب کرده است. مسئله‌ی دومِ آن است که plug-in ها به مرورگر این توانایی را می‌دهند که انواع جدید محتوا را تفسیر کنند. غالباً این محتوا خودش یک زبان برنامه‌نویسی کامل و با بیشترین توانایی‌هاست. نوع PDF و Flash مثال‌های خوبی هستند. هنگامی‌که کاربران صفحات جدیدی با محتوای PDF و Flash را بازدید می‌کند، plug-in های موجود در مرورگرهای آن‌ها، کد PDF و Flash را اجرا می‌کنند. بهتر است این کد سالم و امن باشد؛ در اغلب اوقات آسیب‌پذیری‌هایی وجود دارد که می‌توانند مورد سوءاستفاده‌ی کد قرار گیرند. با توجه به همه‌ی این دلایل، add-on‌ها و plug-in‌ها فقط بایستی بر حسب نیاز نصب شوند، و فقط هم از طریق فروشندگان قابل اعتماد.

1. Browser Extension

ویروس‌ها

ویروس‌ها شکل دیگری از کدِ در حال حرکت می‌باشند. فقط برخلاف مثال بالا اصلاً دعوتی از ویروس‌ها نمی‌شود. تفاوت میان یک ویروس و یک کدِ در حال حرکتِ عادی آن است که ویروس‌ها برای بازتولیدِ خودشان نوشته شده‌اند. هنگامی‌که یک ویروس از راه می‌رسد، خواه از یک صفحه‌ی وب، ضمیمه‌ی یک ایمیل، یا به هر طریق دیگری، معمولاً کارش را با آلوده کردنِ برنامه‌های قابل اجرا که بر روی دیسک هستند، شروع می‌کند. وقتی یکی از این برنامه‌ها اجرا شوند، کنترل به ویروس منتقل می‌شود، که ویروس هم معمولاً سعی دارد خودش را بر روی ماشین‌های دیگر تکثیر کند. به طور مثال با ایمیل کردنِ کپی‌هایی از خودش به هر کسی که در فهرست آدرس‌های ایمیلِ قربانی قرار دارد. بعضی ویروس‌ها ناحیه‌ی مربوط به بوت شدن بر روی دیسک سخت (boot sector) را آلوده می‌کنند لذا هنگامی‌که ماشین بوت می‌شود، ویروس هم اجرا می‌گردد. ویروس‌ها تبدیل به یک معضل بر روی اینترنت شده‌اند و میلیاردها دلار خسارت به بار آورده‌اند. هیچ راه‌حل قطعی‌ای وجود ندارد. احتمالاً یک نسلِ کاملاً جدید از سیستم‌های عامل که مبتنی بر میکروکرنل‌های امن باشند، به همراهِ مجموعه‌های کاربران، پردازه‌ها، و منابع که به صورتِ کاملاً درهم تنیده و نفوذناپذیر درآمده باشند، می‌تواند در این رابطه کمک کند.

۸–۱۰ مباحث اجتماعی

اینترنت و فناوری امنیت در آن، موضوعی است که مباحث اجتماعی، سیاست عمومی، و فناوری در آن دچار تلاقی می‌شوند و غالباً نیز با پیامدهای شدیدی همراه است. در زیر تنها سه حوزه را اجمالاً بررسی خواهیم کرد: حفظ حریم خصوصی، آزادی بیان، و کپی‌رایت (یا حق تألیف). نیازی به گفتن نیست که فقط در سطح می‌مانیم. به منظور مطالعات بیشتر در این زمینه به Anderson (b۲۰۰۸)، Gafinkel همراه با Spafford (۲۰۰۲)، و Schneier (۲۰۰۴) مراجعه نمایید. اینترنت نیز پُر از مطالب مرتبط است. فقط کافیست کلماتی نظیر "privacy"، "censorship"، و "copyright" را در هر موتور جستجویی تایپ کنید. همچنین سایت کتاب *http://www.pearsonhighered.com/tanenbaum* را نیز ببینید.

۸–۱۰–۱ حفظ حریم خصوصی

آیا مردم حق داشتنِ حریم خصوصی را دارند؟ پرسش خوبی است. اصلاحیه‌ی چهارم از قانون اساسی ایالات متحده، حکومت را از تفتیش منزل مردم، اسناد و مدارک آن‌ها، و اموال آن‌ها بدون دلایل معتبر، قدغن کرده و در صورت لزوم بایستی این کار با ارائه‌ی حُکمِ بازرسی انجام شود. بنابراین دست کم در ایالات متحده، بیش از ۲۰۰ سال است که این قانون وجود دارد.

آنچه در دهه‌ی اخیر تغییر کرده عبارت است از این‌که جاسوسی شهروندان برای حکومت‌ها آسان شده و در عین حال، شهروندان نیز به راحتی می‌توانند از این‌که مورد جاسوسی قرار گیرند،

جلوگیری نمایند. در قرن هجدهم حکومت برای آن‌که بتواند مدارک یک شهروند را بازرسی کند باید یک مأمور پلیسِ اسب‌سوار را به مزرعه‌ی آن شهروند می‌فرستاد تا مدارک مورد نظر را ببیند. این، کار طاقت‌فرسایی بود. امروزه با ارائه‌ی مجوز قانونی برای کنترل، شرکت‌های تلفن و فراهم‌کنندگان اینترنت به آسانی امکانِ ضبط و کنترلِ سرّی مکالمات را فراهم می‌کنند. این موضوع، زندگی را برای افراد پلیس بسیار راحت‌تر می‌کند و خطر سقوط از اسب نیز در میان نیست!

رمزنگاری همه چیز را تغییر می‌دهد. هر کسی که زحمتِ پایین‌گذاری و نصب PGP را متحمل شودو نیز از یک کلید قدرتمند استفاده کند که محافظت خوبی از آن می‌شود، می‌تواند تا اندازه‌ای اطمینان داشته باشد که هیچ کسی در عالم نمی‌تواند ایمیلش را بخواند یا اسناد و مدارکش را بازرسی کند. حکومت‌ها به خوبی از این موضوع خبر دارند و دلِ خوشی از آن ندارند. حفظ حقیقیِ حریم خصوصی به این معناست که پاییدنِ مجرمان و جنایتکاران، برای پلیس بسیار دشوار شده است، از سوی دیگر پاییدنِ روزنامه‌نگاران و مخالفان سیاسی نیز بسیار دشوار شده است. در نتیجه بعضی حکومت‌ها استفاده یا صدور رمزنگاری را محدود و یا قدغن می‌کنند. مثلاً در فرانسه تا قبل از سال ۱۹۹۹ تمام رمزنگاری‌ها ممنوع بودند به جز مواردی که دولت کلیدهای آن‌ها را داده بود.

فرانسه، تنها نبود. در آپریل ۱۹۹۳ دولت ایالات متحده قصد خود برای ساخت یک cryptoprocessor[۱] سخت‌افزاری (یا پردازشگرِ رمز سخت‌افزاری) به نام **تراشه‌ی کلیپر**[۲] را به عنوان استانداردی برای کلیه‌ی ارتباطات تحت شبکه اعلام کرد. این طور گفته شد که این تراشه محرمانگی را برای شهروندان ضمانت خواهد کرد. شایان توجه است که این تراشه به حکومت اجازه داد کل ترافیک را با استفاده از نظامی به نام **کلیدِ تضامنی**[۳]، رمزبَرداری کند، که به این ترتیب حکومت اجازه‌ی دستیابی به تمام کلیدها را پیدا کرد. اما دولت قول داد تنها در صورتی مبادرت به استراق سمع کند که یک مجوز بازرسیِ معتبر داشته باشد. نیازی به گفتن نیست که جنجال عظیمی برپا شد، طرفدارانِ حفظ حریم خصوصی کلِ این طرح را تقبیح و مجریانِ قانون آن را ستایش می‌کردند. نهایتاً دولت کوتاه آمد و از این ایده دست برداشت.

اطلاعات زیادی درباره‌ی محرمانگیِ الکترونیکی در سایت *www.eff.org* مربوط به بنیاد مرزهای الکترونیکی[۴] در دسترس می‌باشد.

---

۱. Cryptoprocessor : از ترکیب "crypto" و "processor" تشکیل شده و عبارت است از یک کامپیوتر اختصاصی جهت انجام عملیات رمزگونه (cryptographic) که بر روی یک تراشه یا یک میکروپروسسور تعبیه شده است (مترجم).

2. Clipper chip

۳. Key escrow : به این معناست که کلیدهایی که برای رمزبَرداری کردن از داده‌ی رمزگذاری شده، ضروری هستند، به عنوان وجه‌الضمان نگهداری می‌شوند تا در شرایط ویژه، یک شخص ثالثِ دارای حق دسترسی، بتواند به این کلیدها دسترسی داشته باشد (مترجم).

4. EFF: Electronic Frontier Foundation

باز ـ ایمیل کننده‌های بی‌نام

فناوری‌هایی مانند PGP، SSL، و سایر فناوری‌ها، به دو طرف امکانِ برقراری اتصالی امن و تصدیق هویت شده را می‌دهند، بدونِ آن‌که این اتصال از سوی یک شخصِ ثالث مورد پایش یا نفوذ قرار گیرد. با این حال گاهی اوقاتِ حفظ حریم خصوصی هنگامی بهتر محقق می‌شود که تصدیق هویت *نداشته باشیم*، در حقیقت با ساختِ ارتباط بی‌نام. بی‌نام بودن ممکن است برای پیغام‌های نقطه ـ به ـ نقطه، گروه‌های خبری، یا برای هر دو مورد مطلوب باشد.

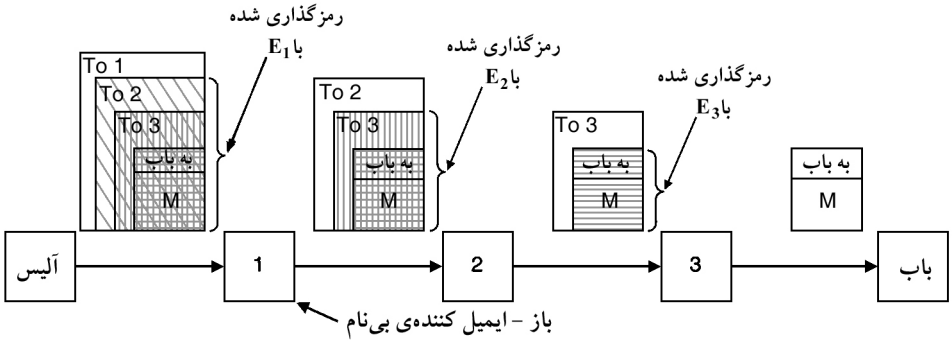
بیایید چند مثال را بررسی کنیم. اول: دگراندیشانِ سیاسی که تحت رژیم‌های استبدادی زندگی می‌کنند غالباً تمایل دارند به صورت بی‌نام یا ناشناس ارتباط برقرار کنند تا از پیامد آن بگریزند. دوم: در بیشتر مواقع اَعمال شیطنت‌آمیز در بسیاری از شرکت‌ها، مراکز آموزشی، سازمان‌های دولتی، و سایر سازمان‌ها توسط افرادی افشا می‌شود که غالباً از ترس مجازات، ترجیح می‌دهند ناشناس بمانند. سوم: افرادی که مرتبط با جوامع غیرمحبوبِ اجتماعی، سیاسی، یا مذهبی هستند، عمدتاً مایلند از طریق ایمیل یا گروه‌های خبری، و بدونِ آن‌که در معرض دید باشند، با یکدیگر ارتباط برقرار کنند. چهارم: گروه‌های خاص از قبیل افرادی که از بیماری‌های روانی رنج می‌برند، مایل به ایجاد گروه‌های مباحثه می‌باشند ولی بدونِ آن‌که شناخته شوند. مثال‌های متعددِ دیگری نیز وجود دارند.

در این‌جا یک مثال مشخص را بررسی می‌کنیم.در دهه‌ی ۱۹۹۰، تعدادی از افرادی که با یک گروه مذهبیِ نامتعارف مخالف بودند، از طریق یک **باز ـ ایمیل کننده‌ی بی‌نام**[1]، نظراتشان را در یک گروه خبریِ USENET قرار دادند. این خدمتگزار به کاربران اجازه می‌داد نام‌های مستعاری را تولید کرده و به خدمتگزار، ایمیل بفرستند. سپس این ایمیل‌ها با استفاده از نام‌های مستعار، باز ـ ایمیل[2] یا باز ـ اعلان[3] می‌شدند بنابراین هیچ کسی نمی‌توانست بگوید این پیغام‌ها واقعاً از کجا آمده‌اند. بعضی از اعلان‌ها مطالبی را افشا می‌کردند که آن گروه مذهبی ادعا می‌کرده، اسرار تجاری و اسنادِ دارای کپی‌رایت بوده‌اند. گروه مذهبی به مقامات محلی پاسخ داد که اسرار تجاری‌اش فاش شده و کپی‌رایتش مورد تعدی قرار گرفته است، که هر دوی این موارد در محلی که خدمتگزار در آن قرار داشت، جُرم محسوب می‌شد. یک دادگاه موضوع را دنبال نمود و اپراتورِ آن خدمتگزار را وادار کرد تا نگاشتی از اطلاعات مربوط به شناسه‌های حقیقیِ افرادی که اعلان‌ها را داده بودند، به دادگاه تفویض نماید.

بخش قابل توجهی از جامعه‌ی اینترنتی کاملاً از این نقضِ رازداری غضبناک گردید. نتیجه‌ای که عاید می‌شود آن است که یک باز ـ ایمیل کننده‌ی بی‌نام که یک نگاشت از آدرس‌های ایمیل واقعی و نام‌های مستعار نگه می‌دارد (این باز ـ ایمیل کننده‌ها، اینک با نام باز ـ ایمیل کننده‌ی نوعِ ۱ نامیده می‌شوند) ارزش زیادی ندارد. این مورد، افرادِ گوناگون را به سمت طراحیِ باز ـ ایمیل کننده‌های بی‌نامی که در برابر احضاریه‌های این چنینی از سوی دادگاه بتوانند مقاومت کنند، تحریک می‌کند.

---

1. Anonymous remailer 2. Remail 3. Re-post



شکل ۸-۵۳ نحوه‌ی استفاده‌ی آلیس از سه باز- ایمیل کننده جهت ارسال یک پیغام به باب.

این باز- ایمیل کننده‌های جدید که غالباً **باز- ایمیل کننده‌های cypherpunk**[1] نامیده می‌شوند، به صورت زیر عمل می‌کنند. کاربر یک پیغام ایمیل تولید کرده و سرآیندهای RFC 822 را تکمیل می‌کند (البته به جز فیلد *From:*)، آن را با کلید عمومیِ باز- ایمیل کننده رمزگذاری می‌کند، و آن را به باز- ایمیل کننده ارسال می‌کند. در آن‌جا، سرآیندهای بیرونیِ RFC 822 باز می‌شوند (strip off)، محتوا رمزبَرداری شده و پیغام باز- ایمیل می‌گردد. باز- ایمیل کننده هیچ صورت حسابی ندارد و هیچ log ای نگه نمی‌دارد بنابراین حتی اگر خدمتگزار در آینده توقیف شود، هیچ ردّی از پیغام‌هایی که از این خدمتگزار عبور کرده‌اند، باقی نمی‌مانَد.

بسیاری از کاربران که مایلند ناشناس بمانند، همان‌طور که در شکل ۸-۵۳ نشان داده شده از طریق باز- ایمیل کننده‌های متعدد، اقدام به ساختِ زنجیره‌ای از درخواست‌هایشان می‌کنند. در این شکل، آلیس می‌خواهد یک کارت تبریکِ واقعاً، واقعاً، واقعاً بی‌نام به باب ارسال کند لذا از سه عدد باز- ایمیل کننده استفاده می‌کند. آلیس پیغام M را ایجاد می‌کند و یک سرآیند در آن قرار می‌دهد که شامل آدرس ایمیل باب است. سپس کلِ آن را با کلید عمومیِ باز- ایمیل کننده‌ی شماره‌ی 3 (یعنی E_3) رمزگذاری می‌کند (در شکل با هاشورهای افقی مشخص شده است). آلیس به ابتدای این مجموعه، یک سرآیند با آدرس ایمیلِ باز- ایمیل کننده‌ی شماره‌ی 3 در متنِ آشکارِ آن، اضافه می‌کند. این همان پیغامی است که در شکل، مابین باز- ایمیل کننده‌های شماره‌ی 2 و شماره‌ی 3 نشان داده شده است.

سپس آلیس این پیغام را با کلید عمومیِ باز- ایمیل کننده‌ی شماره‌ی 2 (یعنی E_2) رمزگذاری می‌کند (در شکل با هاشورهای عمودی مشخص شده است) و یک سرآیند به صورت متن آشکار که شامل آدرس ایمیلِ باز- ایمیل کننده‌ی شماره‌ی 2 است به ابتدای آن اضافه می‌کند. در شکل ۸-۵۳، این پیغام مابین پیغام‌های 1 و 2 نشان داده شده است. سرانجام، آلیس کلِ پیغام را با کلید عمومیِ باز- ایمیل کننده‌ی شماره‌ی 1 (یعنی E_1) رمزگذاری و یک سرآیند به صورت متن آشکار با آدرس ایمیلِ

1. Cypherpunk remailer

بازـ ایمیل کننده‌ی شماره‌ی 1 به ابتدای آن اضافه می‌کند. این همان پیغامی است که در سمت راست آلیس در شکل نشان داده شده و همان پیغامی است که آلیس واقعاً ارسال می‌کند.

هنگامی‌که پیغام به بازـ ایمیل کننده‌ی شماره‌ی 1 می‌رسد، سرآیند بیرونی باز می‌شود. بدنه رمزبَرداری می‌شود و سپس به بازـ ایمیل کننده‌ی شماره‌ی 2 ایمیل می‌گردد. مراحل مشابهی در دو باز–ایمیل کننده‌ی دیگر اتفاق می‌افتد.

گرچه ردیابیِ پیغام نهایی تا رسیدن به آلیس بینهایت دشوار است، باز هم بسیاری از بازـ ایمیل کننده‌ها اقدامات احتیاطیِ بیشتری نیز به جهت امنیتی انجام می‌دهند. برای مثال، ممکن است پیغام‌ها را برای یک مدت زمانِ تصادفی نگه دارند، اطلاعات بی‌ارزشی را به انتهای پیغام اضافه یا کم کنند، و پیغام‌ها را مجدداً مرتب‌سازی کنند، و همه‌ی این کارها برای آن‌است‌که تشخیصِ این‌که کدام خروجی در یک بازـ ایمیل کننده متناظر با کدام ورودی می‌باشد، هر چه دشوارتر گردد و تلاش برای تحلیلِ ترافیک خنثی شود. برای اطلاعاتی درباره‌ی این نوع بازـ ایمیل کننده‌ها، Maziéres و Kaashoek (۱۹۹۸) را مطالعه نماید.

بی‌نامی مختص ایمیل نیست. سرویس‌هایی وجود دارند که اجازه‌ی گشت‌وگذار بی‌نام در وب را با استفاده از همان شکلی که در مسیرِ لایه‌بندی شده داشتیم، می‌دهند یعنی هر گره فقط گره بعدی در زنجیره را می‌شناسد. این روش **مسیریابی پیازی**[1] نامیده می‌شود زیرا هر گره برای آن‌که معلوم کند بسته را به کجا باید پیش برانَد، پوسته‌ی لایه‌ی بعدی را برمی‌دارد. کاربر برای آن‌که از سرویسِ بی‌نام‌ساز[2] به عنوان وکیل (proxy) استفاده کند، مرورگرش را پیکربندی می‌کند. یک مثال خوب از چنین سیستمی، سیستم Tor است (Dingledine و همکاران، ۲۰۰۴). از این پس تمام درخواست‌های HTTP از طریق شبکه‌ی بی‌نام‌ساز رد می‌شوند، که صفحه را درخواست می‌کند و آن را پس می‌فرستد. سایت وب، کاربر را نمی‌بیند بلکه یک گره خروجیِ شبکه‌ی بی‌نام‌ساز را به عنوان منبع درخواست مشاهده می‌کند. چون شبکه‌ی بی‌نام‌ساز هیچ رویدادی را نگه نمی‌دارد، بعد از انجام عمل، هیچ کسی نمی‌تواند معلوم کند چه کسی درخواست کننده‌ی چه صفحه‌ای بوده است.

۸–۱۰–۲ آزادی بیان

حفظ حریم خصوصی مربوط به افرادی است که می‌خواهند آنچه سایر مردم می‌توانند از آن‌ها بدانند را محدود سازند. دومین مفهوم کلیدی در مباحث اجتماعی، آزادی بیان است. ممکن است سایت‌های وبی ایجاد شوند که مطالب ممنوعه‌ای شامل موارد زیر داشته باشند:

۱. مطالب نامناسب برای کودکان و نوجوانان.
۲. خصومت نسبت به قومیت‌ها، مذاهب، یا سایر گروه‌ها.

1. Onion routing 2. Anonymizer

۳. نقل قول‌هایی در ارتباط با وقایع تاریخی، که با تفسیر حکومت رسمی در تناقض باشد.
۴. راهنمایی‌هایی در ارتباط با دستکاری و باز کردنِ قفل‌ها، ساخت اشیای ممنوعه، رمزگذاری پیغام‌ها، و امثال این‌ها.

واکنش متداول عبارت است از ممنوع کردنِ سایت‌های "بَد".

بعضی اوقات نتایج غیرمنتظره هستند. برای مثال بعضی از کتابخانه‌های عمومی فیلترهایی در ارتباط با وب بر روی کامپیوترهایشان نصب کرده‌اند تا با مسدود کردنِ سایت‌های بَد، کامپیوترها را برای کودکان مناسب سازند. این فیلترها نه تنها سایت‌هایی که در لیستِ سیاهِ آن‌ها قرار دارند را منع (وتو) می‌کنند، بلکه قبل از نمایش صفحه‌ها، آن‌ها را از بابت کلمات نامناسب کنترل می‌نمایند.

این‌که WWW تارجهان‌گستر است، از ذهن بسیاری از مردم بیرون رفته. وب، تمام دنیا را پوشش می‌دهد. این طور نیست که همه‌ی کشورها درباره‌ی آنچه باید در وب مُجاز باشد، توافق داشته باشند. برای مثال در نوامبر سال ۲۰۰۰ یک دادگاه فرانسوی به شرکت یاهو که یک شرکت کالیفرنیایی است، دستور داد مشاهده‌ی حراج اشیای نازی‌ها بر روی سایت یاهو را برای کاربران فرانسوی مسدود نماید زیرا در اختیار داشتنِ چنین مطالبی، مغایر قانون فرانسه است. شرکت یاهو به دادگاهی در ایالات متحده درخواست استیناف داد و دادگاه نیز حق را به یاهو داد. اما این‌که قانون کدام کشور باید اجرا شود هنوز موضوع نامشخصی است.

فکرش را بکنید. چه می‌شد اگر دادگاهی در یوتا به فرانسه دستور می‌داد بعضی از سایت‌هایش را مسدود کند، چرا که با قوانین صریح یوتا مغایرت دارند؟ آیا می‌توان قوانین مذهبیِ کشور دیگری را به سوئد اِعمال نمود؟ آیا عربستان سعودی می‌تواند سایت‌های مرتبط با حقوق زنان را مسدود کند؟ تمام این مباحث حقیقتاً مصداق جعبه‌ی پاندورا[۱] هستند!

در این رابطه تفسیری از جان گیلمور[۲] وجود دارد: "شبکه، محدودیت اطلاعات را به عنوان یک خرابی تلقی کرده و آن را دور می‌زند." به عنوان یک پیاده‌سازیِ عینی، **سرویس اترنیتی**[۳] (Anderson، ۱۹۹۶) را در نظر بگیرید. هدف این روش عبارت‌است‌از اطمینان از این‌که نتوان مانع نشرِ اطلاعاتِ منتشر شده گردید و یا نتوان آن‌ها را بازنویسی و ویرایش نمود. به منظور استفاده از سرویس اترنیتی، کاربر مشخص می‌سازد که مطالب برای چه مدتی باید نگهداری شوند؟ کاربر یک فیِ هزینه که متناسب با مدت زمان نگهداری و اندازه‌ی مطلب است، پرداخت می‌کند و مطالبش را بالاگذاری می‌کند.از آن پس هیچ کس، حتی شخصی که بالاگذاری را انجام داده، نمی‌تواند این اطلاعات را حذف یا ویرایش کند.

---

۱. Pandora's box : بر طبق افسانه‌های یونانی جعبه‌ای است مملو از بلایا و شوربختی‌های ناشناخته برای بشر (مترجم).
2. John Gilmore
۳. Eternity service : یک حافظه‌ی ایمن و بسیار اتکاپذیر برای داده‌های با درجه‌ی اهمیتِ بالا می‌باشد. این سرویس مبتنی بر ساخت رسانه‌ی حافظه با ویژگی‌های مشابه با کانال‌های ارتباطی در اینترنت است، به طوری که در برابر حمله‌های DoS مصون باشد. اساس ایده بر مبنای استفاده از روش‌های افزونگی و پراکنده‌سازی (scattering) جهت تکرارسازیِ داده میان یک مجموعه‌ی بزرگ از ماشین‌ها می‌باشد، یعنی چیزی مشابه اینترنت (مترجم).

چنین سرویسی چگونه می‌تواند پیاده‌سازی شود؟ ساده‌ترین مدل عبارت‌استاز استفاده از یک سیستم همتا ـ به ـ همتا به طوری که اسنادِ ذخیره شده، بر روی خدمتگزارهای متعددِ شرکت کننده در این مدل، قرار داده شوند. هر کدام از این خدمتگزارها بخشی از حق‌الزحمه را دریافت می‌کنند که همین امر انگیزه‌ای برای پیوستن به این سیستم است. به منظور دستیابی به بیشترین حد از انعطاف‌پذیری، خدمتگزارها بایستی در سطح تعداد زیادی قلمروهای قانونی گسترده شوند. فهرست‌هایی از ۱۰ خدمتگزار که به صورت تصادفی انتخاب شده‌اند، به صورتی امن در چندین محل ذخیره خواهند شد بنابراین اگر بعضی از آن‌ها در خطر کشف رمز باشند، بقیه هنوز برقرار خواهند بود. یک اسم رمزِ تأیید شده که برای فرآیند خراب کردنِ اسناد تعیین شده است، هرگز نمی‌تواند مطمئن باشد که تمام کپی‌ها را پیدا کرده است. ضمناً سیستمَ در شرایطی که متوجه شود بعضی کپی‌ها تخریب شده‌اند، قادر به انجام عملیات خودـ اصلاحی[1] می‌باشد به طوری که سایت‌های باقیمانده تلاش خواهند کرد منابع جدیدی برای جایگزین کردن پیدا کنند.

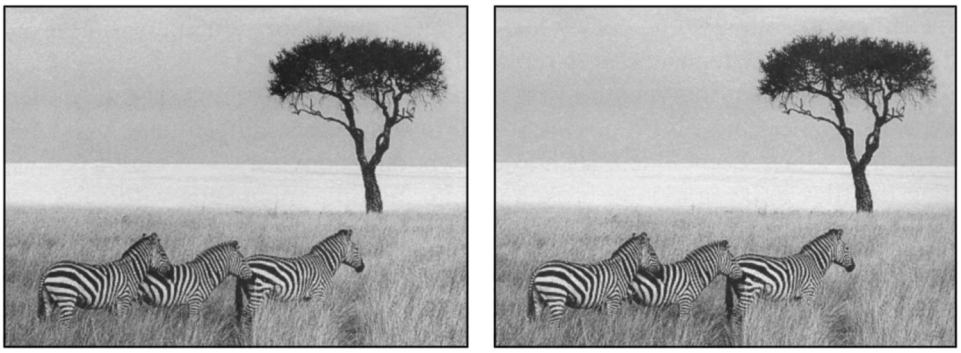
سرویس اترنیتی اولین پیشنهاد در رابطه با یک سیستم مقاوم بود. از آن زمان تاکنون موارد دیگری نیز پیشنهاد شده و بعضی از آنها پیاده‌سازی نیز شده‌اند. انواع ویژگی‌های جدید نیز اضافه شده‌اند از قبیل رمزگذاری، بی‌نامی، و تحمل خرابی[2]. غالباً فایل‌های ذخیره شده به چندین تکه شکسته می‌شوند بطوریکه هر تکه بر روی تعداد زیادی خدمتگزار ذخیره می‌شود. بعضی از این سیستم‌ها عبارتند از Freenet (Clarke و همکاران، ۲۰۰۲)، PASIS (Wylie و همکاران، ۲۰۰۰)، و Publius (Waldman و همکاران، ۲۰۰۰). کارهای دیگری هم توسط Serjantov (۲۰۰۲) گزارش شده است.

کشورهای متعددی بطور روزافزون تلاش می‌کنند تا صادرات ناملموس (که عمدتاً شامل سایت‌های وب، نرم‌افزار، مقاله‌های علمی، ایمیل، خدمات پشتیبانی، و سایر موارد می‌باشد) را سامان داده و تحت نظارت درآورند.حتی بریتانیا که سابقهٔ قرن‌ها آزادی بیان را دارد، هم‌اینک بصورت جدّی قوانین بسیار محدود کننده‌ای را در دست بررسی دارد که بطور مثال مباحث فنی مابین یک پروفسور بریتانیایی و دانشجوی خارجیِ دورهٔ Ph.D او که هر دو هم در دانشگاه کمبریج مستقر هستند را تعریف خواهد کرد زیرا صادراتِقاعده‌مند، نیازمند مجوزِ دولتی است (Anderson، ۲۰۰۲). نیازی به گفتن نیست که بسیاری از مردم چنین سیاستی را ناعادلانَه می‌دانند.

نهان‌نگاری

در نقاطی که محدودیت وجود دارد، عده‌ای سعی می‌کنند از فناوری جهت مقابله استفاده کنند. رمزنگاری اجازه‌ی ارسال پیغام‌های سرّی را می‌دهد (هر چند ممکن است این کار مُجاز نباشد). بازـ ایمیل کننده‌های بی‌نام می‌توانند کمک کنند اما چنانچه این بازـ ایمیل کننده‌ها در آن نقطه قدغن باشند و ارسال پیغام به خارج نیز نیازمند مجوز باشد، این ابزار هم کمک چندانی نمی‌توانند بکنند. اما وب می‌تواند.

1. Self-repairing 2. Fault-tolerance



(الف) (ب)

شکل ۸-۵۴ (الف) سه گورخر و یک درخت. **(ب)** سه گورخر، یک درخت، و متن کامل پنج نمایشنامه از ویلیام شکسپیر.

کسانی که می‌خواهند به صورت سرّی ارتباط داشته باشند غالباً سعی می‌کنند این حقیقت که هرگونه ارتباطی با هم داشته‌اند را مخفی کنند. علمِ اختفای پیغام‌ها **نهان‌نگاری**[1] نامیده می‌شود. این واژه، یونانی عبارتِ "نوشتنِ مخفیانه" ("covered writing") می‌باشد. در حقیقت، این اصطلاح در بین یونانیان قدیم استفاده می‌شد. هرودوت[2] درباره‌ی ژنرالی نوشته است که موی سر پیک‌ها را می‌تراشید، پیغام را بر روی پوست سرشان خالکوبی می‌کرد، و پس از آن‌که موهایشان مجدداً می‌رویید آن‌ها را روانه‌ی مأموریت می‌نمود. روش‌های امروزی نیز به لحاظ مفهومی مشابه همان روش هستند، فقط پهنای باند بالاتر و تأخیر کمتری دارند و نیازی هم به آرایشگر ندارند!

به عنوان یک نمونه، شکل ۸-۵۴ (الف) را در نظر بگیرید. این عکس توسط یکی از مؤلفان این کتاب (تاننبوم) در کنیا تهیه شده است و سه گورخر را غرق تماشای یک درخت اقاقیا نشان می‌دهد. شکل ۸-۵۴ (ب) به نظر می‌رسد که همان عکس باشد، اما یک مورد جالب در خود دارد. این شکل، متن کامل و خلاصه نشده‌ی ۵ نمایشنامه از شکسپیر[3] را در خود دارد: *هملت*[4]، *شاه لیر*[5]، *مکبث*[6]، *تاجر ونیزی*[7]، و *ژولیوس سزار*[8]. این نمایشنامه‌ها روی هم رفته بیش از 700 KB متن می‌باشند.

این طریقه‌ی نهان‌نگاری چگونه کار می‌کند؟ تصویر رنگی اولیه ۷۶۸ × ۱۰۲۴ پیکسل است. هر پیکسل سه عدد ۸ ـ بیتی دارد، هر عدد برای هر کدام از رنگ‌های قرمز، سبز، و آبی. رنگ یک پیکسل، از برهم‌نهیِ[9] خطیِ این سه رنگ به دست می‌آید. روش کدگذاری به صورت نهان‌نگاری، از سه بیتِ با ارزش کمترِ در هر رنگ RGB به عنوان یک کانال مخفی استفاده می‌کند. بنابراین هر پیکسل جا برای ۳ بیت اطلاعات سرّی دارد: ۱ بیت در مقدار مربوط به قرمز، ۱ بیت در مقدار مربوط به سبز، و ۱ بیت در مقدار مربوط به آبی. برای تصویری در این اندازه، یعنی ۷۶۸ × ۱۰۲۴ پیکسل، تا ۳ × ۷۶۸ × ۱۰۲۴ بیت یعنی برابر با ۲۹۴,۹۱۲ بایت از اطلاعات سرّی می‌تواند در تصویر ذخیره شود.

1. Steganography 2. Herodotus 3. Shakespear 4. Hamlet 5. King Lear 6. Macbeth
7. The Merchant of Venice 8. Julius Caesar 9. Superposition

متن کامل پنج نمایشنامه و یک اطلاعیه‌ی کوتاه در مجموع به ۷۳۴,۸۹۱ بایت می‌رسد. این متن ابتدا با استفاده از الگوریتم فشرده‌سازیِ استاندارد، به تقریباً 274 KB رسید. سپس خروجیِ فشرده شده با استفاده از IDEA رمزگذاری شده و در بیت‌های با ارزش پایین از هر رنگ قرار داده شد. همان‌طور که دیده می‌شود (یا در واقع، دیده نمی‌شود) وجود اطلاعات کاملاً غیرقابل تشخیص است. این موضوع در عکس بزرگ و تمام ـ رنگی نیز کاملاً غیرقابل تشخیص است. چشم قادر نیست به آسانی رنگ ۲۱ـ بیتی را از رنگ ۲۴ـ بیتی تمایز دهد.

مشاهده‌ی این دو تصویر به صورت سیاه و سفید و با دانه‌بندیِ پایین، به درستی میزان قدرتمندی این روش را نشان نمی‌دهد. برای آن‌که درک بهتری از نحوه‌ی عملِ نهان‌نگاری داشته باشید، نمایشی تدارک دیده‌ایم که شامل یک تصویر تمام ـ رنگی و با دانه‌بندی بالا از شکل ۸-۵۴ (ب) می‌باشد، در حالی‌که ۵ نمایشنامه‌ی مورد نظر هم در آن قرار داده شده‌اند. این نمایش شامل ابزاری برای درج و استخراج متن درون تصاویر می‌باشد و در سایت وبِ مربوط به این کتاب قرار دارد.

مخالفان می‌توانند به منظور استفاده از نهان‌نگاری برای برقراری ارتباطِ کشف نشدنی، یک سایت وب ایجاد کنند که مملو از تصاویری باشد که قابل قبول باشند از قبیل عکس‌های افراد مهم جامعه، ورزش‌های محلی، بازیگران سینما و تلویزیون، و امثال این‌ها. البته این عکس‌ها به سبب پیغام‌های نهان‌نگاری به صورت پازل درخواهند آمد. اگر پیغام‌ها ابتدا فشرده‌سازی و سپس رمزگذاری شوند، در آن صورت حتی برای کسانی که به وجود پیغام‌ها مشکوک شده باشند بینهایت دشوار است که پیغام‌ها را از نویز سفید[1] تشخیص دهند. البته تصاویر بایستی نو و دست اول باشند؛ کپی کردنِ یک عکس از اینترنت و تبدیل بعضی بیت‌های آن، به راستی یک سوء استفاده است.

تصاویر به هیچ وجه تنها حامل برای پیغام‌های نهان‌نگاری نیستند. فایل‌های صوتی نیز به خوبی عمل می‌کنند. اطلاعاتِ نهان می‌توانند در یک تماس صوت ـ روی ـ IP از طریق دستکاری در تأخیرهای بسته که صوت را نامفهوم می‌کند، یا حتی دستکاری در فیلدهای سرآیندِ بسته‌ها، حمل شوند (Lubacz و همکاران، ۲۰۱۰). حتی چیدمان و ترتیب برچسب‌ها در یک فایل HTML می‌تواند اطلاعات حمل کند.

گرچه نهان‌نگاری را تحت عنوان اطلاعاتِ بدون محدودیت بررسی کردیم، ولی کاربردهای متعددِ دیگری هم دارد. یک کاربرد رایج، مربوط به صاحبان تصاویر است تا پیغام‌های سرّی را در آن‌ها کد کرده و به این ترتیب، حقوق مالکیت را تصریح نمایند. اگر چنین تصویری مورد سرقت قرار گرفته و بر روی یک سایت وب قرار گیرد، مالک قانونیِ آن می‌تواند جهت اثباتِ مالکیتش، پیغامِ نهان‌نگاری شده را به دادگاه افشا کند. این شیوه را **چاپ سفید در متن کاغذ سفید**[2] می‌نامند و در Piva و همکاران (۲۰۰۲) تشریح شده است.

برای اطلاعات بیشتر راجع به نهان‌نگاری به Wayner (۲۰۰۸) مراجعه نمایید.

1. White noise 2. Watermarking

۸-۱۰-۳ کپی رایت

حفظ حریم خصوصی و محدودیت اطلاعات فقط دو مثالی هستند از جایی که فناوری به رویارویی با سیاست عمومی می‌رود. یک مورد سوم هم وجود دارد: قانون کپی‌رایت. **کپی‌رایت**[1] به پدیدآوردندگانِ **IP** (**حقوق معنوی**[2]) اعطا می‌گردد. پدیدآورندگان شامل نویسندگان، شاعران، هنرمندان، آهنگسازان، موسیقیدانان، عکاسان، فیلم‌برداران، صحنه‌آراها، و موارد دیگر می‌شوند. کپی‌رایت عبارت‌است‌از حق انحصاری در بهره‌برداری از IP پدیدآورندگان برای یک دوره‌ی زمانی (که معمولاً مدت عمر مؤلف است به‌علاوه‌ی ۵۰ سال، و یا در شرایطی که مالکیت شرکت در بین باشد، به‌علاوه‌ی ۷۵ سال). بعد از انقضای مدت کپی‌رایتِ یک اثر، آن اثر وارد میدان عمومی شده و همه می‌توانند از آن استفاده کرده و یا آن را به فروش برسانند. به طور مثال، پروژه‌ی گوتنبرگ[3] (*www.promo.net/pg*) هزاران اثر که در دامنه‌ی عمومی هستند را بر روی وب قرار داده است (از قبیل آثار شکسپیر، مارک تواین، و چارلز دیکنز). در سال ۱۹۹۸ کنگره‌ی ایالات متحده بر اساس درخواست هالیوود، کپی‌رایت در ایالات متحده را ۲۰ سال افزایش داد زیرا هالیوود ادعا می‌کرد بدونِ این کار دیگر هیچ کس اثری تولید نخواهد کرد. به عنوان مقایسه، اختراعاتِ ثبت شده فقط ۲۰ سال دوام دارند و افراد همچنان اختراعاتشان را انجام می‌دهند.

کپی‌رایت هنگامی به ردیف جلو آمد که شرکت Nabsterکه یک سرویس اشتراک موسیقی بود صاحب ۵۰ میلیون عضو شد. هرچند Nabster واقعاً هیچ موسیقی‌ای را کپی نمی‌کرد اما دادگاه رأی داد که نگهداری از یک پایگاه داده‌ی مرکزی از این‌که چه کسی کدام ترانه و آوا را دارد، موجب تخلف شده یعنی Nabster به تخلفِ سایر افراد کمک کرده است. هرچند هیچ کس صراحتاً ادعا نمی‌کند که کپی‌رایت ایده‌ی بدی است، نسل بعدی اشتراک موسیقی هم‌اکنون سبب بروز مباحث مهم اخلاقی شده است.

برای مثال، یک شبکه‌ی همتا ـ به ـ همتا را در نظر بگیرید که مردم در حال به اشتراک گذاشتنِ فایل‌های قانونی هستند (از قبیل موسیقی مربوط به دامنه‌ی عمومی، ویدیوهای خانگی، رساله‌های مذهبی که اسرار را مورد دادوستد قرار نمی‌دهند، و امثال این‌ها) و احتمالاً تعداد کمی فایل که دارای کپی‌رایت هستند. فرض کنید که همه همیشه از طریق ADSL یا کابل، برخط می‌شوند. هر ماشین یک شاخص (ایندکس) از محتویات دیسک سخت دارد، به‌علاوه‌ی یک فهرست از سایر اعضا. کسی که به دنبال یک موضوع مشخص است می‌تواند یک عضو را به طور تصادفی انتخاب کرده و ببیند آیا موضوعِ مورد نظرش، نزد او هست یا خیر. اگر موضوعی که به دنبالش است نزد او نباشد، می‌تواند

---

1. Copyright

۲. Intellectual Property : این اصطلاح با عنوان "مالکیت ناملموس" نیز شناخته می‌شود (مترجم).

۳. Gutenberg Project : یک کوشش داوطلبانه برای دیجیتالی کردن و بایگانی کارهای فرهنگی به منظور تشویق تولید و پخش کتاب‌های دیجیتالی است. این پروژه در سال ۱۹۷۱ بنیان‌گذاری شد و امروزه قدیمی‌ترین **کتابخانه‌ی دیجیتال** و دارای گنجینه‌ای منحصر به فرد از کتاب‌های الکترونیکی رایگان به شمار می‌رود (مترجم).

تمام اعضایی که در فهرست آن شخص هستند را جستجو کند، سپس تمام اعضایی که در فهرست این اعضا هستند را جستجو کند، و همین طور تا آخر. کامپیوترها این قبیل کارها را خیلی خوب انجام می‌دهند. پس از پیدا کردنِ موضوع مورد نظر، درخواست کننده فقط آن را کپی می‌کند.

اگر آن اثر دارای کپی‌رایت باشد، ممکن است درخواست کننده یک متخلف باشد (هر چند در مورد انتقال‌های بین‌المللی، قوانینی که اِعمال می‌شوند به موضوع بستگی دارند زیرا در بعضی کشورها بالاگذاری غیرقانونی است ولی پایین‌گذاری غیرقانونی نیست). اما در مورد تهیه‌کننده چطور؟ آیا نگه داشتنِ موسیقی‌ای که هزینه‌اش را پرداخته‌اید و به طور قانونی آن را بر روی دیسک سخت کامپیوترتان پایین‌گذاری کرده‌اید، یعنی جایی که دیگران می‌توانند آن را پیدا کنند، جرم است؟ اگر شما اطاقکی در یک کشور دارید که آن را قفل نمی‌کنید و یک دزدِ حقوق معنوی (دزدِ IP) که یک نوت‌بوک و یک پویشگر با خودش دارد، یک کتابِ دارای کپی‌رایت که روی دیسک سختِ نوت بوک شما است را پویش کرده و با خودش ببرد، آیا *شما* به جرم عدم محافظت از کپی‌رایتِ یک شخص دیگر، محکوم می‌شوید؟

لیکن دردسرهای بیشتری بر سر کپی‌رایت وجود دارد. هم‌اکنون نبرد عظیمی میان هالیوود و صنعت کامپیوتر در جریان است. اولی محافظت دقیق از حقوق معنوی را، آن هم به طور کامل، خواستار است اما دومی مایل نیست که برای هالیوود نقش پلیس را داشته باشد. در اکتبر سال ۱۹۹۸ کنگره‌ای با موضوع **DMCA** برگزار شد (**قانون کپی‌رایتِ هزاره در موضوعات دیجیتالی**[1]) که مقرر نمود سرپیچی کردن از هر گونه مکانیزمِ حفاظتی که در ارائه‌ی یک اثرِ دارای کپی‌رایت به کار می‌رود، و همچنین یاد دادنِ نحوه‌ی این گونه سرپیچی‌ها به دیگران، جرم تلقی می‌شود. قوانین مشابهی نیز در اتحادیه‌ی اروپا وضع گردیده است. هرچند مطمئناً هیچ کس عقیده ندارد که به سارقان ادبی در شرق دور باید اجازه‌ی کپی کردن از آثار ادبی داده شود، ولی بسیاری از مردم نیز فکر می‌کنند DMCA کاملاً تعادل میان منافع صاحبان کپی‌رایت و منافع عمومی را بر هم زده است.

یک مورد جالب: در سپتامبر سال ۲۰۰۰، یک کنسرسیوم صنعت موسیقی که عهده‌دارِ ساخت یک سیستم فروشِ برخط موسیقی شده بود، رقابتی برگزار کرده و از مردم دعوت نمود تا سعی کنند سیستمش را بشکنند (که البته انجام این کار برای هر سیستم امنیتیِ جدیدی، عملی کاملاً صحیح است). گروهی از پژوهشگرانِ حوزه‌ی امنیت به رهبری پروفسور ادوارد فِلِتِن[2]، این چالش را بر عهده گرفته و این سیستم را شکستند. سپس مقاله‌ای درباره‌ی یافته‌هایشان نوشته و آن را به کنفرانس امنیتی USENIX تسلیم کردند، که در آن‌جا مورد بررسی قرار گرفته و پذیرفته شد. قبل از ارائه‌ی این مقاله، فِلِتِن نامه‌ای از انجمن صنعت ضبط موسیقی ایالات متحده آمریکا[3] دریافت کرد که در آن تهدید کرده بود اگر مقاله‌ی مذکور را منتشر کنند، آن‌ها را طبق قانون DMCA تحت پیگرد قانونی قرار خواهد داد.

1. Digital Millennium Copyright Act 2. Edward Felten
3. RIAA: Recording Industry Association of America

واکنش آن‌ها عبارت بود از ارائه‌ی دادخواستی به یک دادگاه فدرال و طرح این پرسش که آیا انتشار مقالات علمی در موضوع پژوهش امنیتی، قانونی و مُجاز هست یا خیر. انجمن از ترس آن‌که رأی دادگاه بر ضد آن‌ها باشد، تهدیدش را پس گرفت و دادگاه نیز دادخواست فِلتِن را غیروارد تشخیص داد. بدون تردید آنچه انجمن را به این کار ترغیب کرد، موضع ضعف خودش بود: خودش مردم را دعوت به تلاش برای شکستن سیستمش کرده بود و سپس تهدید کرده بود از تعدادی از همین مردم برای آن‌که دعوت انجمن به چالش را پذیرفته بودند، شکایت خواهد کرد. با پس گرفتنِ تهدید، مقاله هم به چاپ رسید (Craver و همکاران، ۲۰۰۱). یک مواجهه‌ی جدید به طور مَجازی محقق گردید.

در همین اثناء، سرقت ادبیِ موسیقی و فیلم‌ها سببی گردید برای رشدی عظیم در شبکه‌های همتا ـ به ـ همتا. این موضوع خوشایندِ دارندگانِ کپی‌رایت (که برای ایفای نقششان از DMCA استفاده کرده‌اند) نیست. هم‌اکنون سیستم‌های خودکاری وجود دارند که شبکه‌های همتا ـ به ـ همتا را جستجو می‌کنند و به اپراتورها و کاربرانِ شبکه که مشکوک به نقض قوانین باشند، هشدار می‌دهند. این هشدارها در ایالات متحده با عنوان **اخطار بازپس‌گیری از طرف DMCA**[1] شناخته می‌شوند. این گشت‌زنی، یک جور مسابقه است زیرا دستگیریِ متخلفان کپی‌رایت به صورت قابل اتکا، کار دشواری است. حتی چاپگر شما هم ممکن است سهواً به عنوان یک مجرم شناخته شود (Piatek و همکاران، ۲۰۰۸).

موضوعی که با این بحث مرتبط می‌باشد، حوزه‌ی **دکترین استفاده‌ی منصفانه**[2] است که توسط متولیان امر قضا در کشورهای مختلف، برقرار شده است. این دکترین می‌گوید که خریدارانِ یک اثرِ دارای کپی‌رایت برای کپی کردنِ آن اثر، حقوق مشخص و محدودی دارند، شامل حق نقل قول کردنِ بخشی از آن اثر برای مقاصد علمی، استفاده از آن جهت آموزش مطالب درسی در مدارس و دانشگاه‌ها، و در بعضی موارد تهیه‌ی کپی‌های پشتیبان (backup) برای مصرف خود شخص در هنگام بروز خرابی در رسانه‌ی ذخیره کننده‌ی اصلی. ضوابط مربوط به آنچه که مصداق استفاده‌ی منصفانه واقع می‌شوند، شامل این موارد می‌باشند: (۱) به صرفه بودنِ این استفاده از نظر اقتصادی، (۲) درصدی از کل اثر که در حال کپی می‌باشد، و (۳) تأثیر این کپی کردن در میزان فروش اثر. از آن‌جا که DMCA و قوانین مشابهِ آن در اتحادیه‌ی اروپا، هر آنچه که محافظت از کپی کردن را در معرض خطر قرار دهد، منع می‌کنند لذا بر طبق این قوانین، استفاده‌ی منصفانه‌ی مُجاز نیز ممنوع می‌باشد. در حقیقت، DMCA حقوق قبلیِ کاربران را از آن‌ها گرفت تا قدرت بیشتری به فروشندگان محتوا بدهد. برخورد قدرت‌ها در آینده اجتناب‌ناپذیر است.

مورد دیگری که در حال ایجاد و توسعه است و بر روی تعادلی که DMCA میان حقوق صاحبان اثر و کاربران برقرار می‌کند، تأثیر می‌گذارد، **محاسبات قابل اعتماد**[3] می‌باشد که از طرف گروه‌های صنعتی مانند **TCG**[4] (که رهبری آن با شرکت‌هایی مانند اینتل و مایکروسافت است) پشتیبانی

1. DMCA takedown notice 2. Fair use doctrine 3. Trusted computing 4. Trusted Computing Group

می‌شود. ایده عبارت‌است‌از تأمین حمایت از نظارت دقیق بر رفتارهای مختلف کاربر (مانند پخش موسیقیِ دزدیده شده) در سطح زیرینِ سیستم عامل به منظور جلوگیری از رفتار ناخواسته. این کار با یک تراشه‌ی کوچک به نام **TPM** (**ماجول platform مورد اعتماد**[1]) که انجام مداخلات تخریبی در آن دشوار است، انجام می‌شود. امروزه اغلب PCهایی که فروخته می‌شوند، مجهز به یک TPM هستند. این سیستم اجازه می‌دهد تا نرم‌افزاری که توسط صاحبان محتوا نوشته شده است، PC ها را جوری دستکاری کند که کاربران نتوانند تغییری در آن ایجاد کنند. در این‌جا پرسش این است که در محاسباتِ قابل اعتماد، چه کسی قابل اعتماد است؟ قاعدتاً این شخص، کاربر نیست. نیازی به گفتن نیست که پیامدهای اجتماعی این نظام بسیار گسترده است. اتفاق خوبی است که بالاخره بخش صنعت نسبت به امنیت توجه می‌کند، اما جای تأسف دارد که عاملِ آن، اجبار از سوی قانون کپی‌رایت است و دلیلِ این توجه به امنیت، مبارزه با ویروس‌ها، کرک کننده‌ها (cracker)، نفوذگرها، و سایر مباحث امنیتی که دغدغه‌ی اغلب مردم می‌باشد، نبوده است.

خلاصه آن‌که تا سال‌ها، قانون‌گذاران و وکلا بابت ایجاد تعادل مابین افراد ذی‌نفوذ در امور اقتصادی و صاحبان کپی‌رایت از یک طرف، با دغدغه و علاقه‌ی عموم مردم از طرف دیگر، مشغول خواهند بود. در این‌جا هیچ تفاوتی میان فضای سایبری (cyberspace) با فضای زندگی واقعی (meatspace) وجود ندارد: دائماً گروهی در حال رقابت با گروه دیگر است که نتیجه‌اش جنگ قدرت، دعواهای قضایی، و سرانجام (امیدواریم) یک جور گره‌گشایی در کار می‌باشد. این روند دست کم تا زمانی که فناوریِ جدیدی از راه برسد که این نظام را برهم بزند، ادامه خواهد داشت.

۸-۱۱ خلاصه

رمزنگاری ابزاری است که می‌تواند برای محرمانه نگه داشتنِ اطلاعات و اطمینان از جامعیت و اصالت[2] اطلاعات به کار رود. تمام سیستم‌های رمزگونه‌ی امروزی مبتنی بر اصل کیرشهف هستند، یعنی داشتنِ یک الگوریتم شناخته شده برای عموم و یک کلید سرّی. بسیاری از الگوریتم‌های رمزگونه از تبدیلات پیچیده‌ای استفاده می‌کنند که مستلزم جایگزین‌سازی‌ها و جایگشت‌هایی هستند تا متن آشکار را به متن رمزی تبدیل کنند. با این وجود اگر رمزنگاری کوانتومی عملاً انجام‌پذیر شود، ممکن است استفاده از پَدهای یک بار ـ مصرف منجر به ایجاد سیستم‌های رمزنگاری‌ای شود که حقیقتاً غیرقابل شکستن باشند.

الگوریتم‌های رمزگونه را می‌توان به دو گروه الگوریتم‌های کلید ـ متقارن و الگوریتم‌های کلید ـ عمومی تقسیم نمود. الگوریتم‌های کلید ـ متقارن بیت‌ها را طی یک سری راندهای پارامتربندی شده توسط کلید، تکه ـ تکه می‌کنند تا متن آشکار را به متن رمزی تبدیل کنند. در حال حاضر روش‌های AES (ریندال) و DES سه‌گانه پُرطرفدارترین الگوریتم‌های کلید ـ متقارن هستند. این الگوریتم‌ها

1. Trusted Platform Module 2. Authenticity

می‌توانند در مودِ کتاب کد الکترونیکی، مودِ زنجیر کردنِ بلوک رمزی، مودِ رمز جریانی، مودِ شمارنده، و سایر مودها استفاده شوند.

الگوریتم‌های کلید ـ عمومی دارای این ویژگی هستند که برای رمزگذاری و رمزبَرداری از کلیدهای متفاوتی در آن‌ها استفاده می‌شود، و کلید رمزبَرداری نمی‌تواند از کلید رمزگذاری به دست آید. این ویژگی به ما این امکان را می‌دهد که کلید عمومی را منتشر کرده و به اطلاع عموم برسانیم. الگوریتم اصلیِ کلید ـ عمومی، RSA است. این الگوریتم قدرت خود را از این حقیقت برگرفته که مضرب‌گیری از اعداد بزرگ بسیار دشوار است.

لازم است که اسناد قانونی، تجاری و سایر اسناد امضا شوند. از همین رو نظام‌های مختلفی برای امضاهای دیجیتالی ابداع شده‌اند که هم از الگوریتم‌های کلید ـ متقارن و هم از الگوریتم‌های کلید ـ عمومی استفاده می‌کنند. عمدتاً پیغام‌هایی که بایستی امضا شوند، با استفاده از الگوریتم‌هایی نظیر SHA-1 درهم‌سازی شده و سپس به جای آن‌که پیغام‌های اولیه امضا شوند، این پیغام‌های درهم‌سازی شده امضا می‌شوند.

مدیریت کلید ـ عمومی می‌تواند با استفاده از گواهینامه‌ها انجام شود. گواهینامه‌ها اسنادی هستند که یک موجودیت اصلی (principal) را به یک کلید عمومی هم‌بند (bind) می‌کنند. گواهینامه‌ها یا توسط یک مرجع قابل اعتماد امضا می‌شوند و یا توسط کسی امضا می‌شوند که به صورت بازگشتی (recursive) به یک مرجع قابل اعتماد می‌رسند. ریشه‌ی این زنجیره مجبور است از قبل به دست آید، اما مرورگرها عموماً تعداد زیادی گواهینامه‌ی ریشه درون خودشان دارند.

این ابزارِ رمزگونه می‌توانند در ترافیک امنِ شبکه مورد استفاده قرار گیرند. در لایه‌ی شبکه IPsec عمل می‌کند، به این ترتیب که جریان‌های بسته را از یک میزبان به میزبانِ دیگر، رمزگذاری می‌کند. دیواره‌های آتش می‌توانند ترافیک ورودی یا خروجی یک سازمان را غربالَ کنند، که این کار غالباً بر مبنای پروتکل و پورتِ مورد استفاده انجام می‌شود. شبکه‌های خصوصیِ مَجازی می‌توانند یک شبکه‌ی استیجاریِ قدیمی را شبیه‌سازی کنند تا ویژگی‌های امنیتیِ خاصی که مورد نظر هستند تأمین شوند. و سرانجامِ آن‌که، شبکه‌های بی‌سیم برای آن‌که مبادا کسی بتواند همه‌ی پیغام‌ها را بخواند، نیازمند امنیت خوبی هستند که پروتکل‌هایی از قبیل 802.11i چنین امنیتی را فراهم می‌کنند.

هنگامی‌که دو طرف، یک نشست را برقرار می‌کنند، بایستی یکدیگر را تصدیق هویت کنند و در صورت لزوم یک کلید نشستِ اشتراکی ایجاد کنند. انواع پروتکل‌های تصدیق هویت وجود دارند از قبیل پروتکل‌هایی که از یک عاملِ غیر از دو طرفِ ارتباط استفاده می‌کنند، پروتکل دیفی ـ هِلمن، پروتکل کربروس، و رمزنگاری کلید ـ عمومی.

امنیت ایمیل می‌تواند با ترکیبی از روش‌هایی که در این فصل مطالعه کردیم، حاصل شود. برای مثال، PGP پیغام‌ها را فشرده کرده، سپس آن‌ها را با یک کلید سرّی رمزگذاری می‌کند و کلید سرّی، که

با کلید عمومیِ دریافت کننده رمزگذاری شده است را ارسال می‌کند. به‌علاوه، پیغام را نیز درهم‌سازی کرده و پیغام درهم شده‌ی امضا شده را هم ارسال می‌کند تا جامعیت پیغام راستی‌آزمایی گردد.

امنیت وب نیز یک موضوع مهم است که با نام‌گذاریِ امن آغاز می‌شود. پروتکل DNSsec روشی جهت جلوگیری از تقلیدِ DNS فراهم می‌کند. اغلب سایت‌های وبِ مرتبط با تجارت الکترونیک برای برقراری نشست‌های امن و تصدیق هویت شده مابین مشتری و خدمتگزار، از SSL/TLS استفاده می‌کنند. روش‌های متنوعی استفاده می‌شوند که با کدِ در حال حرکت مرتبط هستند، مخصوصا سندباکس کردن و امضا کردنِ کد.

اینترنت مباحث متعددی را مطرح می‌سازد که در این مباحث، فناوری به شدت با سیاست عمومی در تعامل قرار دارد. بعضی از این حوزه‌ها شامل حفظ حریم خصوصی، آزادی بیان، و کپی‌رایت می‌باشند.

مسائل

۱. رمز جایگزین‌سازیِ تک ـ الفبایی (Monoalphabetic substitution cipher) زیر را بشکنید. متن آشکار فقط از حروف تشکیل شده است و قطعه شعر معروفی از لوئیس کارول[1] است.

kfd ktbd fzm eubd kfd pzyiom mztx ku kzyg ur bzha kfthcm
ur mftnm zhx mfudm zhx mdzythc pzq ur ezsszcdm zhx gthem
zhx pfa kfd mdz tm sutythc fuk zhx pfdkfdi ntcm fzld pthem
sok pztk z stk kfd uamkdim eitdx sdruid pd fzld uoi efzk
rui mubd u rom zid uok ur sidzkf zhx zyy u rom zid rzk
hu foiia mztx kfd ezindhkdi kfda kfzhgdx ftb boef rui kfzk

۲. رمز مبتنی بر جابجاسازی ستونی زیر را بشکنید. متن آشکار برگرفته از یک کتاب کامپیوتری پُرطرفدار است، بنابراین احتمالاً واژه‌ی "computer" در آن قرار دارد. متن آشکار تماماً شامل حروف است (بدون فضای خالی). متن رمزی به بلوک‌های ۵ کاراکتری شکسته شده تا خواناتر باشد.

aauan cvlre rurnn dltme aeepb ytust iceat npmey iicgo gorch srsoc
nntii imiha oofpa gsivt tpsit lbolr otoex

۳. باب برای رمزگذاری پیغامش به آلیس از یک رمز جایگزین‌سازی استفاده کرده است. او سپس برای امنیتِ بیشتر، مجدداً پیغام را با استفاده از یک رمز جابجاسازی رمزگذاری نمود. اگر باب اول از رمز جابجاسازی استفاده می‌کرد و سپس بر روی نتیجه، رمز جایگزین‌سازی را انجام می‌داد، آیا نتیجه فرق می‌کرد؟ پاسخ خود را شرح دهید.

1. Lewis Carroll

۴. یک پَد یک بارـ مصرف ۷۷ ـ بیتی پیدا کنید که عبارت "Donald Duck" را از متن رمزی شکل ۸-۴ تولید کند.

۵. اگر هنگامی‌که در حال استفاده از رمزنگاری کوانتومی هستیم، ترودی فوتون‌ها را به دست آورده و آن‌ها را بازـ تولید کند، بعضی از آن‌ها را اشتباه خواهد فهمد و در نتیجه خطاهایی در پَد یک‌بارـ مصرف باب به وجود خواهد آمد. به طور متوسط چه کسری از بیت‌های پَد یک بارـ مصرف باب خطا خواهند شد؟

۶. در شکل ۸-۶، جعبه‌های P و جعبه‌های S به صورت یک ـ درـ میان قرار داده شده‌اند. هر چند این آرایش به لحاظ زیبایی ـ شناختی مطبوع است، ولی اگر ابتدا تمام جعبه‌های P باشند و سپس تمام جعبه‌های S آیا امن‌تر خواهد بود؟ در مورد پاسختان بحث کنید.

۷. در متن، چنین محاسبه کردیم که یک ماشینِ رمزـ شکن با یک میلیون پردازشگر که توان تحلیل یک کلید را در یک نانوثانیه دارد، 10^{16} سال طول خواهد کشید تا ویرایش ۱۲۸ـ بیتیِ AES را بشکند. بیایید حساب کنیم چقدر طول خواهد کشید تا این زمان به یک سال کاهش یابد، که البته زمانِ یکسال، هنوز هم زمانی طولانی است.برای رسیدن به این هدف ما نیاز به کامپیوترهایی داریم که 10^{16} بار سریع‌تر باشند. اگر قانون مور همچنان برقرار بماند (یعنی هر ۱۸ ماه یک بار، توان محاسباتی دو برابر شود) چند سال طول خواهد کشید تا یک کامپیوتر موازی بتواند زمانِ شکستن رمز را به یک سال برساند؟

۸. استاندارد AES کلید ۲۵۶ـ بیتی را حمایت می‌کند. استاندارد AES-256 چند کلید دارد؟ جستجو کنید آیا می‌توانید اعدادی با همین اندازه را در فیزیک، شیمی، یا ستاره‌شناسی بیابید؟ جهت جستجوی اعداد بزرگ از اینترنت کمک بگیرید. از تحقیقی که انجام داده‌اید، برداشتی را استنباط نمایید.

۹. فرض کنید یک پیغام با استفاده از DES در مود شمارنده رمزگذاری شده است. یک بیت از متن رمزی در بلوک C_i به یک باره از 0 به 1 تغییر می‌کند. متن آشکار چه اندازه تحریف خواهد شد؟

۱۰. این بار روش زنجیر کردنِ بلوک متن رمزی را در نظر بگیرید. به جای آن‌که یک بیت 0 به 1 تبدیل شود، یک 0 اضافی بعد از C_i در جریان متن رمزی درج می‌شود. متن آشکار چه اندازه تحریف خواهد شد؟

۱۱. مود زنجیر کردنِ بلوک رمزی را با مود بازخورد رمز، از نظر تعداد عملیات رمزگذاریِ لازم برای تبدیل یک فایل بزرگ، مقایسه کنید. کدام یک کارآمدتر است و چقدر؟

۱۲. کلیدهای RSA ترودی به این صورت هستند: $n_t = 33$, $d_t = 3$, $e_t = 7$. ترودی درمی‌یابد که کلید عمومی باب $n_b = 33$, $e_b = 3$ است.

الف. ترودی برای خواندنِ پیغام‌های رمزگذاری شده که به باب می‌روند، چگونه می‌تواند از این اطلاعات استفاده کند؟

ب. بر اساس نتایجی که از قسمت (الف) گرفته‌اید، تعداد جفت ـ کلیدهای عمومیِ امن برای یک جفت مقدار خاصِ p و q را محاسبه کنید.

۱۳. فرض کنید کاربری به نام ماریا متوجه می‌شود که کلید RSA خصوصی‌اش یعنی $(d1, n1)$ عین کلید RSA عمومی یک کاربر دیگر به نام فرانسیس است: $(e2, n2)$. به عبارت دیگر $d1 = e2$ و $n1 = n2$ است. آیا ماریا باید کلیدهای عمومی و خصوصی‌اش را تغییر دهد؟ پاسختان را شرح دهید.

۱۴. استفاده از مود شمارنده را به صورتی که در شکل ۸–۱۵ نشان داده، اما با $IV = 0$ در نظر بگیرید. آیا استفاده از 0 به طور کلی تهدیدی برای امنیت رمز می‌باشد؟

۱۵. پروتکل امضا کردن که در شکل ۸–۱۸ نشان داده شده، یک ضعف دارد. اگر باب دچار خرابی شود، ممکن است محتوای RAM اش را از دست بدهد. این موضوع چه مشکلاتی به وجود می‌آورَد و باب برای اجتناب از این مشکلات چه کار می‌تواند بکند؟

۱۶. در شکل ۸–۲۰ می‌بینیم که آلیس چگونه می‌تواند یک پیغامِ امضا شده به باب ارسال کند. اگر ترودی P را عوض کند، باب می‌تواند تشخیص دهد. اما اگر ترودی هم P و هم امضا را تغییر دهد، چه اتفاقی می‌افتد؟

۱۷. یک کلاس ریاضی ۲۵ دانشجو دارد. فرض کنید تمام دانشجویان در نیمه‌ی اول سال متولد شده‌اند ــ بین اول ژانویه و ۳۰ ام ژوئیه ــ چقدر احتمال دارد که دست کم دو دانشجو یک تاریخ تولد داشته باشند؟ فرض کنید هیچ‌کدام‌شان در روز اضافه‌ی مربوط به سال‌های کبیسه متولد نشده‌اند، بنابراین ۱۸۱ روز تولدِ محتمل وجود دارند.

۱۸. تلاش شکست خورده‌ی آلیس در به دست آوردنِ کلید عمومی باب در شکل ۸–۲۳ را در نظر بگیرید. فرض کنید باب و آلیس هم‌اکنون در یک کلید سرّی، اشتراک دارند، اما آلیس کلید عمومی باب را می‌خواهد. آیا راهی برای به دست آوردنِ آن به صورت امن وجود دارد؟ اگر بله، چگونه؟

۱۹. آلیس قصد دارد با استفاده از رمزنگاریِ کلید ـ عمومی با باب ارتباط داشته باشد. آلیس یک اتصال با شخصی برقرار می‌کند که امیدوار است باب باشد. آلیس از این شخص کلید عمومی‌اش را می‌خواهد، و او نیز کلید عمومی‌اش را به صورت متن آشکار ارسال می‌کند، همراه با یک گواهینامه‌ی X.509 که توسط CA ریشه امضا شده است. آلیس در حال حاضر کلید عمومی CA ریشه را دارد. آلیس چه مراحلی را بایستی طی کند تا راستی‌آزمایی کند که در حال گفتگو با باب است؟ فرض کنید باب در این مورد که با چه کسی گفتگو می‌کند، مواظبتی نمی‌کند (مثلاً باب نوعی خدمتگزار عمومی است).

۲۰. فرض کنید سیستمی از PKI ای استفاده می‌کند که مبتنی است بر یک سلسله مراتبِ با ساختار درخت‌واره‌ای از CA ها. آلیس می‌خواهد با باب مرتبط شود، و بعد از برقراریِ یک کانالِ ارتباطی با باب، از باب گواهینامه‌ای دریافت می‌کند که توسط CA X امضا شده است. فرض کنید آلیس

هرگز درباره‌ی X چیزی به گوشش نخورده. آلیس چه مراحلی را باید طی کند تا راستی‌آزمایی کند که در حال گفتگو با باب است؟

۲۱. آیا IPsec ای که از AH استفاده می‌کند، اگر یکی از ماشین‌ها در پشت یک جعبه‌ی NAT باشد، می‌تواند در مود حمل به کار رود؟ پاسخ خود را شرح دهید.

۲۲. یک مزیت برای استفاده از HMAC ها علاوه بر RSA، در امضا کردنِ پیغام‌های درهم‌سازی شده‌ی SHA-1 ارائه دهید.

۲۳. یک دلیل ارائه دهید که چرا دیواره‌ی آتش پیکربندی می‌شود تا ترافیک ورودی را بررسی کند. یک دلیل ارائه دهید که چرا برای بررسی ترافیک خروجی پیکربندی می‌شود. فکر می‌کنید آیا احتمال دارد این بررسی‌ها موفق باشند؟

۲۴. سازمانی را در نظر بگیرید که جهت اتصال امنِ سایت‌هایش از طریق اینترنت، از VPNاستفاده می‌کند. جیم که یکی از کاربران در این سازمان است، از VPN استفاده می‌کند تا با کارفرمایش، مری، مرتبط شود. یک نوع ارتباط میان جیم و مری را توضیح دهید که به رمزگذاری یا مکانیزم امنیتیِ دیگری نیاز نداشته باشد، همچنین نوع دیگری از ارتباط را توضیح دهید که نیازمند رمزگذاری یا سایر مکانیزم‌های امنیتی باشد. پاسخ خود را شرح دهید.

۲۵. در پروتکلِ شکل ۸-۳۴، تغییر کوچکی در یکی از پیغام‌ها بدهید تا پروتکل نسبت به حمله‌ی بازتاب مقاوم گردد. توضیح دهید که چرا تغییر شما، عمل می‌کند.

۲۶. از روش تبادل کلید دیفی ـ هِلمَن جهت برقراری یک کلید سرّی میان آلیس و باب استفاده می‌شود. آلیس به باب (227, 5, 82) را ارسال می‌کند. باب با ارسال (125) پاسخ می‌دهد. شماره‌ی سرّی آلیس، یعنی x، عدد 12 است و شماره‌ی سرّی باب، یعنی y، عدد 3 است. نشان دهید آلیس و باب چگونه کلید سرّی را محاسبه می‌کنند.

۲۷. اگر آلیس و باب هرگز یکدیگر را ملاقات نکرده باشند، هیچ کلید سرّی‌ای را به اشتراک نگذاشته باشند، و هیچ گواهینامه‌ای نداشته باشند، معهذا هنوز هم می‌توانند با استفاده از الگوریتم دیفی ـ هِلمَن، یک کلید سرّیِ اشتراکی را برقرار کنند. توضیح دهید به چه علت حفاظت در برابر یک حمله‌ی فردی ـ در ـ میانه، بسیار دشوار می‌باشد.

۲۸. در پروتکل شکل ۸-۳۹، چرا A متن آشکار را همراه با کلید نشستِ رمزگذاری شده، ارسال می‌کند؟

۲۹. در پروتکل شکل ۸-۳۹، اشاره کردیم که شروع کردنِ پیغام‌های متن آشکار با ۳۲ بیت صفر، یک ریسک امنیتی است. فرض کنید که هر پیغام با یک عدد تصادفی به ازای هر کاربر آغاز می‌شود، که در واقع یک کلید سرّیِ دوم است که فقط کاربرِ آن کلید و KDC آن را می‌شناسند. آیا این کار حمله‌ی متن آشکارِ شناخته شده (known plaintext attack) را محدود می‌کند؟ چرا؟

۳۰. در پروتکل نیدهام ـ شرودر، آلیس دو چالش تولید می‌کند: R_A و R_{A2} . این کار به نظر نوعی افراط است. آیا یک چالش، همین کار را انجام نخواهد داد؟

۳۱. سازمانی را در نظر بگیرید که برای تصدیق هویت از کربروس استفاده می‌کند. به لحاظ امنیت و دسترس‌پذیریِ سرویس، اگر AS یا TGS ساقط شوند چه اثری دارد؟

۳۲. در پیغام 7 از پروتکل تصدیق هویتِ کلید عمومیِ شکل ۸–۴۳، R_B با K_S رمزگذاری می‌شود. آیا این رمزگذاری ضروری است، یا کافی است آن را به صورت متن آشکار، برگردانَد؟ پاسختان را شرح دهید.

۳۳. پایانه‌های POS که از کارت‌های مغناطیسی و کدهای PIN استفاده می‌کنند، یک نقص مهلک دارند: یک مغازه‌دارِ بَدنهاد می‌تواند کارت ـ خوانش را به نحوی تنظیم کند که تمام اطلاعاتِ روی کارت و کد PIN را ثبت کند، با این هدف که بعداً تراکنش‌های (جعلی) اضافی اعلان نماید. نسل آینده‌ی پایانه‌ها از کارت‌هایی استفاده خواهندکرد که یک CPU کامل، یک صفحه‌کلید، و یک نمایشگر بسیار کوچک بر روی کارت دارند. پروتکلی برای اینِ سیستم ابداع کنید که مغازه‌دارانِ بَدنهاد نتوانند آن را بشکنند.

۳۴. دو دلیل ذکر کنید که چرا PGP پیغام‌ها را فشرده می‌کند.

۳۵. آیا یک پیغام PGP می‌تواند چندپخشی شود؟ چه محدودیت‌هایی اِعمال خواهند شد؟

۳۶. با این فرض که همه در اینترنت از PGP استفاده می‌کنند، آیا یک پیغام PGP می‌تواند به یک آدرس دلخواه در اینترنت ارسال شود به‌طوری‌که همه‌ی علاقمندانِ آن بتوانند این آدرس را به صورت صحیح کدبَرداری کنند؟ درباره‌ی پاسخ خود بحث نمایید.

۳۷. حمله‌ای که در شکل ۸–۴۷ نشان داده شده، یک مرحله را جا انداخته است. مرحله‌ی مورد نظر برای انجام "تقلید" ضروری نیست اما لحاظ کردنِ آن، در حقیقت می‌تواند پتانسیلِ بَدگمانی را کاهش دهد. این مرحله‌ی جا افتاده چیست؟

۳۸. پروتکل حمل داده‌ی SSL مستلزم دو نانس است، همان‌طور که مستلزم یک کلید premaster هم می‌باشد. استفاده از نانس‌ها چه مقداری (در صورتِ وجود) دارد؟

۳۹. یک تصویر ۵۱۲ × ۲۰۴۸ پیکسل را در نظر بگیرید. می‌خواهید یک فایل به اندازه‌ی 2.5 MB را رمزگذاری کنید. چه کسری از فایل را می‌تواند در این تصویر رمزگذاری کنید؟ اگر فایل را به میزان یک ـ چهارمِ اندازه‌ی اولیه‌اش فشرده کنید، چه کسری را خواهید توانست رمزگذاری کنید؟ محاسباتتان را بیان نمایید.

۴۰. تصویر شکل ۸–۵۴ (ب) حاوی متن ASCII مربوط به ۵ نمایشنامه از شکسپیر است. آیا به جای متن، می‌تواند موسیقی را در میان گورخرهاَ پنهان نمود؟ اگر بله، چطور عمل خواهد کرد و چه میزان موسیقی می‌تواند در این عکس پنهان گردد؟ اگر خیر، دلیلش چیست؟

۴۱. به شما یک فایل متنی با اندازه‌ی 60 MB داده شده که بایستی با استفاده از نهان‌نگاری در بیت‌های با درجه‌ی پایینِ هر یک از رنگ‌ها در یک فایل تصویر، رمزگذاری شود. اندازه‌ی تصویر چقدر باید باشد تا کل فایل رمزنگاری شود؟ اگر فایل ابتدا فشرده‌سازی شود و به یک سوم از اندازه‌ی اولیه‌اش برسد، آنگاه اندازه‌ی تصویر چقدر باید باشد؟ پاسخ خود را بر حسب پیکسل بدهید و محاسباتتان را ارائه کنید. فرض کنید تصاویر دارای نسبت ابعادِ (aspect ratio) ۳:۲ هستند، برای مثال، ۲۰۰۰ × ۳۰۰۰ پیکسل.

۴۲. آلیس یک کاربر پُرکارِ (heavy user) بازـ ایمیل‌کننده‌ی نوعِ ۱ بود. او پیغام‌های زیادی به گروه خبریِ مورد نظرش می‌فرستاد (یعنی گروه *alt.fanclub.alice*) و همه باید متوجه می‌شدند که پیغام‌ها از سوی آلیس آمده زیرا آن‌ها همگی یک اسم مستعار یکسان داشتند. با فرضِ این‌که بازـ ایمیل کننده به طور صحیح کار می‌کرده، ترودی نمی‌توانسته خودش را جای آلیس جا بزند. بعد از آن‌که بازـ ایمیل کننده‌های نوع ۱ از کار افتادند، آلیس به یک بازـ ایمیل کننده‌ی cypherpunk سوئیچ کرد و یک ریسمان جدید به گروه خبری‌اش را آغاز نمود. راهی برای آلیس پیدا کنید تا از این‌که ترودی پیغام‌های جدیدی را از قولِ آلیس به گروه خبری بفرستد، جلوگیری نماید؟

۴۳. با جستجو در اینترنت، یک مورد جالب که مستلزم حفظ حریم خصوصی باشد را پیدا کرده و یک گزارش تک صفحه‌ای درباره‌اش بنویسید.

۴۴. برنامه‌ای بنویسید که ورودی‌اش را با XOR کردنِ همان ورودی با یک جریان کلید (keystream)، رمزگذاری کند. برای تولید جریان کلید، یک تولید کننده‌ی خوب (تا جایی که می‌توانید) برای اعداد تصادفی یا بنویسید و یا یکی پیدا کنید. برنامه باید به عنوان یک فیلتر عمل کند، یعنی متن آشکار را از ورودیِ استاندارد بگیرد و متن رمزی را بر روی خروجیِ استاندارد ایجاد کند (و برعکس). برنامه بایستی یک پارامتر بگیرد: کلیدی که هسته‌ی تولید کننده‌ی عدد تصادفی خواهد بود.

۴۵. یک رویه بنویسید که شکل درهم شده‌ی SHA-1 یک بلوک داده را محاسبه کند. این رویه بایستی دو پارامتر داشته باشد: یک اشاره‌گر به بافر ورودی و یک اشاره‌گر به یک بافر خروجیِ ۲۰ـ بایتی. برای مشاهده‌ی مشخصات دقیق SHA-1، در اینترنت به دنبال FIPS 180-1 جستجو کنید تا مشخصات کاملی را به دست بیاورید.

۴۶. یک تابع بنویسید که یک جریان از کاراکترهای ASCII را بپذیرد و این ورودی را با استفاده از یک رمز جایگزین‌سازی در مود زنجیرکردنِ بلوکِ رمزی، رمزگذاری کند. اندازه‌ی بلوک باید ۸ بایت باشد. برنامه‌ی شما بایستی متن آشکار را از ورودی استاندارد بردارد، و متن رمزی را بر روی خروجی استاندارد چاپ کند. در انجامِ این مسئله، برای آن‌که معلوم شود به انتهای ورودی رسیده‌اید یا خیر، و / یا برای اضافه کردنِ بایت‌های زائد جهت تکمیل بلوک، مُجاز به انتخاب یک

سیستمِ متعارف و قابل قبول می‌باشید. می‌توانید هر فرمت خروجی‌ای که مبهم نباشد را انتخاب نمایید. برنامه باید دو پارامتر بگیرد:

۱. یک اشاره‌گر به بُردارِ راه‌انداز؛ و
۲. یک عدد مانند k که نشان دهنده‌ی جابجاییِ رمز جایگزین‌سازی می‌باشد، چنانکه هر کاراکتر ASCII به وسیله‌ی k امین کاراکتری که جلوتر از خودش است (در حروف الفبا) رمزگذاری خواهد شد.

برای مثال، اگر $x = 3$ باشد، آنگاه A توسط D ، B توسط E ، و همین‌طور تا آخر، رمزگذاری می‌شود. در ارتباط با رسیدنِ آخرین کاراکتر در سری کاراکترهای ASCII مورد نظر، فرض‌های قابل قبول داشته باشید. مطمئن شوید که هر نوع فرضی که در ارتباط با ورودی و الگوریتم رمزگذاری در نظر گرفته‌اید را به روشنی مستند نموده‌اید.

۴۷. هدف از این مسئله آن است که یک درک بهتر از مکانیزم RSA به شما داده شود. تابعی بنویسید که p و q را به عنوان پارامترهای ورودی‌اش بگیرد، کلیدهای RSA عمومی و خصوصی را با استفاده از این پارامترها حساب کند، و n ، z ، d ، و e را به عنوانِ خروجی بر روی خروجی استاندارد چاپ کند. این تابع همچنین باید یک جریان از کاراکترهای ASCII را بپذیرد و این ورودی را با استفاده از کلیدهای RSA حساب شده، رمزگذاری کند. برنامه بایستی متن آشکار را از ورودی استاندارد بردارد و متن رمزی را در خروجی استاندارد چاپ کند. رمزگذاری بایستی کاراکتر ـ محور باشد یعنی هر یک از کاراکترها را از ورودی بردارد و آن را مستقل از سایر کاراکترهای ورودی، رمزگذاری کند. در انجام این مسئله شما برای تعیین رسیدن به انتهای ورودی، مُجاز به استفاده از هر سیستمِ قابل قبولی می‌باشید. می‌توانید از هر فرمتی برای خروجی، مادام که غیرمبهم باشد، استفاده کنید. مطمئن شوید که هر نوع فرضی که در ارتباط با ورودی و الگوریتم رمزگذاری در نظر گرفته‌اید را به روشنی مستند نموده‌اید.