

نمذ دوازدهم: طرح‌های امضای دیجیتال

۱۲-۱- مروری بر امضاهای دیجیتال

در فصل قبلی، این مسئله را بررسی کردیم که رمزگذاری کلید عمومی را چگونه می‌توان برای رسیدن به «محرمانگی» در موقعیت کلید عمومی، مورد استفاده قرار داد. یکپارچگی (یا اصالت) در موقعیت کلید عمومی با استفاده از «طرح‌های امضای دیجیتال» ارائه می‌شود. این طرح‌ها را می‌توان به‌عنوان مشابه کلید عمومی کدهای احراز اصالت پیام در نظر گرفت، هرچند، همان‌طور که مشاهده خواهیم کرد، چندین تفاوت مهم بین این پایه‌ها وجود دارد.

طرح‌های امضا به یک «امضاکننده» S که یک کلید عمومی pk را ایجاد نموده است اجازه می‌دهند تا یک پیام را با استفاده از کلید خصوصی مربوطه sk به‌گونه‌ای امضا نماید که هر فردی که pk را بداند (و بداند که این کلید عمومی توسط S مشخص شده است) می‌تواند «راست آزمایی» کند که پیام از سوی S فرستاده شده است و در زمان انتقال، تغییر نکرده است. (توجه کنید که برخلاف رمزگذاری کلید عمومی، در حوزه‌ی امضاهای دیجیتال، مالک کلید عمومی به‌عنوان یک «فرستنده» عمل می‌کند.) به‌عنوان یک کاربرد معمول، یک شرکت نرم‌افزاری را در نظر بگیرید که می‌خواهد به‌روزرسانی‌های نرم‌افزاری را به شکلی توأم با احراز اصالت، توزیع نماید؛ یعنی، زمانی که شرکت یک به‌روزرسانی را منتشر می‌کند، باید برای هر کدام از مشتریان آن امکان بررسی این امر وجود داشته باشد که این به‌روزرسانی دارای اصالت است و یک طرف سوم مخرب نباید قادر به فریب دادن یک مشتری برای پذیرش یک به‌روزرسانی شود که واقعاً توسط شرکت منتشر نشده است. برای انجام این کار، شرکت می‌تواند یک کلید عمومی pk را به همراه یک کلید خصوصی sk تولید کند و سپس pk را به شیوه‌ای پایدار برای مشتریان خود توزیع نماید درحالی‌که sk را مخفی نگه می‌دارد. (به‌مانند حالت رمزگذاری کلید عمومی، فرض می‌کنیم که این توزیع اولیه‌ی کلید عمومی به‌درستی انجام می‌شود به‌گونه‌ای که همه‌ی مشتریان دارای یک کپی درست از pk باشند. در مثال کنونی، pk را می‌توان به همراه نرم‌افزار اولیه‌ی خریداری شده توسط مشتری، ارسال نمود.) هنگام انتشار یک به‌روزرسانی نرم‌افزاری m ، شرکت یک امضای دیجیتال σ را بر روی m با استفاده از کلید خصوصی

خود sk محاسبه کرده و (m, σ) را برای همه‌ی مشتریان خود ارسال می‌کند. هر مشتری می‌تواند اصالت m را با بررسی این امر واریسی کند که σ یک امضای صحیح بر روی m بر اساس کلید عمومی pk است.

طرف خراب‌کار ممکن است تلاش کند یک به‌روزرسانی جعلی را با ارسال (m', σ') به یک مشتری، صادر نماید که در آن، m' نشان‌دهنده‌ی یک به‌روزرسانی است که هیچ‌گاه توسط شرکت منتشر نشده است. این m' ممکن است یک نسخه‌ی تغییر یافته از یک به‌روزرسانی قبلی باشد یا ممکن است کاملاً جدید بوده و ارتباطی به هیچ‌کدام از به‌روزرسانی‌های قبلی نداشته باشد. با این حال، اگر طرح امضا «امن» باشد (به معنایی که به‌زودی با دقت بیشتر تعریف خواهیم کرد)، آنگاه زمانی که مشتری تلاش می‌کند تا σ' را واریسی کند، درمی‌یابد که یک امضای «نامعتبر» بر روی m' بر اساس pk است و بنابراین امضا را «رد» می‌کند. مشتری عمل رد کردن را انجام خواهد داد حتی اگر m' تنها اندکی با به‌روزرسانی واقعی m متفاوت باشد.

مطلب بالا تنها یک کاربرد نظری امضاهای دیجیتال نیست، بلکه کاربردی است که امروزه به‌صورت گسترده برای توزیع به‌روزرسانی‌های نرم‌افزاری، استفاده می‌شود.

مقایسه با کدهای احراز اصالت پیام

هم کدهای احراز اصالت پیام و هم طرح‌های امضای دیجیتال برای تضمین یکپارچگی پیام‌های ارسالی مورد استفاده قرار می‌گیرند. هرچند بحث موجود در فصل دهم در خصوص مقایسه‌ی وضعیت‌های کلید عمومی و کلید خصوصی اصولاً بر رمزگذاری متمرکز داشت، همان بحث برای یکپارچگی پیام هم اعمال می‌شود. استفاده از امضاهای دیجیتال به‌جای کدهای احراز اصالت پیام باعث ساده شدن توزیع و مدیریت کلید می‌شود، علی‌الخصوص زمانی که فرستنده نیازمند برقراری ارتباط با چندین گیرنده به‌مانند مثال به‌روزرسانی نرم‌افزار در بالا، است. با استفاده از یک طرح امضای دیجیتال، فرستنده از ایجاد یک کلید رمز متمایز برای هر گیرنده‌ی بالقوه اجتناب خواهد کرد و از محاسبه‌ی یک برچسب MAC مجزا برای هر کدام از چنین کلیدهایی نیز اجتناب خواهد نمود. درواقع، فرستنده تنها نیازمند محاسبه‌ی یک امضای یکتا است که می‌تواند توسط تمامی گیرنده‌ها، واریسی شود.

یک مزیت «کیفی» امضاهای دیجیتال در مقایسه با کدهای احراز اصالت پیام این است که امضاها «به‌صورت عمومی قابل واریسی» هستند. این امر به این معنا است که اگر یک گیرنده واریسی نماید که یک امضا بر روی یک پیام مفروض، قانونی است، آنگاه تمامی دیگر طرفینی که این پیام امضا شده را دریافت می‌کنند هم آن را به‌عنوان یک پیام قانونی، واریسی خواهند کرد. این ویژگی توسط کدهای احراز اصالت پیام به دست نخواهد آمد اگر امضاکننده یک کلید مجزا را با هر گیرنده، تسهیم کند؛ در چنین وضعیتی، یک فرستنده‌ی خراب‌کار ممکن است یک برچسب MAC صحیح را بر اساس کلیدی که با گیرنده‌ی A تسهیم می‌کند محاسبه نماید ولی یک برچسب MAC نادرست را بر اساس کلیدی

که با یک کاربر متفاوت B تسهیم کرده است، محاسبه کنند. در این حالت، A می‌داند که یک پیام دارای اصالت را از فرستنده دریافت کرده است ولی هیچ تضمینی ندارد که B موافق این امر باشد.

قابلیت واریسی عمومی به این معنا است که امضاها، «انتقال‌پذیر» هستند: یک امضای σ بر روی یک پیام m توسط یک امضاکننده‌ی S را می‌توان به یک طرف سوم نشان دهد که وی در ادامه می‌تواند خودش واریسی کند که σ یک امضای قانونی بر روی m بر اساس کلید عمومی S است (در اینجا فرض می‌کنیم که این طرف ثالث نیز کلید عمومی S را می‌داند). با ایجاد یک کپی از امضا، این طرف سوم در ادامه می‌تواند امضا را به یک طرف دیگر نشان دهد و وی را قانع کند که S پیام m را تأیید اصالت کرده است و الی‌آخر. قابلیت واریسی عمومی و انتقال‌پذیری برای کاربرد امضاهای دیجیتال برای گواهی‌ها و زیرساخت‌های کلید عمومی، دارای اهمیت حیاتی هستند. همان‌طور که در بخش ۱۲-۷ بحث خواهیم کرد.

طرح‌های امضای دیجیتال همچنین ویژگی بسیار مهم «انکارناپذیری» را فراهم می‌کنند این مقوله به این معنا است که زمانی که S یک پیام را امضا می‌کند، در ادامه نمی‌تواند امضا کردن آن را انکار کند (با این فرض که کلید عمومی S به‌صورت گسترده منتشر و توزیع شده باشد). این بعد از امضاهای دیجیتال برای کاربردهای حقوقی دارای اهمیت حیاتی است که در آن‌ها، گیرنده ممکن است مجبور باشد برای یک طرف ثالث (مثلاً یک قاضی) اثبات نماید که یک امضاکننده واقعاً یک پیام خاص را «گواهی» کرده است (برای مثال، یک قرارداد): با فرض اینکه کلید عمومی S برای قاضی معلوم باشد، پایه هر صورت به شکل عمومی در دسترس باشد، یک امضای معتبر بر روی یک پیام به‌عنوان مدرک قطعی برای این امر عمل خواهد کرد که S واقعاً این پیام را امضا کرده است. کدهای احراز اصالت پیام نمی‌توانند انکارناپذیری را فراهم کنند. برای مشاهده‌ی این امر، فرض کنید کاربران S و R یک کلید k_{SR} را تسهیم می‌کنند و S یک پیام m را برای R به همراه یک برچسب MAC (معتبر) t محاسبه شده با استفاده از این کلید، ارسال می‌کند. از آنجاکه قاضی، k_{SR} را نمی‌داند (درواقع این کلید توسط S و R مخفی می‌شود)، هیچ راهی برای قاضی وجود ندارد که مشخص کند آیا t معتبر است یا خیر. اگر قرار باشد R کلید k_{SR} را برای قاضی افشا کند، هیچ راهی برای قاضی وجود نخواهد داشت که تعیین کند که آیا این کلید همان کلید واقعی تسهیم شده توسط S و R هست یا خیر یا اینکه یک کلید «جعلی» است که توسط R تولید شده است. درنهایت، حتی اگر فرض کنیم که قاضی به شیوه‌ای بتواند کلید واقعی k_{SR} تسهیم شده توسط طرفین را به دست آورد، هیچ راهی برای قاضی وجود ندارد که مشخص کند آیا S برچسب t را تولید کرده است یا R آن را تولید کرده است- این امر از این واقعیت نتیجه می‌شود که کدهای احراز اصالت پیام، یک پایه‌ی کلید «متقارن» هستند؛ هر کاری که می‌تواند انجام دهد، R هم می‌تواند انجام دهد.

به‌مانند حالت رمزگذاری کلید خصوصی در برابر رمزگذاری کلید عمومی، کدهای احراز اصالت پیام دارای این مزیت هستند که کوتاه‌تر بوده و تقریباً ۲ تا ۳ برابر نسبت به امضاهای دیجیتال برای

تولید/وارسی کارا تر هستند. بنابراین، در وضعیت‌هایی که در آن‌ها، قابلیت واریسی عمومی، انتقال‌پذیری و/یا انکارناپذیری مورد نیاز نیستند و فرستنده اصولاً با یک گیرنده‌ی یکتا در ارتباط است (که قادر به تسهیم یک کلید رمز با وی است)، باید از کدهای احراز اصالت پیام استفاده شود.

ارتباط با رمزگذاری کلید عمومی

امضاهای دیجیتال اغلب به اشتباه به عنوان «وارون» رمزگذاری کلید عمومی نگریسته می‌شوند که نقش‌های فرستنده و گیرنده در آن عکس شده‌اند. در واقع، از لحاظ تاریخی^{۴۷}، پیشنهاد شده است که امضاهای دیجیتال را می‌توان با «وارون سازی» رمزگذاری کلید عمومی به دست آورد، یعنی با امضای یک پیام m از طریق رمزگشایی آن (با استفاده از کلید خصوصی) برای به دست آوردن σ و راست آزمایی یک امضای σ از طریق رمزگذاری آن (با استفاده از کلید عمومی مربوطه) و بررسی اینکه آیا نتیجه، برابر با m است یا خیر. توصیه برای ساختن طرح‌های امضا به این شیوه، کاملاً بی‌اساس است: در بسیاری از موارد، این شیوه اصولاً قابل استفاده نیست و در حالت‌هایی که این شیوه را می‌توان به کار برد، منجر به طرح‌های امضایی می‌شود که امن نیستند.

۱۲-۲- تعاریف

امضاهای دیجیتال، همتای کلید عمومی کدهای احراز اصالت پیام هستند و ساختار و تضمین‌های امنیت آن‌ها مشابه هستند. الگوریتمی که فرستنده در اینجا بر روی پیام اعمال می‌کند را با $Sign$ (به جای MAC) نشان می‌دهیم و خروجی این الگوریتم در اینجا یک «امضا» (به جای یک برچسب) نامیده می‌شود. الگوریتمی که گیرنده برای یک پیام و یک امضا اعمال می‌کند تا اعتبار آن را راست آزمایی کند همچنان با $Vrfy$ نشان داده می‌شود.

تعریف ۱-۱۲

یک طرح امضا (دیجیتال) شامل سه الگوریتم زمان چندجمله‌ای تصادفی $(Gen, Sign, Vrfy)$ است به صورتی که:

- ۱) الگوریتم تولید کلید Gen ، یک پارامتر امنیتی 1^n را به عنوان ورودی دریافت کرده و یک زوج کلید (pk, sk) را به عنوان خروجی ارائه می‌کند. این کلیدها به ترتیب کلید عمومی و کلید خصوصی نامیده می‌شوند. فرض می‌کنیم که pk و sk هر کدام دارای طول حداقل n هستند و اینکه n را می‌توان از روی pk یا sk تعیین نمود.

^{۴۷} این دیدگاه بدون شک به این دلیل به وجود می‌آید که، همان‌طور که در بخش ۱۲-۴-۱ مشاهده خواهیم کرد، امضاهای RSA ساده، وارون رمزگذاری RSA ساده هستند. با این حال، نه امضاهای RSA ساده و نه رمزگذاری RSA ساده حتی در مفاهیم حدائلی امنیت صدق نمی‌کنند.

۲) الگوریتم امضاکننده $Sign$ یک کلید خصوصی sk و یک پیام m از فضای پیام (که ممکن است به pk بستگی داشته باشد) را به عنوان ورودی دریافت می‌کند. این الگوریتم، یک امضای σ را به عنوان خروجی ارائه می‌کند و این امر را به صورت $\sigma \leftarrow Sign_{sk}(m)$ می‌نویسیم.

۳) الگوریتم راست آزمایی قطعی $Vrfy$ یک کلید عمومی pk ، یک پیام m و یک امضای σ را به عنوان ورودی دریافت می‌کند. این الگوریتم یک بیت b را به عنوان خروجی ارائه می‌کند که $b = 1$ به معنای «معتبر» و $b = 0$ به معنای «نامعتبر» است. این امر را به صورت $b = Vrfy_{pk}(m, \sigma)$ می‌نویسیم.

الزامی است که مگر با احتمال ناچیز روی (pk, sk) خروجی توسط $Gen(1^n)$ ، برای هر پیام (قانونی) m داشته باشیم $Vrfy_{pk}(m, sign_{sk}(m)) = 1$.

اگر یک تابع l وجود داشته باشد به گونه‌ای که برای هر (pk, sk) خروجی از $Gen(1^n)$ ، فضای پیام برابر با $\{0, 1\}^{l(n)}$ باشد، آنگاه می‌گوییم $(Gen, Sign, Vrfy)$ یک «طرح امضا برای پیام‌های با طول $l(n)$ » است.

ما σ را یک امضای معتبر بر روی یک پیام m (بر اساس یک کلید عمومی مفروض pk که مبتنی بر زمینه درک خواهد شد) می‌نامیم اگر $Vrfy_{pk}(m, \sigma) = 1$.

یک طرح امضا به شیوه‌ی زیر مورد استفاده قرار می‌گیرد. یک طرف S که به عنوان فرستنده عمل می‌کند، $Gen(1^n)$ را اجرا می‌کند تا کلیدهای (pk, sk) را به دست آورد. در ادامه، کلید عمومی pk به عنوان کلید متعلق به S اعلام می‌شود؛ برای مثال، S می‌تواند کلید عمومی را بر روی وبسایت خود یا در یک فهرست عمومی قرار دهد. به مانند حالت رمزگذاری کلید عمومی، فرض می‌کنیم که هر طرف دیگری قادر به دریافت یک کپی قانونی از کلید عمومی S است (ن.ک. بحث ذیل). هنگامی که S می‌خواهد یک پیام m را احراز اصالت کند، امضای $\sigma \leftarrow sign_{sk}(m)$ را محاسبه کرده و (m, σ) را ارسال می‌کند. بعد از دریافت (m, σ) ، گیرنده‌ای که pk را می‌داند می‌تواند اصالت m را با بررسی اینکه آیا $Vrfy_{pk}(m, \sigma) \stackrel{?}{=} 1$ ، راست آزمایی کند. این واریسی هم مشخص می‌کند که S پیام m را ارسال کرده و هم اینکه m طی انتقال، تغییر داده نشده است. با این حال، به مانند حالت کدهای احراز اصالت پیام، این واریسی هیچ چیزی در مورد زمان ارسال m بیان نمی‌کند و حملات تکرار همچنان امکان پذیر هستند (ن.ک. بخش ۴-۲).

این فرض که طرفین می‌توانند یک کپی قانونی از کلید عمومی S را به دست آورند به این معنی است که S قادر به ارسال حداقل یک پیام (یعنی خود pk) به شکلی مطمئن و احراز اصالت شده، خواهد بود. با این حال، اگر ما قادر به ارسال پیام‌ها به شکل مطمئن هستیم، آنگاه اصلاً چه نیازی به یک طرح امضا داریم؟ پاسخ این است که توزیع مطمئن pk یک امر دشوار و پرهزینه است ولی استفاده از یک

۳-۱۲- الگوی چکیده‌سازی-و-امضا

به‌مانند حالت رمزگذاری کلید عمومی در برابر رمزگذاری کلید خصوصی، طرح‌های امضای «بومی» چندین برابر کارایی کمتری نسبت به کدهای احراز اصالت پیام دارند. خوشبختانه، به‌مانند رمزگذاری ترکیبی (ن.ک. بخش ۳-۱۱)، می‌توان عملکرد امضاهای دیجیتال را با هزینه‌ی مجانبی یک عملیات کلید خصوصی، حداقل برای پیام‌هایی که به‌اندازه‌ی کافی طولانی باشند، به دست آورد. این امر را می‌توان با استفاده از رویکرد چکیده‌سازی-و-امضا انجام داد که در ادامه مورد بحث قرار می‌گیرد.

برداشت شهودی مبنای رویکرد چکیده‌سازی-و-امضا، سراسر است. فرض کنید ما یک طرح امضا برای پیام‌های با طول l داریم و می‌خواهیم یک پیام (طولانی‌تر) $m \in \{0,1\}^*$ را امضا کنیم. به‌جای امضای خود m ، می‌توانیم از یک تابع چکیده ساز H برای چکیده‌سازی پیام و رساندن آن به یک «چکیده» با طول ثابت l و سپس امضای چکیده‌ی به‌دست‌آمده، استفاده نماییم. این رویکرد دقیقاً مشابه با رویکرد چکیده‌سازی-و- MAC است که در بخش ۳-۵، بحث شد.

الگوی چکیده‌سازی-و-امضا

ساختمان ۳-۱۲

فرض کنید $\Pi = (Gen, Sign, Vrfy)$ یک طرح امضا برای پیام‌های با طول $l(n)$ بوده و $\Pi_H = (Gen_H, H)$ یک تابع چکیده ساز با طول خروجی $l(n)$ باشد. طرح امضای $\Pi' = (Gen', Sign', Vrfy')$ را به شکل زیر می‌سازیم:

- Gen' : با دریافت 1^n به‌عنوان ورودی، $Gen(1^n)$ را برای به دست آوردن (pk, sk) اجرا کرده و $(Gen_H(1^n))$ را برای به دست آوردن s اجرا می‌کنیم؛ کلید عمومی برابر با (pk, s) و کلید خصوصی برابر با (sk, s) است.
- $Sign'$: با دریافت یک کلید خصوصی (sk, s) و یک پیام $m \in \{0,1\}^*$ خروجی $\sigma \leftarrow Sign_{sk}(H^s(m))$ را ارائه می‌کنیم.
- $Vrfy'$: با دریافت یک کلید عمومی (pk, s) ، یک پیام $m \in \{0,1\}^*$ و یک امضای σ به‌عنوان ورودی، مقدار خروجی 1 را ارائه می‌کنیم اگر و تنها اگر $Vrfy_{pk}(H^s(m), \sigma) = 1$ باشد.

قضیه‌ی ۴-۱۲

اگر Π یک طرح امضای امن برای پیام‌های با طول l و Π_H مقاوم در برابر برخورد باشد، آنگاه ساختمان ۳-۱۲ یک طرح امضای امن (برای پیام‌های با طول دلخواه) است.

طرح امضا به این معنا است که این امر تنها باید «یک‌بار» انجام شود و بعد از آن، تعداد نامحدودی پیام را می‌توان به شکل مطمئن ارسال نمود. به‌علاوه، همان‌طور که در بخش ۷-۱۲ بحث خواهیم کرد، خود طرح‌های امضا برای تضمین توزیع مطمئن «دیگر» کلیدهای عمومی مورد استفاده قرار می‌گیرند. بنابراین، این طرح‌ها یک ابزار اساسی برای ایجاد یک «زیرساخت کلید عمومی» برای حل مسئله‌ی توزیع کلید هستند.

امنیت طرح‌های امضا

برای یک کلید عمومی ثابت pk تولیدشده توسط یک امضاکننده‌ی $\mathcal{K}$ یک «جعل» عبارت است از یک پیام m همراه با یک امضای معتبر σ که در آن، m قبلاً توسط S امضا نشده است. امنیت یک طرح امضا به این معنا است که متخاصم نباید بتواند یک جعل را خروجی دهد حتی اگر امضاهایی بر روی بسیاری دیگر از پیام‌های انتخابی خود، به دست آورد. این تعریف، مشابه مستقیم تعریف امنیت برای کدهای احراز اصالت پیام است و برای بررسی دلایل این تعریف و بحث بیشتر در خصوص آن، خواننده می‌تواند به بخش ۴-۲ مراجعه کند.

اکنون تعریف رسمی امنیت را ارائه می‌کنیم که اصولاً با تعریف ۴-۲ یکسان است. فرض کنید $\Pi = (Gen, Sign, Vrfy)$ یک طرح امضا باشد و آزمایش زیر را برای متخاصم $\mathcal{A}$ و پارامتر n در نظر بگیرید:

آزمایش امضا $Sig - forge_{\mathcal{A}, \Pi}(n)$:

- (۱) $Gen(1^n)$ اجرا می‌شود تا کلیدهای (pk, sk) به دست آیند.
- (۲) متخاصم $\mathcal{A}$ به‌عنوان ورودی، pk و دسترسی به یک پیشگوی $(\cdot, Sign_{sk}(\cdot))$ را دریافت می‌کند. در ادامه، متخاصم، خروجی (m, σ) را ارائه می‌کند. فرض کنید $\mathcal{Q}$ نشان‌دهنده‌ی مجموعه‌ی تمام پرسش‌هایی باشد که $\mathcal{A}$ از پیشگوی خود می‌پرسد.
- (۳) $\mathcal{A}$ موفق می‌شود اگر و تنها اگر $Vrfy_{pk}(m, \sigma) = 1$ (۱) و $m \notin \mathcal{Q}$ (۲) در این حالت، خروجی آزمایش برابر با 1 تعریف می‌شود.

تعریف ۲-۱۲

یک طرح امضای $\Pi = (Gen, Sign, Vrfy)$ غیرقابل جعل ورودی تحت یک حمله‌ی مبتنی بر پیام منتخب وقتی است یا تنها «امن» است اگر برای تمام متخاصمین زمان-چندجمله‌ای تصادفی $\mathcal{A}$ یک تابع ناچیز $negl$ وجود داشته باشد به صورتی که

$$\Pr[Sig - forge_{\mathcal{A}, \Pi}(n) = 1] \leq negl(n).$$

اثبات این قضیه تقریباً با اثبات قضیه ۵-۶ یکسان است.

۱۲-۴- امضاهای RSA

بررسی طرح‌های امضای عینی را با بحثی در خصوص طرح‌های مبتنی بر فرض RSA آغاز می‌کنیم.

۱۲-۴-۱- RSA ساده

در ابتدا یک طرح امضای مبتنی بر RSA ساده را توصیف می‌کنیم. هرچند این طرح، ناامن است ولی نقطه‌ی شروع مفیدی است.

مثل همیشه، فرض می‌کنیم $GenRSA$ یک الگوریتم PPT باشد که با دریافت ورودی 1^n ، به‌عنوان خروجی یک پیمانه N که حاصل ضرب دو عدد اول n -بیتی (مگر با احتمال ناچیز) است را به همراه اعداد صحیح e, d که در رابطه‌ی $ed = 1 \bmod \phi(N)$ صدق می‌کنند، ارائه می‌کند. تولید کلید در RSA ساده، شامل اجرای $GenRSA$ ، اعلام (N, e) به‌عنوان کلید عمومی و اعلام (N, d) به‌عنوان کلید خصوصی است. به‌منظور امضای یک پیام $m \in \mathbb{Z}_N^*$ ، امضاکننده، $\sigma := [m^d \bmod N]$ را محاسبه می‌کند. واریسی یک امضای σ بر روی یک پیام m بر اساس کلید عمومی (N, e) با بررسی این امر انجام می‌شود که آیا $m \stackrel{?}{=} \sigma^e \bmod N$ ن.ک. ساختار ۱۲-۵.

طرح امضای RSA ساده

ساختار ۱۲-۵

فرض کنید $GenRSA$ به شکل تعریف‌شده در متن باشد. یک طرح امضا را به‌صورت زیر تعریف می‌کنیم:

- Gen : با دریافت 1^n به‌عنوان ورودی، $GenRSA(1^n)$ را برای به دست آوردن (N, e, d) اجرا می‌کنیم. کلید عمومی برابر با (N, e) و کلید خصوصی برابر با (N, d) است.
- $Sign$: با دریافت یک کلید خصوصی $sk = (N, d)$ و یک پیام $m \in \mathbb{Z}_N^*$ ، امضای زیر را محاسبه می‌کنیم:

$$\sigma := [m^d \bmod N]$$

- $Verify$: با دریافت یک کلید عمومی $pk = (N, e)$ ، یک پیام $m \in \mathbb{Z}_N^*$ و یک امضای $\sigma \in \mathbb{Z}_N$ به‌عنوان ورودی، خروجی ۱ را ارائه می‌کنیم اگر و تنها اگر $m \stackrel{?}{=} [\sigma^e \bmod N]$

به‌سادگی می‌توان مشاهده نمود که واریسی یک امضا که به‌صورت قانونی تولید شده است، همواره موفق است به این دلیل که

$$\sigma^e = (m^d)^e = m^{[ed \bmod \phi(N)]} = m^1 = m \bmod N.$$

ممکن است انتظار داشته باشیم که این طرح، امن باشد به این دلیل که برای یک متخاصم که تنها کلید عمومی (N, e) را می‌داند، محاسبه‌ی یک امضای معتبر بر روی یک پیام m به نظر نیازمند حل کردن مسئله‌ی RSA است (از آنجاکه امضا دقیقاً برابر با ریشه‌ی m است). متأسفانه، این تفکر اشتباه است. یکی از دلایل اشتباه بودن آن این است که فرض RSA تنها نشان‌دهنده‌ی سختی محاسبه‌ی یک امضا (یعنی، محاسبه‌ی ریشه‌ی m) یک پیام یکنواخت m است؛ این امر هیچ چیزی در مورد سختی محاسبه‌ی یک امضا بر روی یک m غیریکنواخت یا بر روی یک پیام m مفروض انتخاب‌شده توسط مهاجم، نمی‌گوید. به‌علاوه، فرض RSA هیچ چیزی در مورد این مسئله نمی‌گوید که بعد از فهمیدن امضاها بر روی دیگر پیام‌ها، مهاجم قادر به انجام چه کارهایی است. مثال‌های زیر نشان می‌دهند که هر دو این مشاهدات منجر به حملاتی علیه طرح امضای RSA ساده می‌شوند.

یک حمله‌ی بدون پیام

اولین حمله‌ای که توصیف می‌کنیم یک جعل را «تنها با استفاده از کلید عمومی» تولید می‌کند بدون اینکه هیچ امضایی از امضاکننده‌ی قانونی دریافت کند. این حمله به این شکل عمل می‌کند: با داشتن یک کلید عمومی $pk = (N, e)$ ، یک $\sigma \in \mathbb{Z}_N^*$ یکنواخت را انتخاب کرده و $m := [\sigma^e \bmod N]$ را محاسبه می‌کنیم. آنگاه، جعل (m, σ) را ارائه می‌کنیم. فوراً می‌فهمیم که σ یک امضای معتبر بر روی m است و این، یک جعل است چراکه اصلاً هیچ امضایی توسط صاحب کلید عمومی، صادر نشده است. نتیجه می‌گیریم که طرح امضای RSA ساده، در تعریف ۱۲-۲ صدق نمی‌کند.

ممکن است استدلال شود که این مقوله یک حمله‌ی «واقع‌بینانه» نیست چراکه متخاصم «هیچ کنترلی» بر روی پیام m که برای آن، یک امضای معتبر جعل کرده است، نخواهد داشت. این مسئله ناجایی که به تعریف ۱۲-۲ مربوط می‌شود، اهمیتی ندارد و قبلاً بحث کردیم (در فصل چهارم) که چرا فرض معناشناسی برای پیام‌هایی که قرار است با استفاده از یک طرح رمزنگاری، احراز اصالت شوند، خطرناک است. به‌علاوه، متخاصم دارای سطحی از کنترل بر روی m است: برای مثال، با انتخاب چندین مقدار یکنواخت برای σ ، متخاصم (با احتمال بالا) می‌تواند یک m را به دست آورد که تعدادی از بیت‌های آن به شکل مطلوب وی، چیده شده باشند. با انتخاب σ به شکلی خاص، این امکان هم می‌تواند وجود داشته باشد که بر پیام به‌دست‌آمده که برای آن، یک جعل خروجی داده شده است، تأثیر بگذاریم.

جعل یک امضا بر روی یک پیام دلخواه

یک حمله‌ی دشوارتر علیه طرح امضای RSA ساده نیازمند آن است که متخاصم «دو» امضا از امضاکننده به دست آورد ولی به متخاصم اجازه می‌دهد که یک امضای جعل‌شده بر روی هر پیام انتخابی خود، خروجی دهد. فرض کنید متخاصم می‌خواهد امضایی بر پیام $m \in \mathbb{Z}_N^*$ بر اساس کلید عمومی $pk = (N, e)$ جعل نماید. متخاصم، $m_1, m_2 \in \mathbb{Z}_N^*$ دلخواه متمایز از m را به‌گونه‌ای انتخاب می‌کند که $m = m_1 \cdot m_2 \bmod N$ باشد. در ادامه، امضاها σ_1, σ_2 را به ترتیب بر روی m_1, m_2 به

ساختار ۱۲-۶

فرض کنید $GenRSA$ به شکل موجود در بخش‌های قبلی باشد و یک طرح امضا را به صورت زیر بسازید:

- Gen : با دریافت 1^n به عنوان ورودی، $GenRSA(1^n)$ را اجرا می‌کنیم تا (N, e, d) به دست آید. کلید عمومی برابر با (N, e) و کلید خصوصی برابر با (N, d) است. به عنوان بخشی از تولید کلید، یک تابع $H: \{0, 1\}^* \rightarrow \mathbb{Z}_N^*$ مشخص می‌شود. ولی ما این مقوله را به شکل ضمنی باقی می‌گذاریم.
- $Sign$: با دریافت یک کلید خصوصی (N, d) و یک پیام $m \in \{0, 1\}^*$ مقدار زیر را محاسبه می‌کنیم:

$$\sigma = [H(m)^d \bmod N]$$

- $Vrfy$: با دریافت یک کلید عمومی (N, e) ، یک پیام m و یک امضای σ خروجی ۱ را ارائه می‌کنیم اگر و تنها اگر $H(m) \stackrel{?}{=} \sigma^e \bmod N$ باشد.

هیچ روشی برای انتخاب H به گونه‌ای که بتوان امن بودن ساختار ۱۲-۶ را اثبات کرد، شناخته نشده است. باین حال، می‌توان امنیت را اثبات نمود اگر H به عنوان یک پیشگوی تصادفی مدل سازی شده باشد که ورودی‌های خود را به شکل یکنواخت بر روی $\mathbb{Z}_N^*$ نگاشت می‌کند؛ طرح در این حالت، طرح امضای «چکیده ساز دامنه-کامل RSA » ($RSA - FDH$) نامیده می‌شود. می‌توان بررسی کرد که یک تابع تصادفی از این نوع، در الزامات بحث شده در بند قبلی صدق می‌کند؛ وارون سازی یک تابع تصادفی (با برد بزرگ) سخت است، دارای روابط ضربی نیست که به سادگی یافت شوند و در برابر برخورد مقاوم است. البته، این منطق غیررسمی، تمام حملات ممکن را منتفی نمی‌کند ولی اثبات امنیت ارائه شده در ذیل، چنین کاری انجام می‌دهد.

قبل از پرداختن به اثبات رسمی، تعدادی برداشت شهودی ارائه می‌کنیم. هدف ما اثبات این است که اگر مسئله RSA نسبت به $GenRSA$ سخت باشد، آنگاه $RSA - FDH$ زمانی امن است که H به عنوان یک پیشگوی تصادفی، مدل سازی شده باشد. در ابتداء، امنیت در برابر یک حمله‌ی بدون پیام را در نظر می‌گیریم، یعنی، زمانی که متخاصم $\mathcal{A}$ نمی‌تواند هیچ امضایی درخواست کند. در اینجا، متخاصم محدود به پرسمان از پیشگوی تصادفی است و بدون کاستن از کلیت، فرض می‌کنیم که $\mathcal{A}$ همواره دقیقاً q پرسمان (متمایز) از H می‌پرسد و اینکه اگر متخاصم یک جعل (m, σ) را خروجی دهد، آنگاه قبلاً m را از H پرسمان کرده است.

یک متخاصم کارا $\mathcal{A}$ را فرض کنید که یک حمله‌ی بدون پیام را اجرا می‌کند و دقیقاً q پرسمان از H می‌پرسد. یک الگوریتم کارای $\mathcal{A}'$ را می‌سازیم که مسئله RSA را نسبت به $GenRSA$ حل

دست می‌آورد. در نهایت، خروجی $\sigma = [\sigma_1 \cdot \sigma_2 \bmod N]$ را به عنوان یک امضای معتبر بر روی m ارائه می‌کند. این روش عمل می‌کند به این دلیل که

$$\sigma^e = (\sigma_1 \cdot \sigma_2)^e = (m_1^d \cdot m_2^d)^e = m_1^{ed} \cdot m_2^{ed} = m_1 \cdot m_2 = m \bmod N,$$

با استفاده از این واقعیت که σ_1, σ_2 امضاهای معتبری بر روی m_1, m_2 هستند.

داشتن توانایی جعل یک امضا بر روی یک پیام دلخواه، بسیار مخرب است. باین حال، می‌توان استدلال کرد که این حمله، واقع بینانه نیست به این دلیل که متخاصم قادر به قانع کردن یک امضا دهنده برای امضای پیام‌های m_1 و m_2 به شکل دقیق، نخواهد بود. باز هم، این امر تا جایی که تعریف ۱۲-۲ مطرح است، اهمیتی نخواهد داشت. به علاوه، در نظر گرفتن فرضیاتی در مورد اینکه امضاکننده مایل به امضای چه پیام‌هایی خواهد بود و مایل به امضای کدام نخواهد بود، خطرناک است. برای مثال، یک کارخواه ممکن است از یک طرح امضا برای احراز اصالت برای یک کارگذار با امضای یک چالش تصادفی ارسال شده توسط کارگذار، استفاده کند. در اینجا، یک کارگذار مخرب قادر به دریافت یک امضا بر روی هر پیام (یا پیام‌های) دلخواه خود خواهد بود. به شکل کلی‌تر، ممکن است متخاصم قادر باشد m_1 و m_2 را به عنوان پیام‌های «قانونی» انتخاب کند که امضاکننده با امضای آن‌ها موافقت کند. در نهایت، توجه کنید که این حمله را می‌توان تعمیم داد: اگر متخاصم امضاهای معتبر بر روی q پیام دلخواه $M = \{m_1, \dots, m_q\}$ را به دست آورد، آنگاه متخاصم می‌تواند یک امضای معتبر را بر روی هر کدام از $2^q - q$ پیام دیگر به دست آمده با محاسبه‌ی ضرب‌های زیرمجموعه‌های M (با اندازه‌ی متفاوت از ۱) اعلام نماید.

 $RSA - FDH$ - ۲-۴-۱۲ و $PKCS\#1 v2.1$

می‌توان با اعمال نوعی تبدیل به پیام‌ها قبل از امضای آن‌ها تلاش نمود که مانع حملات بیان شده در بخش قبلی شویم. یعنی، امضاکننده، یک تابع (قطعی) H با بعضی ویژگی‌های رمزنگاری خاص (توصیف شده در ذیل) نگاشت پیام‌ها به $\mathbb{Z}_N^*$ را به عنوان بخشی از کلید عمومی خود، مشخص خواهد نمود؛ امضاها بر روی یک پیام m عبارت خواهند بود از $\sigma = [H(m)^d \bmod N]$ و واریسی امضای σ بر روی پیام m بررسی $\sigma^e \stackrel{?}{=} H(m) \bmod N$ انجام خواهد شد. ن.ک. ساختار ۱۲-۶.

H باید چه ویژگی‌هایی داشته باشد تا این ساختار، امن باشد؟ به شکل حدافلی، به منظور جلوگیری از حمله‌ی بدون پیام، شروع با σ ، محاسبه‌ی $\hat{m} := [\sigma^e \bmod N]$ و سپس یافتن یک پیام m به صورتی که $H(m) = \hat{m}$ باشد، باید برای مهاجم، غیرعملی باشد. این امر علی‌الخصوص به این معنا است که وارون سازی H باید سخت باشد. برای جلوگیری از حمله‌ی دوم، به یک H نیاز داریم که «روابط ضرب» را نپذیرد، یعنی، برای آن، یافتن سه پیام m, m_1, m_2 به شکلی که $H(m) = H(m_1) \cdot H(m_2) \bmod N$ سخت باشد. در نهایت، یافتن برخوردها در H باید سخت باشد؛ اگر $H(m_1) = H(m_2)$ ، آنگاه m_1 و m_2 دارای امضای یکسان هستند و جعل بدیهی خواهد شد.

می‌کند. با داشتن ورودی (N, e, y) ، الگوریتم $\mathcal{A}'$ الگوریتم $\mathcal{A}$ را بر روی کلید عمومی $pk = (N, e)$ اجرا می‌کند. فرض کنید $m_1, \dots, m_q$ نشان‌دهنده q پرسمان (متمایزی) باشد که $\mathcal{A}$ از H می‌پرسد. الگوریتم $\mathcal{A}'$ ما، این پرسمان‌های $\mathcal{A}$ از پیشگوی تصادفی را با عناصر یکنواخت از $\mathbb{Z}_N^*$ پاسخ می‌دهد مگر برای یک پرسمان مثلاً، پرسمان i ام انتخاب‌شده به‌صورت یکنواخت از میان پرسمان‌های پیشگوی $\mathcal{A}$ - که با خود y پاسخ داده می‌شود. توجه کنید که از دید $\mathcal{A}$ تمام پرسمان‌های پیشگوی تصادفی آن با عناصر یکنواخت از $\mathbb{Z}_N^*$ پاسخ داده می‌شوند (یادآوری می‌کنیم که y هم یکنواخت است، هرچند که توسط $\mathcal{A}'$ انتخاب نشده است) و بنابراین، $\mathcal{A}$ هیچ اطلاعاتی در مورد i ندارد. به‌علاوه، دیدگاه $\mathcal{A}$ هنگام اجرا شدن به‌عنوان یک زیرروال توسط $\mathcal{A}'$ دارای توزیعی برابر با دیدگاه $\mathcal{A}$ هنگام حمله به طرح امضای اصلی است.

اگر $\mathcal{A}$ یک جعل (m, σ) را ارائه دهد، آنگاه به این دلیل که $m \in \{m_1, \dots, m_q\}$ ، به احتمال $1/q$ خواهیم داشت $m = m_i$ در این حالت،

$$\sigma^e = H(m) = H(m_i) = y \bmod N$$

و $\mathcal{A}'$ می‌تواند σ را به‌عنوان راه‌حلی برای نمونه‌ی RSA داده‌شده به آن، یعنی (N, e, y) خروجی دهد. نتیجه می‌گیریم که اگر $\mathcal{A}$ یک جعل را با احتمال ϵ خروجی دهد، آنگاه $\mathcal{A}'$ مسئله‌ی RSA را با احتمال ϵ/q حل می‌کند. از آنجاکه q چندجمله‌ای است، نتیجه می‌گیریم که ϵ باید ناچیز باشد اگر مسئله‌ی RSA نسبت به $GenRSA$ سخت باشد.

مدیریت حالتی که در آن، متخاصم اجازه‌ی درخواست امضاها بر روی پیام‌های انتخابی خود را دارد، دشوارتر است. پیچیدگی به این دلیل به وجود می‌آید که الگوریتم $\mathcal{A}'$ ما در بالا، نمای رمزگشایی d را نمی‌داند، با این حال، اکنون باید امضاها را بر روی پیام‌های پرسمان شده توسط $\mathcal{A}$ از پیشگوی امضای خود را محاسبه نماید. این امر به نظر غیرممکن (و شاید حتی متناقض!) می‌رسد تا اینکه درمی‌یابیم که $\mathcal{A}'$ «می‌تواند» به‌درستی یک امضا را بر روی یک پیام m محاسبه کند به شرطی که $H(m)$ را برابر با $[\sigma^e \bmod N]$ برای یک مقدار معلوم σ قرار دهد. (در اینجا از این واقعیت بهره می‌بریم که پیشگوی تصادفی، «قابل برنامه‌نویسی» است.) اگر σ یکنواخت باشد، آنگاه $[\sigma^e \bmod N]$ نیز یکنواخت خواهد بود و بنابراین پیشگوی تصادفی همچنان «به‌درستی» توسط $\mathcal{A}'$ شبیه‌سازی می‌شود.

برداشت شهودی بالا در اثبات قضیه‌ی زیر، به‌صورت رسمی بیان شده است:

قضیه‌ی ۱۲-۷

اگر مسئله‌ی RSA نسبت به $GenRSA$ سخت باشد و H به‌عنوان یک پیشگوی تصادفی مدل‌سازی شده باشد، آنگاه ساختار ۱۲-۶، امن است.

اثبات

فرض کنید $\Pi = (Gen, Sign, Vrfy)$ نشان‌دهنده‌ی ساختار ۱۲-۶ باشد و فرض کنید $\mathcal{A}$ یک متخاصم زمان چندجمله‌ای تصادفی باشد. بدون کاستن از کلیت، فرض می‌کنیم که اگر $\mathcal{A}$ یک امضا را بر روی یک پیام m درخواست کند یا یک جعل (m, σ) را خروجی دهد، آنگاه قبلاً m را از H پرسمان کرده است. فرض کنید $q(n)$ یک چندجمله‌ای تحت کران بالا بر روی تعداد پرسمان‌های انجام‌شده توسط $\mathcal{A}$ از H بر روی پارامتر امنیتی n باشد؛ بدون کاستن از کلیت، فرض می‌کنیم که $\mathcal{A}$ دقیقاً $q(n)$ پرسمان متمایز از H می‌پرسد.

برای راحتی بیشتر، گام‌های آزمایش $Sig - forge_{\mathcal{A}, \Pi}(n)$ را فهرست می‌کنیم:

- (۱) $GenRSA(1^n)$ اجرا می‌شود تا (N, e, d) به دست آید. یک تابع تصادفی $H: \{0, 1\}^* \rightarrow \mathbb{Z}_N^*$ انتخاب می‌شود.
- (۲) متخاصم $\mathcal{A}$ مقدار $pk = (N, e)$ را دریافت کرده و می‌تواند H و همین‌طور یک پیشگوی امضاکننده $Sign_{(N, d)}(\cdot)$ را مورد پرسمان قرار دهد که با دریافت یک پیام m به‌عنوان ورودی، $\sigma := [H(m)^d \bmod N]$ را باز می‌گرداند.
- (۳) متخاصم $\mathcal{A}$ خروجی (m, σ) را ارائه می‌کند که در آن، قبلاً یک امضا بر روی m درخواست نکرده است. خروجی آزمایش برابر با ۱ است اگر و تنها اگر $\sigma^e = H(m) \bmod N$.

یک آزمایش تعدیل‌شده‌ی $Sig - forge'_{\mathcal{A}, \Pi}(n)$ را تعریف می‌کنیم که در آن، یک حدس در ابتدا بر این مبنا زده می‌شود که کدام پیام (از میان q پیامی که $\mathcal{A}$ از H پرسمان می‌کند) متناظر با جعل نهایی (در صورت وجود) اعلام شده توسط $\mathcal{A}$ است:

- (۱) $j \in \{1, \dots, q\}$ ز یکنواخت را انتخاب می‌کنیم.
- (۲) $GenRSA(1^n)$ را اجرا می‌کنیم تا (N, e, d) به دست آید. یک تابع تصادفی $H: \{0, 1\}^* \rightarrow \mathbb{Z}_N^*$ انتخاب می‌شود.
- (۳) متخاصم $\mathcal{A}$ مقدار $pk = (N, e)$ را دریافت می‌کند و می‌تواند H و همین‌طور یک پیشگوی امضاکننده $Sign_{(N, d)}(\cdot)$ را مورد پرسمان قرار دهد که با دریافت یک پیام m به‌عنوان ورودی، $\sigma := [H(m)^d \bmod N]$ را باز می‌گرداند.
- (۴) متخاصم $\mathcal{A}$ خروجی (m, σ) را ارائه می‌کند که در آن، قبلاً یک امضا بر روی m درخواست نکرده است. فرض کنید i به‌گونه‌ای باشد که $m = m_i$ باشد.^{۱۱} خروجی آزمایش برابر با ۱ است اگر و تنها اگر $\sigma^e = H(m) \bmod N$ و $j = i$ باشد.

^{۱۱} در اینجا، m_i نشان‌دهنده‌ی پرسمان i ام پرسیده شده از H است. یادآوری می‌کنیم که بر اساس فرضی اگر $\mathcal{A}$ یک امضا بر روی یک پیام m را درخواست کند، آنگاه باید قبلاً m را از H پرسمان کرده باشد.

۴ در انتهای اجرای $\mathcal{A}$ ، این الگوریتم، خروجی (m, σ) را می‌دهد. اگر $m = m_j$ و $\sigma = y \bmod N$ ، آنگاه σ را به عنوان خروجی ارائه می‌کنیم.

مختص است که $\mathcal{A}'$ در زمان چندجمله‌ای تصادفی اجرا می‌شود. فرض کنید ورودی (N, e, y) برای $\mathcal{A}'$ یا اجرای $GenRSA(1^n)$ برای به دست آوردن (N, e, d) و سپس انتخاب $y \in \mathbb{Z}_N^*$ یکنواخت، تولید می‌شود. مشاهده‌ی مهم آن است که دیدگاه $\mathcal{A}$ هنگام اجرا به عنوان یک زیرروال توسط $\mathcal{A}'$ پیرایه دیدگاه $\mathcal{A}$ در آزمایش $Sig - forge''_{\mathcal{A}, \Pi}(n)$ است. علی‌الخصوص، تمام پرسمان‌های پیشگوی $Sign$ به درستی پاسخ داده می‌شوند و هر کدام از پرسمان‌های پیشگوی تصادفی $\mathcal{A}$ هنگام برداشتن به عنوان یک زیرروال توسط $\mathcal{A}'$ با یک عنصر یکنواخت از $\mathbb{Z}_N^*$ پاسخ داده می‌شود:

- پرسمان $H(m_j)$ با y پاسخ داده می‌شود که یک عنصر یکنواخت از $\mathbb{Z}_N^*$ است.
- پرسمان‌های $H(m_i)$ با $i \neq j$ با $y_i = [\sigma_i^e \bmod N]$ پاسخ داده می‌شوند که در آن، σ_i در $\mathbb{Z}_N^*$ یکنواخت است. از آنجاکه توان رسانی تا توان e یک تابع یک به یک است، y_i نیز دارای توزیع یکنواخت است.

در نهایت، توجه کنید که هرگاه آزمایش $Sig - forge''_{\mathcal{A}, \Pi}(n)$ خروجی ۱ را ارائه کند، آنگاه $\mathcal{A}'$ یک راه حل درست برای نمونه‌ی داده شده‌ی RSA خروجی می‌دهد. این امر به این دلیل نتیجه می‌شود که $Sig - forge''_{\mathcal{A}, \Pi}(n) = 1$ به این معنا است که $i = j$ و $\sigma = H(m_i) \bmod N$ و اکنون، زمانی که $i = j$ باشد، الگوریتم $\mathcal{A}'$ متوقف نمی‌شود و به علاوه، $H(m_i) = y$ است. بنابراین، $\sigma^e = y \bmod N$ و بنابراین، σ همان وارون مطلوب است. با استفاده از معادله‌ی (۱۲.۱)، این امر به این معنا است که

$$\Pr[RSA - inv_{\mathcal{A}', GenRSA}(n) = 1] = \Pr[Sig - forge''_{\mathcal{A}, \Pi}(n) = 1] = \frac{\Pr[Sig - forge_{\mathcal{A}, \Pi}(n) = 1]}{q(n)}. \quad (12.2)$$

اگر مسئله‌ی RSA نسبت به $GenRSA$ سخت باشد، یک تابع ناچیز $negl$ وجود خواهد داشت به صورتی که $\Pr[RSA - inv_{\mathcal{A}', GenRSA}(n) = 1] \leq negl(n)$. از آنجاکه $q(n)$ چندجمله‌ای است، از معادله‌ی (۱۲.۲) نتیجه می‌گیریم که $\Pr[Sig - forge_{\mathcal{A}, \Pi}(n) = 1]$ نیز ناچیز است. این امر، اثبات را کامل می‌کند. ■

$RSA PKCS\#1 v2.1$

استاندارد $RSA PKCS\#1 v2.1$ شامل یک طرح امضا است که می‌توان آن را نسخه‌ای از $RSA - FDH$ در نظر گرفت. به شکلی دقیق‌تر، این استاندارد، طرحی را تعریف می‌کند که در آن، امضا بر روی یک پیام به یک «چاشنی» (یعنی یک مقدار تصادفی) انتخاب شده توسط امضاکننده در ابتدای

از آنجاکه j یکنواخت و مستقل از هر چیز دیگری است، احتمال اینکه $j = i$ (حتی مشروط به این رخداد که $\mathcal{A}$ یک جعل را خروجی دهد) دقیقاً برابر با $1/q$ است. بنابراین، $\Pr[Sig - forge'_{\mathcal{A}, \Pi}(n) = 1] = \frac{1}{q(n)} \cdot \Pr[Sig - forge_{\mathcal{A}, \Pi}(n) = 1]$

اکنون آزمایش تعدیل شده‌ی $Sig - forge''_{\mathcal{A}, \Pi}(n)$ را در نظر بگیرید که در آن، آزمایش متوقف می‌شود اگر $\mathcal{A}$ دست به درخواست یک امضا بر روی پیام m_j بزند (که در آن، m_j نشان‌دهنده‌ی پیام زام پرسمان شده از H است و j عبارت است از مقدار یکنواخت انتخاب شده در ابتدای آزمایش). این امر تغییر در این احتمال که خروجی آزمایش برابر با ۱ باشد ایجاد نخواهد کرد چراکه اگر $\mathcal{A}$ دست به درخواست یک امضا بر روی m_j بزند، آنگاه برای آن، امکان اعلام یک جعل بر روی m_j وجود ندارد. به عبارتی

$$\Pr[Sig - forge''_{\mathcal{A}, \Pi}(n) = 1] = \Pr[Sig - forge'_{\mathcal{A}, \Pi}(n) = 1] = \frac{\Pr[Sig - forge_{\mathcal{A}, \Pi}(n) = 1]}{q(n)}. \quad (12.1)$$

در نهایت، الگوریتم $\mathcal{A}'$ زیر که مسئله‌ی RSA را حل می‌کند، در نظر بگیرید:

الگوریتم $\mathcal{A}'$

این الگوریتم به عنوان ورودی (N, e, y) را دریافت می‌کند.

(۱) $\{1, \dots, q\}$ یکنواخت را انتخاب می‌کنیم.

(۲) با دریافت کلید عمومی $pk = \langle N, e \rangle$ را اجرا می‌کنیم. سه گانه‌های (m_i, σ_i, y_i) را در یک جدول که در ابتدا خالی است، ذخیره می‌کنیم. یک مدخل (m_i, σ_i, y_i) نشان می‌دهد که $\mathcal{A}'$ قرار داده است $H(m_i) = y_i$ و $\sigma_i^e = y_i \bmod N$.

(۳) هنگامی که $\mathcal{A}$ پرسمان نام پیشگوی تصادفی خود، یعنی $H(m_i)$ را انجام می‌دهد، آن را به صورت زیر پاسخ می‌دهیم:

- اگر $i = j$ ، y را به عنوان پاسخ به پرسمان بازمی‌گردانیم.
- در غیر این صورت، $\sigma_i \in \mathbb{Z}_N^*$ یکنواخت را انتخاب می‌کنیم، $y_i = [\sigma_i^e \bmod N]$ را محاسبه می‌کنیم، y_i را به عنوان پاسخ به پرسمان بازمی‌گردانیم و (m_i, σ_i, y_i) را در جدول ذخیره می‌کنیم.
- هنگامی که $\mathcal{A}$ یک امضا بر روی پیام m درخواست می‌کند، فرض می‌کنیم i به صورتی باشد که $m = m_i$ و پرسمان را به شکل زیر پاسخ می‌دهیم:
- اگر $j = i$ باشد، آنگاه $\mathcal{A}'$ متوقف می‌شود.
- اگر $j \neq i$ ، آنگاه یک مدخل (m_i, σ_i, y_i) در جدول وجود خواهد داشت. σ_i را به عنوان پاسخ پرسمان ارائه می‌کنیم.

تولید امضا، بستگی دارد. اگر این چاشنی توسط امضاکننده برابر با $NULL$ قرار داده شود-کاری که این استاندارد اجازه می‌دهد- طرح به‌دست‌آمده بسیار مشابه با $RSA - FDH$ خواهد بود.

ضروری است که برد H برابر با (نزدیک به) کل $\mathbb{Z}_N$ باشد؛ علی‌الخصوص، این که اجازه دهیم H یک تابع چکیده ساز رمزنگاری «تولید انبوه» مثل $SHA - 1$ باشد، کافی نخواهد بود. (طول خروجی $SHA - 1$ بسیار کوچک‌تر از طول پیمانه‌های RSA استفاده‌شده در عمل، است). درواقع، حملات «عملی» علیه ساختار ۱۲-۶ ایجاد شده‌اند اگر طول H بیش‌ازحد کوتاه باشد (برای مثال، اگر طول خروجی برابر با ۱۶۰ بیت باشد که در حالت استفاده‌ی مستقیم از $SHA - 1$ به‌عنوان H به وجود می‌آید). در طرح امضای $PKCS\#1 v2.1$ ، H از طریق اعمال مکرر یک تابع چکیده ساز رمزنگاری مینا، ساخته می‌شود.

۱۲-۵- امضاها از روی مسئله‌ی لگاریتم گسسته

طرح‌های امضا می‌توانند بر اساس فرض لگاریتم گسسته هم باشند، هرچند این فرض به‌سادگی فرض RSA برای امضاها قابل استفاده نیست. در بخش ۱۲-۵-۱، یک طرح امضای معرفی‌شده توسط کلاوس اشنور را توصیف می‌کنیم که می‌توان امنیت آن را در مدل پیشگوی تصادفی اثبات نمود. در بخش ۱۲-۵-۲، طرح‌های امضای DSA و $ECDSA$ را توصیف می‌کنیم که به‌صورت گسترده مورد استفاده قرار می‌گیرند.

۱۲-۵-۱- طرح امضای اشنور

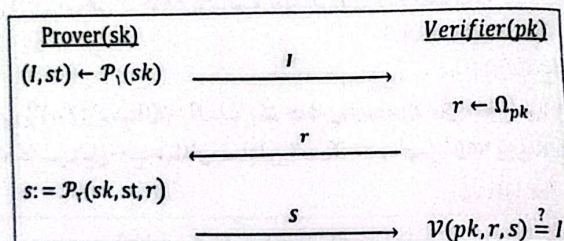
اگر بخواهیم برداشت شهودی مینا برای طرح امضای اشنور را به‌خوبی توضیح دهیم باید ابتدا به‌صورت مختصر «طرح‌های شناسایی» (کلید عمومی) را مورد بحث قرار دهیم. در ادامه، «تبدیل فیات-شامیر» را توصیف می‌کنیم که می‌توان از آن برای تبدیل طرح‌های شناسایی به طرح‌های امضا در مدل پیشگوی تصادفی استفاده نمود. در نهایت، طرح شناسایی اشنور و طرح امضای متناظر با آن- را بر اساس مسئله‌ی لگاریتم گسسته، ارائه می‌کنیم.

طرح‌های شناسایی

یک طرح شناسایی یک پروتکل تعاملی است که به یکی از طرفین اجازه می‌دهد تا هویت خود را برای طرف دیگر اثبات کند (یعنی خود را احراز اصالت نماید). این مقوله یک مفهوم بسیار طبیعی است و امروزه هنگام ورود به یک وب‌سایت معمولاً باید خود را احراز اصالت نمایید. ما طرف شناسایی‌کننده‌ی خود (برای مثال، کاربر) را «اثبات‌کننده» (*Prover*) و طرف واریسی‌کننده‌ی هویت (برای مثال، کارگذار وب) را «واریسی‌کننده» (*Verifier*) می‌نامیم. در اینجا، ما علاقه‌مند به موقعیت کلید عمومی هستیم که در آن، اثبات‌کننده و واریسی‌کننده هیچ اطلاعات محرمانه‌ای (از قبیل گذروژه) را از قبل باهم تسهیم نمی‌کنند؛ درواقع، واریسی‌کننده تنها کلید عمومی اثبات‌کننده را می‌داند. اجرای موفق

پروتکل شناسایی، واریسی‌کننده را قانع می‌کند که در حال ارتباط با اثبات‌کننده‌ی هدف است و نه یک دشمن.

پروتکل‌های شناسایی سه-دور با یک شکل خاص را در نظر می‌گیریم که در آن‌ها، اثبات‌کننده به دو الگوریتم P_1, P_2 و طرف واریسی‌کننده پروتکل توسط یک الگوریتم V مشخص می‌شود. اثبات‌کننده، $P_1(sk)$ را با استفاده از کلید خصوصی sk خود اجرا می‌کند تا یک پیام اولیه‌ی I را به طرف واریسی‌کننده st به دست آورد و پروتکل را با ارسال I به واریسی‌کننده، آغاز می‌کند. در پاسخ، واریسی‌کننده یک چالش r انتخاب‌شده به‌صورت یکنواخت از یک مجموعه‌ی Ω_{pk} تعریف‌شده توسط کلید عمومی pk اثبات‌کننده را ارسال می‌کند. در ادامه، اثبات‌کننده، $P_2(sk, st, r)$ را اجرا می‌کند تا یک پاسخ s را محاسبه کند که برای واریسی‌کننده ارسال خواهد نمود. در نهایت، واریسی‌کننده، $V(pk, r, s)$ را محاسبه کرده و پذیرش را انجام می‌دهد اگر و تنها اگر این محاسبه منجر به پیام بلی شود؛ ن.ک. تصویر ۱۲-۱. البته، برای درستی محاسبات، ملزم می‌کنیم که اگر اثبات‌کننده‌ی فیک، پروتکل را به‌درستی اجرا کند، آنگاه واریسی‌کننده باید همواره پذیرش را انجام دهد.



تصویر ۱۲-۱: یک طرح شناسایی سه-دور

دلایل فنی، طرح‌های شناسایی را فرض می‌کنیم که «زوال‌ناپذیر» باشند، که به شکل شهودی به این معناست که تعداد زیادی پیام اولیه‌ی I ممکن وجود دارد و هیچ کدام دارای احتمال بالایی برای بل شدن نیستند. به شکل رسمی، یک طرح، «زوال‌ناپذیر» است اگر برای هر کلید خصوصی sk و پیام اولیه‌ی ثابت I ، احتمال اینکه $P_1(sk)$ دارای خروجی I باشد، ناچیز است. (هر طرح شناسایی می‌توان به شکل بدیهی تغییر داد تا غیر تبهگن باشد به این صورت که یک رشته‌ی n -بیتی یکنواخت را به همراه پیام اولیه ارسال نماید).

از جنبه‌ی اساسی یک طرح شناسایی آن است که مشخص می‌کند کلید رمز اثبات‌کننده را نمی‌داند و قادر به فریب دادن واریسی‌کننده برای پذیرش باشد. این مسئله باید برقرار باشد حتی اگر پیام قادر به شوند غیرفعال بر روی چندین اجرای (صادقانه‌ی) پروتکل بین اثبات‌کننده و واریسی‌کننده باشد. ما چنین شندهایی را از طریق پیشگوی $Trans_{sk}$ رسمی می‌کنیم که هنگام فراخوانی

بدون ورودی، یک اجرای صادقانه از پروتکل را اجرا کرده و کل رونوشت (I, r, s) آن تکرار را برای متخاصم باز می گرداند.

فرض کنید $\Pi = (\text{Gen}, \mathcal{P}_1, \mathcal{P}_2, \mathcal{V})$ یک طرح شناسایی باشد و آزمایش زیر را برای یک متخاصم $\mathcal{A}$ و پارامتر n در نظر بگیرید:

آزمایش شناسایی $\text{Ident}_{\mathcal{A}, \Pi}(n)$:

- (۱) $\text{Gen}(1^n)$ اجرا می شود تا کلیدهای (pk, sk) به دست آیند.
- (۲) متخاصم $\mathcal{A}$ مقدار pk و دسترسی به یک پیشگوی Trans_{sk} را دریافت می کند که به هر تعداد که می خواهد می تواند آن را پرسمان نماید.
- (۳) در هر نقطه طی آزمایش، $\mathcal{A}$ یک پیام I را خروجی می دهد. یک چالش یکنواخت $r \in \Omega_{pk}$ انتخاب شده و به $\mathcal{A}$ داده می شود که با یک مقدار s به آن پاسخ خواهد داد. ($\mathcal{A}$ ممکن است همچنان به پرسمان کردن Trans_{sk} ادامه دهد حتی بعد از دریافت r).
- (۴) آزمایش دارای خروجی ۱ خواهد بود اگر و تنها اگر $\mathcal{V}(pk, r, s) = 1$.

تعریف ۱۲-۸

یک طرح شناسایی $\Pi = (\text{Gen}, \mathcal{P}_1, \mathcal{P}_2, \mathcal{V})$ علیه یک حمله ی غیرفعال، امن است یا تنها «امن» است اگر برای تمام متخاصمین زمان-چندجمله ای تصادفی $\mathcal{A}$ ، یک تابع ناچیز negl وجود داشته باشد به صورتی که

$$\Pr[\text{Ident}_{\mathcal{A}, \Pi}(n) = 1] \leq \text{negl}(n).$$

همچنین می توان مفاهیم قوی تری برای امنیت در نظر گرفت، مثلاً، حالتی که در آن، متخاصم همچنین می تواند حملات «فعال» علیه پروتکل را با جعل هویت یک واریسی کننده و احتمالاً ارسال مقادیر r انتخاب شده به شکل مخرب، اجرا نماید. برای کاربرد خود برای طرح های امضا به این مفهوم نیازی نداریم.

از طرح های شناسایی تا امضاها

تبدیل فیات-شامیر (ساختار ۱۲-۹) روشی برای تبدیل هر طرح شناسایی (تعاملی) به یک طرح امضای (غیرتعاملی) را می دهد. ایده ی اصلی این است که امضاکننده به عنوان یک اثبات کننده عمل کند و پروتکل شناسایی را به تنهایی اجرا نماید. یعنی، به منظور امضای یک پیام m ، امضاکننده در ابتدا I را محاسبه می کند و در ادامه چالش r را با اعمال یک تابع مفروض H بر I و m تولید می نماید. سپس، پاسخ صحیح s را به دست می آورد. امضا بر روی m برابر با (r, s) است که می تواند با (۱) محاسبه ی $I := \mathcal{V}(pk, r, s)$ و سپس (۲) بررسی اینکه $I = H(I, m)$ است، واریسی شود.

فرض کنید $\Pi = (\text{Gen}_{id}, \mathcal{P}_1, \mathcal{P}_2, \mathcal{V})$ یک طرح شناسایی باشد و یک طرح امضا را به صورت زیر ایجاد

Gen : با دریافت 1^n به عنوان ورودی، $\text{Gen}_{id}(1^n)$ را اجرا می کنیم تا کلیدهای pk, sk به دست آیند. کلید عمومی pk مجموعه ای از چالش ها Ω_{pk} را مشخص می کند. به عنوان بخشی از تولید کلید، یک تابع $H: \{0, 1\}^* \rightarrow \Omega_{pk}$ مشخص می شود ولی ما این امر را به صورت ضمنی باقی می گذاریم.

Sign : با دریافت یک کلید خصوصی sk و یک پیام $m \in \{0, 1\}^*$ به عنوان ورودی، موارد زیر را انجام می دهیم:

(۱) محاسبه ی $(I, st) \leftarrow \mathcal{P}_1(sk)$.

(۲) محاسبه ی $r := H(I, m)$.

(۳) محاسبه ی $s := \mathcal{P}_2(sk, st, r)$.

امضای (r, s) را اعلام می کنیم.

Verify : با دریافت یک کلید عمومی pk ، یک پیام m و یک امضای (r, s) به عنوان ورودی، $I := \mathcal{V}(pk, r, s)$ را محاسبه کرده و خروجی ۱ را ارائه می کنیم اگر و تنها اگر $H(I, m) = r$ باشد.

امضای (r, s) به یک پیام خاص m «پیوند زده شده است» به این دلیل که r تابعی از I و m است. بنابراین، تغییر m منجر به یک r کاملاً متفاوت خواهد شد. اگر H به عنوان یک پیشگوی یکنواخت مدلسازی شده باشد که ورودی های را به صورت یکنواخت بر Ω_{pk} نگاشت می کند، آنگاه چالش r یکنواخت خواهد بود. به شکل شهودی، برای یک متخاصم (که sk را نمی داند) یافتن یک امضای معتبر (r, s) بر روی یک پیام m به همان اندازه دشوار خواهد بود که جعل هویت اثبات کننده یک اجرای صادقانه از پروتکل، دشوار است. این برداشت شهودی در اثبات قضیه ی زیر به صورت رسمی بیان می شود.

فرض کنید Π یک طرح شناسایی باشد و فرض کنید Π' طرح امضایی باشد که با اعمال تبدیل فیات-شامیر بر آن، به دست می آید. اگر Π امن باشد و H به عنوان یک پیشگوی تصادفی مدلسازی شده باشد، آنگاه Π' امن است.

پایان می‌دهد با امضاهای معتبر دارای توزیع درست، پاسخ داده می‌شوند. تنها تفاوت بین دیدگاه‌ها آن است که زمانی که $\mathcal{A}'$ به‌عنوان یک زیرروال توسط $\mathcal{A}$ اجرا می‌شود، این امکان وجود دارد که یک عدم سازگاری در پاسخ‌های دریافتی $\mathcal{A}'$ به پرسمان‌های خود از H وجود داشته باشد؛ به شکل خاص، این اتفاق رخ خواهد داد اگر $\mathcal{A}$ یک پرسمان امضا برای یک پیام m را با استفاده از یک رونوشت (l, r, s) پاسخ دهد که برای آن، $H(l, m)$ از قبل تعریف شده است (یعنی، $\mathcal{A}'$ قبلاً (l, m) را از H پرسمان کرده است) و $H(l, m) \neq r$ ، با این حال، اگر Π غیر تبه‌گن باشد، آنگاه این اتفاق تنها ممکن است با احتمال ناچیز رخ دهد. بنابراین، احتمال اینکه $\mathcal{A}'$ یک جعل را هنگام اجرا به‌عنوان یک زیرروال توسط $\mathcal{A}$ خروجی دهد برابر با $\text{negl}(n) - \text{Sig-forge}_{\mathcal{A}', \Pi'}(n)$ برای یک تابع ناچیز negl است.

یک اجرا از آزمایش $\text{Ident}_{\mathcal{A}, \Pi}(n)$ را در نظر بگیرید که در آن، $\mathcal{A}'$ یک امضای جعل‌شده (r, s) را بر روی یک پیام m خروجی می‌دهد و قرار می‌دهد $J := \mathcal{V}(pk, r, s)$ ؛ از آنجاکه J یکنواخت و مستقل از هر چیز دیگری است، احتمال اینکه $(l, m) = (l_r, m_r)$ باشد (حتی مشروط به این رخداد که $\mathcal{A}'$ یک جعل را خروجی دهد) دقیقاً برابر با $1/q$ است. (یادآوری می‌کنیم که فرض کرده ایم اگر $\mathcal{A}'$ یک امضای جعل‌شده (r, s) را بر روی یک پیام m با $\mathcal{V}(pk, r, s) = l$ خروجی دهد، آنگاه $\mathcal{A}'$ قبلاً (l, m) را پرسمان کرده است.) هنگامی که هر دو رخداد اتفاق بیفتند، $\mathcal{A}$ با موفقیت هویت اثبات‌کننده را جعل می‌کند. درواقع، $\mathcal{A}$ مقدار l_r را به‌عنوان پیام اولیه‌ی خود ارسال می‌کند، در پاسخ یک چالش r را دریافت می‌کند و با s پاسخ می‌دهد. ولی $H(l_r, m_r) = r$ و (از آنجاکه امضای جعل شده، معتبر است) $\mathcal{V}(pk, r, s) = l$ است. با قرار دادن همه‌ی این موارد در کنار هم، مشاهده می‌کنیم که

$$\Pr[\text{Ident}_{\mathcal{A}, \Pi}(n) = 1] \geq \frac{1}{q(n)} \cdot (\Pr[\text{Sig-forge}_{\mathcal{A}', \Pi'}(n) = 1] - \text{negl}(n))$$

با

$$\Pr[\text{Sig-forge}_{\mathcal{A}', \Pi'}(n) = 1] \leq q(n) \cdot \Pr[\text{Ident}_{\mathcal{A}, \Pi}(n) = 1] + \text{negl}(n).$$

اگر Π امن باشد، آنگاه $\Pr[\text{Ident}_{\mathcal{A}, \Pi}(n) = 1]$ ناچیز است؛ از آنجاکه $q(n)$ چندجمله‌ای است، این امر به شکل ضمنی نشان می‌دهد که $\Pr[\text{Sig-forge}_{\mathcal{A}', \Pi'}(n) = 1]$ نیز ناچیز است. به این دلیل که $\mathcal{A}'$ دلخواه بود، این امر به این معنا است که Π' امن است. ■

طرح شناسایی اشنور

طرح شناسایی اشنور بر اساس سختی مسئله‌ی لگاریتم گسسته است. فرض کنید G یک الگوریتم زمان-چندجمله‌ای باشد که به‌عنوان ورودی، 1^n را دریافت می‌کند و (مگر با احتمال ناچیز) یک توصیف از یک گروه دوری G ، مرتبه‌ی آن، q ، و $(\|g\| = n)$ و یک مولد g را به‌عنوان خروجی ارائه

بنابراین، ما به مهاجمی کاهش داده‌ایم که یک کلید عمومی γ را دریافت می‌کند، یک پیام اولیه I را ارسال می‌کند، در پاسخ یک چالش یکنواخت r را دریافت می‌کند و در ادامه باید یک پاسخ s را ارسال کند که برای آن، $g^s \cdot y^{-r} = I$ باشد. به شکل غیررسمی، اگر مهاجم قادر به انجام این کار با احتمال بالا باشد، آنگاه باید به صورت خاص قادر به محاسبه‌ی پاسخ‌های درست s_1, s_2 برای حداقل دو چالش متفاوت $r_1, r_2 \in \mathbb{Z}_q$ باشد. توجه کنید که

$$g^{s_1} \cdot y^{-r_1} = I = g^{s_2} \cdot y^{-r_2},$$

و بنابراین، $g^{s_1 - s_2} = y^{r_1 - r_2}$. ولی این امر به شکل ضمنی نشان می‌دهد که مهاجم (که یادآوری می‌کنیم قادر به تولید s_1 در پاسخ به r_1 و s_2 در پاسخ به r_2 است) به صورت ضمنی می‌تواند لگاریتم گسسته‌ی زیر را محاسبه کند:

$$\log_g y = [(s_1 - s_2) \cdot (r_1 - r_2)^{-1} \bmod q],$$

که سختی فرض شده‌ی مسئله‌ی لگاریتم گسسته را نقض می‌کند.

فصلیه ۱۱-۱۲

اگر مسئله‌ی لگاریتم گسسته نسبت به G سخت باشد، آنگاه طرح شناسایی اشنور امن است.

اثبات

فرض کنید Π نشان‌دهنده‌ی طرح شناسایی اشنور و $\mathcal{A}$ یک متخاصم PPT حمله‌کننده به این طرح باشد. ما الگوریتم PPT زیر یعنی $\mathcal{A}'$ را می‌سازیم که مسئله‌ی لگاریتم گسسته را نسبت به G حل می‌کند:

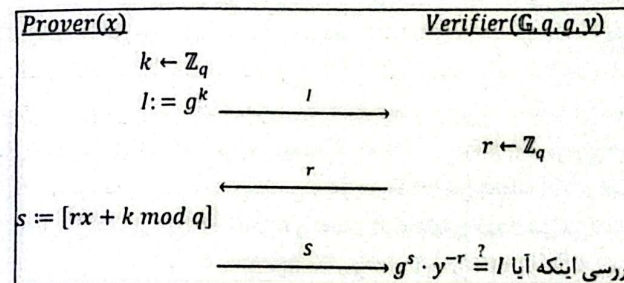
الگوریتم $\mathcal{A}'$

- این الگوریتم، G, q, g, γ را به عنوان ورودی دریافت می‌کند.
- (۱) $\mathcal{A}(pk)$ را اجرا می‌کنیم. تمام پرسمان‌های آن از $Trans_{sk}$ را به صورت توصیف شده در برداشت شهودی داده شده‌ی قبلی، پاسخ می‌دهیم.
- (۲) هنگامی که $\mathcal{A}$ به عنوان خروجی I را ارائه می‌کند، یک $r_1 \in \mathbb{Z}_q$ یکنواخت را به عنوان چالش انتخاب می‌کنیم. r_1 را به $\mathcal{A}$ می‌دهیم که در پاسخ، s_1 را ارائه می‌کند.
- (۳) $\mathcal{A}(pk)$ را دوباره (از اول) با همان مقدار تصادفی قبلی اجرا می‌کنیم به غیر از اینکه این بار برای $r_2 \in \mathbb{Z}_q$ یکنواخت و مستقل اجرا می‌شود. در نهایت، $\mathcal{A}$ با s_2 پاسخ می‌دهد.
- (۴) اگر $r_1 \neq r_2$ و $g^{s_1} \cdot h^{-r_1} = I$ و $g^{s_2} \cdot h^{-r_2} = I$ ، آنگاه به عنوان خروجی، $[(s_1 - s_2) \cdot (r_1 - r_2)^{-1} \bmod q]$ را ارائه می‌کنیم. در غیر این صورت، خروجی ارائه نمی‌کنیم.

می‌کند. به منظور تولید کلیدهای خود، اثبات‌کننده، $G(1^n)$ را اجرا می‌کند تا (G, q, g) را به دست آورد. یک $x \in \mathbb{Z}_q$ یکنواخت را انتخاب می‌کند و قرار می‌دهد $y := g^x$ ؛ کلید عمومی برابر با (G, q, g, y) و کلید خصوصی برابر با x است. به منظور اجرای پروتکل (ن.ک. تصویر ۱۲-۲)، اثبات‌کننده با انتخاب یک $k \in \mathbb{Z}_q^*$ یکنواخت و قرار دادن $I := g^k$ ، فرآیند را آغاز می‌کند؛ I را به عنوان پیام اولیه، ارسال می‌کند. واریسی کننده یک چالش یکنواخت $r \in \mathbb{Z}_q$ را انتخاب و ارسال می‌کند؛ در پاسخ، اثبات‌کننده، $s := [rx + k \bmod q]$ را محاسبه می‌کند. واریسی کننده پذیرش را انجام می‌دهد اگر و تنها اگر $g^s \cdot y^{-r} \stackrel{?}{=} I$ درستی برقرار است به این دلیل که

$$g^s \cdot y^{-r} = g^{rx+k} \cdot (g^x)^{-r} = g^k = I.$$

توجه کنید که I در G یکنواخت است و بنابراین، این طرح، غیر تبهگن است.



تصویر ۱۲-۲: یک اجرا از طرح شناسایی اشنور

قبل از ارائه‌ی اثبات، برداشت شهودی سطح بالا را ارائه می‌کنیم. اولین برداشت مهم این است که شنود غیرفعال هیچ کمکی به مهاجم نمی‌کند. دلیل این امر آن است که مهاجم می‌تواند رونوشت‌های اجرای صادقانه را به تنهایی و تنها بر اساس کلید عمومی و بدون دانستن کلید خصوصی، «شبیه‌سازی» کند. برای انجام این کار، مهاجم تنها ترتیب گام‌ها را وارون می‌کند: در ابتدا، $r, s \in \mathbb{Z}_q$ یکنواخت و مستقل را انتخاب می‌کند و سپس قرار می‌دهد $I := g^s \cdot y^{-r}$. در یک رونوشت صادقانه (I, r, s) ، پیام اولیه I یک عنصر یکنواخت از G است، چالش یک عنصر یکنواخت مستقل از $\mathbb{Z}_q$ است و در نتیجه، s به صورت منحصر به فرد به صورت $s = \log_g(I \cdot y^r)$ تعیین می‌شود.

رونوشت‌های شبیه‌سازی شده‌ی ساخته شده توسط یک مهاجم دارای توزیع یکسان هستند: $r \in \mathbb{Z}_q$ یکنواخت است و به این دلیل که s در $\mathbb{Z}_q$ یکنواخت بوده و از r مستقل است، مشاهده می‌کنیم که I در G یکنواخت بوده و مستقل از r است. در نهایت، s به شکل منحصر به فرد به صورتی تعیین می‌شود که در محدودیت بالا صدق کند. به دلیل این امر، می‌توانیم در اصل فرض کنیم که هنگام حمله به طرح شناسایی، مهاجم اصلاً اجرای صادقانه را شنود نمی‌کند.

طرح امضای اشنور با اعمال تبدیل فیات-شامیر بر طرح شناسایی اشنور به دست می‌آید. ن.ک. ساختار ۱۲-۱۲.

طرح امضای اشنور

ساختار ۱۲-۱۲

فرض کنید G به صورت توصیف شده در متن باشد.

- $Gen: G(1^n)$ را اجرا می‌کنیم تا (G, q, g) به دست آید. یک $x \in \mathbb{Z}_q$ یکنواخت را انتخاب کرده و قرار می‌دهیم $y = g^x$. کلید خصوصی برابر با x و کلید عمومی برابر با (G, q, g, y) است. به عنوان بخشی از تولید کلید، یک تابع $H: \{0, 1\}^* \rightarrow \mathbb{Z}_q$ انتخاب می‌شود ولی ما این امر را به صورت ضمنی باقی می‌گذاریم.
- $Sign$: با دریافت یک کلید خصوصی x و یک پیام $m \in \{0, 1\}^*$ به عنوان ورودی، $k \in \mathbb{Z}_q$ یکنواخت را انتخاب کرده و قرار می‌دهیم $J = g^k$. آنگاه، $r = H(I, m)$ و سپس $s = [rx + k \bmod q]$ را محاسبه می‌کنیم. امضای (r, s) را به عنوان خروجی ارائه می‌کنیم.
- $Verify$: با دریافت یک کلید عمومی (G, q, g, y) ، یک پیام m و یک امضای (r, s) به عنوان ورودی، $I = g^s \cdot y^{-r}$ را محاسبه کرده و خروجی ۱ را ارائه می‌کنیم اگر $H(I, m) \stackrel{?}{=} r$.

ECDSA و DSA - ۲-۵-۱۲

«الگوریتم امضای دیجیتال» (DSA) و «الگوریتم امضای دیجیتال خم بیضوی» ($ECDSA$) بر اساس مسئله‌ی لگاریتم گسسته در دسته‌های مختلفی از گروه‌ها هستند. این الگوریتم‌ها به شکل‌های مختلف از سال ۱۹۹۱ وجود داشته‌اند و هر دو در «استاندارد امضای دیجیتال» (DSS) کنونی صادر شده توسط $NIST$ قرار گرفته‌اند.

هر دو طرح از یک قالب مشترک تبعیت می‌کنند و می‌توان آن را ساخته شده از یک طرح شناسایی مبنای یکتا (ن.ک. بخش قبلی) در نظر گرفت. فرض کنید G یک گروه دوری از مرتبه‌ی اول q با مولد g باشد. طرح شناسایی زیر را در نظر بگیرید که در آن، کلید خصوصی اثبات‌کننده برابر با x و کلید عمومی برابر با (G, q, g, y) با $y = g^x$ است:

- (۱) اثبات‌کننده، $k \in \mathbb{Z}_q^*$ یکنواخت را انتخاب کرده و $I = g^k$ را ارسال می‌کند.
- (۲) واریسی‌کننده، $\alpha, r \in \mathbb{Z}_q$ یکنواخت را انتخاب کرده و به عنوان چالش ارسال می‌کند.
- (۳) اثبات‌کننده در پاسخ $s = [k^{-1} \cdot (\alpha + xr) \bmod q]$ را ارسال می‌کند.

با در نظر گرفتن یک اجرای یکتا از $\mathcal{A}$ به عنوان یک زیرروال از $\mathcal{A}'$ ، فرض کنید ω نشان‌دهنده‌ی مقدار تصادفی استفاده شده در آن اجرا به غیر از برای خود چالش، باشد. بنابراین، ω شامل هر مقدار تصادفی استفاده شده توسط G ، انتخاب کلید خصوصی (نامعلوم) x ، هر مقدار تصادفی استفاده شده توسط خود $\mathcal{A}$ و هر مقدار تصادفی استفاده شده توسط $\mathcal{A}'$ هنگام پاسخ به پرسش‌ها از $Trans_{sk}$ است. $V(\omega, r)$ را برابر با ۱ تعریف می‌کنیم اگر و تنها اگر $\mathcal{A}$ به درستی به چالش r پاسخ دهد هنگامی که مقدار تصادفی ω در مابقی اجرا، مورد استفاده قرار می‌گیرد. برای هر ω ثابت، $\delta_\omega \stackrel{\text{def}}{=} \Pr_r[V(\omega, r) = 1]$ احتمال روی انتخاب چالش r است که $\mathcal{A}$ پاسخ درستی ارائه کند.

$\delta(n) \stackrel{\text{def}}{=} \Pr[\text{Ident}_{\mathcal{A}, n}(n) = 1]$ را تعریف کنید. از آنجاکه شبیه‌سازی پیشگوی $Trans_{sk}$ کامل است، داریم

$$\delta(n) = \Pr_{\omega, r}[V(\omega, r) = 1] = \sum_{\omega} \Pr[\omega] \cdot \delta_\omega.$$

به علاوه، برداشت شهودی قبل از ارائه‌ی اثبات نشان می‌دهد که $\mathcal{A}'$ به درستی لگاریتم گسسته‌ی y را محاسبه خواهد کرد هرگاه $\mathcal{A}$ دو بار موفق شود و $r_1 \neq r_2$. بنابراین:

$$\begin{aligned} \Pr[\text{DLog}_{\mathcal{A}', G}(n) = 1] &= \Pr_{\omega, r_1, r_2}[V(\omega, r_1) \wedge V(\omega, r_2) \wedge r_1 \neq r_2] \\ &\geq \Pr_{\omega, r_1, r_2}[V(\omega, r_1) \wedge V(\omega, r_2)] - \Pr_{\omega, r_1, r_2}[r_1 = r_2] \\ &= \sum_{\omega} \Pr[\omega] \cdot (\delta_\omega)^2 - \frac{1}{q} \\ &\geq \left(\sum_{\omega} \Pr[\omega] \cdot \delta_\omega \right)^2 - \frac{1}{q} \\ &= \delta(n)^2 - 1/q, \end{aligned}$$

با استفاده از نامساوی جنسن^{۴۹} در گام دوم قبل از آخر، اگر مسئله‌ی لگاریتم گسسته نسبت به G سخت باشد، آنگاه $\Pr[\text{DLog}_{\mathcal{A}', G}(n) = 1]$ ناچیز است. از آنجاکه $1/q$ ناچیز است (به این دلیل که $\|q\| = n$ ، این امر به شکل ضمنی نشان می‌دهد که $\delta(n)$ نیز ناچیز است و بنابراین، Π یک طرح شناسایی امن است. ■

^{۴۹} نامساوی جنسن بیان می‌کند که $(\sum_i a_i \cdot b_i)^2 \geq (\sum_i a_i)^2 \cdot (\sum_i b_i)$ برای $\{a_i\}$ مثبت.

(۴) واریسی کننده پاسخ را می پذیرد اگر $s \neq 0$ و $g^{as^{-1}} \cdot y^{rs^{-1}} \stackrel{?}{=} I$

توجه کنید که $s \neq 0$ مگر اینکه $\alpha = -xr \bmod q$ باشد، که با احتمال ناچیز رخ خواهد داد. با فرض اینکه $s \neq 0$ و $s^{-1} \bmod q$ وجود دارد و

$$g^{as^{-1}} \cdot y^{rs^{-1}} = g^{as^{-1}} \cdot g^{xrs^{-1}} = g^{(a+xr) \cdot s^{-1}} = g^{(a+xr) \cdot k \cdot (a+xr)^{-1}} = I.$$

بنابراین، مشاهده می کنیم که درستی برقرار است مگر با احتمال ناچیز.

می توان نشان داد که این طرح شناسایی، امن خواهد بود اگر مسئله ی لگاریتم گسسته نسبت به G سخت باشد. ما تنها نمای کلی از این استدلال را ترسیم می کنیم و فرض می کنیم خواننده با نتایج بخش قبلی آشنایی دارد. اول اینکه، رونوشت های اجراهای صادقانه را می توان شبیه سازی نمود: برای انجام این کار، $\alpha, r \in \mathbb{Z}_q^*$ و $s \in \mathbb{Z}_q^*$ یکنواخت را انتخاب کرده و سپس قرار می دهیم $I := g^{as^{-1}} \cdot y^{rs^{-1}}$. (این فرایند دیگر یک شبیه سازی «کامل» را به دست نمی دهد ولی به اندازه ی کافی نزدیک به حالت کامل هست). به علاوه، اگر مهاجم یک پیام اولیه ی I را خروجی دهد که برای آن بتواند پاسخ های درست $s_1, s_2 \in \mathbb{Z}_q^*$ را به چالش های متمایز $(\alpha_1, r_1), (\alpha_2, r_2)$ بدهد، آنگاه

$$g^{as_1^{-1}} \cdot y^{rs_1^{-1}} = I = g^{as_2^{-1}} \cdot y^{rs_2^{-1}},$$

و بنابراین، $g^{a(s_1^{-1} - s_2^{-1})} = h^{r_1 s_1^{-1} - r_2 s_2^{-1}} = \log_g h$ را می توان به مانند بخش قبلی، محاسبه نمود. همین امر برقرار خواهد بود اگر مهاجم، پاسخ های درست را برای چالش های متمایز $(\alpha_1, r), (\alpha_2, r)$ ارائه کند.

طرح های امضای $DSA/ECDSA$ را می توان با «تقلیل» طرح شناسایی بالا به یک الگوریتم غیرتعاملی اجرا شده توسط امضاکننده، ساخت. باین حال، برخلاف تبدیل فیات-شامیر، تبدیل در اینجا به صورت زیر (ن.ک. ساختار ۱۲-۱۳) انجام داد:

- قرار می دهیم $\alpha := H(m)$ که در آن، m پیام مورد امضا و H یک تابع چکیده ساز رمزنگاری است.
- قرار می دهیم $r := F(I)$ برای یک تابع (مشخص) $F: G \rightarrow \mathbb{Z}_q$. در اینجا، F یک تابع «ساده» است که هدف آن عمل کردن به عنوان یک پیشگوی تصادفی، نیست.

تابع F به گروه G بستگی دارد که آن هم به نوبه ی خود به طرح بستگی دارد. در DSA ، G به عنوان یک زیرگروه مرتبه q از $\mathbb{Z}_p^*$ برای p اول (ن.ک. بخش ۸-۳-۳) و $F(I) \stackrel{\text{def}}{=} [I \bmod q]$ در نظر گرفته

می شود. در $ECDSA$ ، G یک زیرگروه مرتبه q از یک گروه خم بیضوی $E(\mathbb{Z}_p)$ برای p اول است. بخش ۸-۳-۴ به یاد داریم که هر عنصری در چنین گروهی را می توان به صورت یک زوج $(x, y) \in \mathbb{Z}_p \times \mathbb{Z}_p$ نمایش داد. تابع F در این حالت، به صورت $F((x, y)) \stackrel{\text{def}}{=} [x \bmod q]$ تعریف می شود.

$ECDSA$ و DSA به شکل مجرد

ساختار ۱۲-۱۳

فرض کنید G به صورت تعریف شده در متن باشد.

- **Gen**: با دریافت 1^n به عنوان ورودی، $G(1^n)$ را اجرا می کنیم تا (G, q, g) به دست آید. $x \in \mathbb{Z}_q$ یکنواخت را انتخاب کرده و قرار می دهیم $y := g^x$. کلید عمومی برابر با (G, q, g, y) و کلید خصوصی برابر با x است.
- به عنوان بخشی از تولید کلید، دو تابع $F: G \rightarrow \mathbb{Z}_q$ و $H: \{0, 1\}^* \rightarrow \mathbb{Z}_q$ انتخاب می شوند ولی ما این مقوله را به شکل ضمنی باقی می گذاریم.
- **Sign**: با دریافت کلید خصوصی x و یک پیام $m \in \{0, 1\}^*$ به عنوان ورودی، $k \in \mathbb{Z}_q^*$ یکنواخت را انتخاب کرده و قرار می دهیم $r := F(g^k)$. سپس، $s := [k^{-1} \cdot (H(m) + xr) \bmod q]$ را محاسبه می کنیم. (اگر $r = 0$ یا $s = 0$ آنگاه دوباره با یک انتخاب تازه از k شروع می کنیم.) امضای (r, s) را به عنوان خروجی ارائه می کنیم.
- **Verify**: با دریافت یک کلید عمومی (G, q, g, y) ، یک پیام $m \in \{0, 1\}^*$ و یک امضای (r, s) با $r, s \neq 0 \bmod q$ به عنوان ورودی، خروجی ۱ را ارائه می کنیم اگر و تنها اگر $r \stackrel{?}{=} F(g^{H(m) \cdot s^{-1}} y^{r \cdot s^{-1}})$

با فرض سختی مسئله ی لگاریتم گسسته، می توان امنیت DSA و $ECDSA$ را ثابت نمود اگر H و F به صورت پیشگوهای تصادفی مدل سازی شده باشند. باین حال، همان طور که در بالا بحث کردیم، درحالی که مدل پیشگوی تصادفی ممکن است برای H منطقی باشد، ولی یک مدل مناسب برای F نیست. باین حال، $ECDSA$ و DSA به مدت چندین دهه استفاده و مطالعه شده اند بدون اینکه حمله ای علیه آن ها یافت شود.

تولید مناسب k

طرح های $DSA/ECDSA$ مشخص می کنند که امضاکننده باید یک $k \in \mathbb{Z}_q^*$ یکنواخت را هنگام محاسبه ی یک امضا، انتخاب کند. ناتوانی در انتخاب درست k (برای مثال به دلیل تولید ضعیف عدد

^{**} $ECDSA$ همچنین خم های بیضی در سطح دیگر میدان ها را اجازه می دهد ولی ما تنها حالت میدان های اول در بخش ۸-۳-۴ را در نظر گرفته ایم.

کند (Gen, Sign, Vrfy) $\Pi =$ یک طرح امضا باشد و آزمایش زیر را برای یک متخاصم $\mathcal{A}$ در نظر بگیرید:

- (۱) آزمایش امضای یک بار مصرف $(\text{Sign} - \text{forge})_{\mathcal{A}, \Pi}^{-\text{time}}(n)$
- (۲) $\text{Gen}(1^n)$ اجرا می شود تا کلیدهای (pk, sk) به دست آید.
- (۳) متخاصم $\mathcal{A}$ مقدار pk را دریافت می کند و یک پرسمان یکنای m' را از پیشگوی خود یعنی $(\text{Sign}_{sk}(\cdot))$ می پرسد. سپس، $\mathcal{A}$ خروجی (m, σ) را با $m' \neq m$ ارائه می کند.
- (۴) خروجی آزمایش برابر با ۱ تعریف می شود اگر و تنها اگر $\text{Vrfy}_{pk}(m, \sigma) = 1$ باشد.

تعریف ۱۲-۱۴

طرح امضای $\Pi = (\text{Gen}, \text{Sign}, \text{Vrfy})$ جعل ناپذیر وجودی تحت یک حمله ی تک پیامی است یا یک طرح امضای یک-بار-امن است اگر برای تمام متخاصمین زمان-چندجمله ای تصادفی $\mathcal{A}$ یک نایج negl وجود داشته باشد به صورتی که:

$$\Pr[\text{Sign} - \text{forge}_{\mathcal{A}, \Pi}^{-\text{time}}(n) = 1] \leq \text{negl}(n).$$

للی لمپورت ساختاری از یک طرح امضای یک-بار-امن را در سال ۱۹۷۹ نشان داد. ما این ایده را برای حالت امضای پیام های ۳-بیتی نشان می دهیم. فرض کنید H یک تابع چکیده ساز رمزنگاری بلند یک کلید خصوصی شامل شش مقدار یکنواخت $x_{1,1}, x_{1,2}, x_{2,1}, x_{2,2}, x_{3,1}, x_{3,2} \in \{0,1\}^n$ بوده و کلید عمومی متناظر شامل نتایج به دست آمده با اعمال H بر هر کدام از این عناصر است. این کلیدها را می توان به صورت آرایه های دوبعدی تجسم کرد:

$$pk = \begin{pmatrix} y_{1,1} & y_{2,1} & y_{3,1} \\ y_{1,2} & y_{2,2} & y_{3,2} \end{pmatrix} \quad sk = \begin{pmatrix} x_{1,1} & x_{2,1} & x_{3,1} \\ x_{1,2} & x_{2,2} & x_{3,2} \end{pmatrix}.$$

به منظور امضای یک پیام $m = m_1 m_2 m_3$ (که در آن، $m_i \in \{0,1\}$ است)، امضاکننده، پیش تصویر مناسب x_{i,m_i} را برای هر بیت از پیام، منتشر می کند؛ امضای σ شامل سه مقدار $(x_{1,m_1}, x_{2,m_2}, x_{3,m_3})$ است. راست آزمایی به شکل طبیعی انجام می شود: با داشتن نمونه امضای (x_1, x_2, x_3) بر روی پیام $m = m_1 m_2 m_3$ ، پذیرش را انجام می دهیم اگر و تنها اگر $y_{i,m_i} \stackrel{?}{=} H(x_i)$ برای $1 \leq i \leq 3$ باشد. این امر به صورت گرافیکی در تصویر ۱۲-۳ نشان داده شده است و حالت کلی- برای پیام های با هر طول l - به صورت رسمی در ساختار ۱۲-۱۵ توصیف شده است.

بعد از مشاهده ی یک امضا بر روی یک پیام، مهاجمی که می خواهد یک امضا را بر روی هر پیام دیگری جعل نماید باید یک پیش تصویر از یکی از سه عنصر «استفاده نشده» در کلید عمومی، بیابد. اگر H ، یک طرفه باشد (ن.ک. تعریف ۸-۷۲)، آنگاه یافتن چنین پیش تصویری از لحاظ محاسباتی دشوار است.

تصادفی می تواند منجر به نتایج مخرب شود. برای مثال، اگر مهاجم بتواند مقدار k استفاده شده برای محاسبه ی یک امضای (r, s) بر روی یک پیام m را پیش بینی کند، آنگاه می تواند کلید خصوصی امضاکننده را هم محاسبه کند. این امر صدق می کند چرا که $s = k^{-1} \cdot (H(m) + xr) \bmod q$ و اگر k معلوم باشد، آنگاه تنها مجهول در این رابطه، کلید خصوصی x است.

حتی اگر k قابل پیش بینی نباشد، مهاجم می تواند کلید خصوصی امضاکننده را محاسبه کند اگر k یکسانی برای تولید دو امضای متفاوت، مورد استفاده قرار گیرد. مهاجم می تواند به سادگی تشخیص دهد چه زمانی این اتفاق می افتد به این دلیل که در این حالت، r هم تکرار می شود. فرض کنید (r, s_1) و (r, s_2) به ترتیب امضاهایی بر پیام های m_1 و m_2 باشند. آنگاه

$$s_1 = k^{-1} \cdot (H(m_1) + xr) \bmod q \\ s_2 = k^{-1} \cdot (H(m_2) + xr) \bmod q.$$

تفریق می دهد $k^{-1}(H(m_1) - H(m_2)) \bmod q = s_1 - s_2$ که از روی آن می توان k را محاسبه نمود؛ با داشتن k ، مهاجم می تواند کلید خصوصی x را به شکل بیان شده در بند قبلی، تعیین کند. همین حمله توسط هکرها برای استخراج کلید خصوصی اصلی از پلی استیشن سونی (PS۳) در سال ۲۰۱۰ استفاده شد.

۱۲-۶-۴ امضاها از روی توابع چکیده ساز

جالب- و شاید تا حدی عجیب- است که طرح های امضا را می توان بر اساس توابع چکیده ساز رمزنگاری بدون تکیه به فرضیات نظریه ای اعداد، ایجاد نمود. (این امر متضاد با رمزگذاری کلید عمومی است که به نظر می رسد نیازمند مسائل سخت دارای ساختار جبری است). در این بخش، ما چنین ساختارهایی را بررسی می کنیم. طرح هایی که مشاهده می کنیم همچنین به این دلیل جذاب هستند که به پیشگوهای تصادفی تکیه ندارند که برخلاف تمام ساختارهایی است که قبلاً در این فصل توصیف کرده ایم.

۱۲-۶-۱ طرح امضای لمپورت

مطالعه ی خود بر روی طرح های امضای مبتنی بر توابع چکیده ساز را با بررسی مفهوم نسبتاً ضعیف «طرح های امضای یک-بار-امن» آغاز می کنیم. به شکل غیررسمی، چنین طرح هایی، «امن» هستند مادامی که یک کلید خصوصی مفروض برای امضای تنها یک پیام یکتا مورد استفاده قرار گیرد. طرح های صادق در این مفهوم از امنیت ممکن است برای بعضی از کاربردها مناسب باشند و همین طور به عنوان «بلوک های ساختاری» برای رسیدن به مفاهیم قوی تر برای امنیت مورد استفاده قرار گیرند همان طور که در بخش بعدی مشاهده خواهیم کرد.

پیام دیگر $m' \neq m$ را در نظر می‌گیرد. باید حداقل یک جایگاه $\{1, \dots, \ell\}$ $i^* \in \{1, \dots, \ell\}$ وجود داشته باشد که بر روی آن، m و m' باهم متفاوت باشند. فرض کنید $m_i = b \neq m'_i$ آنگاه، جعل یک امضا بر روی m حداقل نیازمند یافتن یک پیش تصویر (تحت H) از عنصر $y_{i^*,0}$ کلید عمومی است. از آنجاکه H یک طرفه است، این امر باید غیرعملی باشد. اکنون این برداشت شهودی را به شکل رسمی بیان می‌کنیم:

فرض کنید Π نشان‌دهنده‌ی طرح لمپورت باشد و فرض کنید $\mathcal{A}$ یک متخاصم زمان-چندجمله‌ای تصادفی باشد. در یک اجرای خاص از $\text{Sig-Forge}_{\mathcal{A}, \Pi}^{\text{time}}(n)$ فرض کنید m' نشان‌دهنده‌ی بیامی باشد که امضای آن توسط $\mathcal{A}$ درخواست شده است (بدون کاستن از کلیت، فرض می‌کنیم که $\mathcal{A}$ همواره یک امضا را بر روی یک پیام، درخواست می‌کند) و فرض کنید (m, σ) آخرین خروجی $\mathcal{A}$ باشد. می‌گوییم $\mathcal{A}$ یک جعل را در (i, b) خروجی می‌دهد اگر $\text{Vrfy}_{pk}(m, \sigma) = 1$ و به‌علاوه $m_i \neq m'_i$ (یعنی، پیام‌های m و m' در جایگاه i با خود باهم تفاوت دارند) و $m_i = b \neq m'_i$ توجه کنید که هرگاه $\mathcal{A}$ یک جعل را خروجی می‌دهد، یک جعل را در یک (i, b) مفروض خروجی خواهد داد.

PPPT: زیرا، $\mathcal{A}$ را در نظر بگیرید که تلاش می‌کند H را تاراج کند.

الگوریتم $\mathcal{J}$

این الگوریتم به‌عنوان ورودی، 1^n و $\mathcal{A}$ را دریافت می‌کند.

- (۱) $i^* \in \{1, \dots, \ell\}$ و $b^* \in \{0, 1\}$ یکنواخت را انتخاب کنید. قرار دهید $y_{i^*,0} := y$.
- (۲) برای تمام $i \in \{1, \dots, \ell\}$ و $b \in \{0, 1\}$ با $(i^*, b^*) \neq (i, b)$:
• $x_{i,b} \in \{0, 1\}^n$ یکنواخت را انتخاب کرده و قرار دهید $y_{i,b} := H(x_{i,b})$.
- (۳) $\mathcal{A}$ را بر روی ورودی $pk := (y_{1,0}, y_{1,1}, \dots, y_{\ell,0}, y_{\ell,1})$ اجرا کنید.
- (۴) زمانی که $\mathcal{A}$ یک امضا را بر روی پیام m' درخواست می‌کند:

- اگر $\mathcal{A}$ یک جعل را در (i^*, b^*) اعلام کرد، آنگاه خروجی x_{i^*,b^*} را ارائه کنید.

هرگاه $\mathcal{A}$ یک جعل را در (i^*, b^*) خروجی می‌دهد، الگوریتم $\mathcal{J}$ موفق به وارون سازی ورودی y داده‌شده به آن، خواهد شد. ما علاقه‌مند به این احتمال هستیم که این اتفاق زمانی رخ دهد که ورودی برای $\mathcal{J}$ با انتخاب $x \in \{0, 1\}^n$ یکنواخت و قرار دادن $y := H(x)$ (ن.ک. تعریف ۷۲-۸) تولید می‌شود. یک «آزمایش ذهنی» را در نظر بگیرید که در آن، $\mathcal{J}$ در ابتدای آزمایش، مقدار x را دریافت می‌کند، $x_{i^*,b^*} := x$ را قرار می‌دهد و سپس همواره یک امضا را در گام چهارم به $\mathcal{A}$ بازمی‌گرداند (یعنی حتی اگر $b^* = m'_{i^*}$ باشد). دیدگاه $\mathcal{A}$ هنگام اجرا شدن به‌عنوان یک زیرروال توسط $\mathcal{J}$ در این آزمایش

امضای $m = 011$

$$sk = \begin{pmatrix} \boxed{x_{1,0}} & x_{1,1} & x_{1,1} \\ x_{1,1} & \boxed{x_{2,1}} & \boxed{x_{2,1}} \end{pmatrix} \Rightarrow \sigma = (x_{1,0}, x_{2,1}, x_{2,1})$$

واریسی برای $m = 011$ و $\sigma = (x_1, x_2, x_2)$

$$pk = \begin{pmatrix} \boxed{y_{1,0}} & y_{1,1} & y_{1,1} \\ y_{1,1} & \boxed{y_{2,1}} & \boxed{y_{2,1}} \end{pmatrix} \Rightarrow \begin{aligned} H(x_1) &\stackrel{?}{=} y_{1,0} \\ H(x_2) &\stackrel{?}{=} y_{2,1} \\ H(x_2) &\stackrel{?}{=} y_{2,1} \end{aligned}$$

تصویر ۳-۱۲: طرح لمپورت استفاده‌شده برای امضای پیام $m = 011$

نسخه‌ی ۱۲-۱۶

فرض کنید l یک چندجمله‌ای باشد. اگر H یک تابع یک‌طرفه باشد، آنگاه ساختار ۱۲-۱۵ یک طرح امضای یک-پارامتر است.

طرح امضای لمپورت

ساختار ۱۲-۱۵

فرض کنید $\{0, 1\}^* \rightarrow \{0, 1\}^*$ یک تابع باشد. یک طرح امضا را برای پیام‌های با طول $l = l(n)$ به شکل زیر، می‌سازیم:

- Gen : با دریافت 1^n به‌عنوان ورودی، مراحل زیر را برای $i \in \{1, \dots, \ell\}$ انجام می‌دهیم:
(۱) $x_{i,0}, x_{i,1} \in \{0, 1\}^n$ یکنواخت را انتخاب می‌کنیم.
(۲) $y_{i,0} := H(x_{i,0})$ و $y_{i,1} := H(x_{i,1})$ را محاسبه می‌کنیم.

- $m_1, \dots, m_\ell$ به‌عنوان ورودی، امضای $(x_{1,m_1}, \dots, x_{\ell,m_\ell})$ را به‌عنوان خروجی ارائه می‌کنیم.
- $Vrfy$: با دریافت یک کلید عمومی pk به شکل بالا، یک پیام $m \in \{0, 1\}^\ell$ با $m = m_1, \dots, m_\ell$ و یک امضای $\sigma = (x_1, \dots, x_\ell)$ به‌عنوان ورودی، خروجی ۱ را ارائه می‌کنیم اگر و تنها اگر برای همه‌ی $1 \leq i \leq \ell$ ، داشته باشیم $H(x_i) = y_{i,m_i}$.

اثبات

فرض کنید در سراسر اثبات $l = l(n)$ باشد. همان‌طور که هم‌اکنون اشاره شد، مشاهده‌ی اصلی به این صورت است: فرض کنید یک مهاجم $\mathcal{A}$ یک امضا را بر روی یک پیام m' درخواست کرده و هر

ذهنی دارای توزیع یکسان با دیدگاه $\mathcal{A}$ در آزمایش $\text{Sig-Forge}_{\mathcal{A}, \Pi}^{\text{time}}(n)$ خواهد بود. به این دلیل که (i^*, b^*) به شکل یکنواخت در ابتدای آزمایش انتخاب شد و دیدگاه $\mathcal{A}$ مستقل از این انتخاب است، احتمال اینکه $\mathcal{A}$ اصلاً یک جعل را خروجی دهد حداقل برابر با $1/2l$ است. (این امر به این دلیل است که یک جعل امضا به شکل ضمنی نشان‌دهنده‌ی یک جعل برای حداقل یک نقطه‌ی (i, b) است. از آنجاکه $2l(n)$ نقطه وجود دارند، احتمال اینکه این جعل در نقطه‌ی (i^*, b^*) باشد حداقل برابر با $1/2l$ است.) نتیجه می‌گیریم که در این آزمایش ذهنی، احتمال اینکه $\mathcal{A}$ یک جعل را در (i^*, b^*) خروجی دهد حداقل برابر با $\frac{1}{2l} \cdot \Pr[\text{Sig-Forge}_{\mathcal{A}, \Pi}^{\text{time}}(n) = 1]$ است.

با بازگشت به آزمایش واقعی مربوط به I به‌صورت توصیف‌شده‌ی اولیه، مشاهده‌ی اصلی عبارت است از اینکه «احتمال اینکه $\mathcal{A}$ یک جعل را در (i^*, b^*) خروجی دهد، تغییری نمی‌کند». این امر به این دلیل است که آزمایش ذهنی و آزمایش واقعی باهم تلاقی خواهند داشت اگر $\mathcal{A}$ یک جعل را در (i^*, b^*) خروجی دهد. یعنی، این آزمایش‌ها تنها در صورتی متفاوت خواهند بود که $m_{i^*}^* = b^*$ ولی اگر این اتفاق رخ دهد، آنگاه برای $\mathcal{A}$ (بر اساس تعریف) غیرممکن است که در ادامه یک جعل را در (i^*, b^*) خروجی دهد. بنابراین، احتمال اینکه $\mathcal{A}$ یک جعل را در (i^*, b^*) خروجی دهد همچنان حداقل برابر با $\frac{1}{2l} \cdot \Pr[\text{Sig-Forge}_{\mathcal{A}, \Pi}^{\text{time}}(n) = 1]$ خواهد بود. به عبارت دیگر

$$\Pr[\text{Invert}_{I, H}(n) = 1] \geq \frac{1}{2l} \cdot \Pr[\text{Sig-Forge}_{\mathcal{A}, \Pi}^{\text{time}}(n) = 1].$$

به این دلیل که H یک تابع یک‌طرفه است، یک تابع ناچیز negl وجود دارد به صورتی که

$$\text{negl}(n) \geq \Pr[\text{Invert}_{I, H}(n) = 1].$$

از آنجاکه I چندجمله‌ای است، این امر به شکل ضمنی نشان می‌دهد که $\Pr[\text{Sig-Forge}_{\mathcal{A}, \Pi}^{\text{time}}(n) = 1]$ ناچیز است، که اثبات را کامل می‌کند. ■

نتیجه‌ی فرعی ۱۷-۱۲

اگر توابع یک‌طرفه وجود داشته باشند، آنگاه برای هر چندجمله‌ای I یک طرح امضای یک-بار-امن برای پیام‌های با طول l وجود خواهد داشت.

۱۲-۶-۲- امضاهای زنجیره‌ای

توانایی برای امضای تنها یک پیام یکتا با یک کلید خصوصی مفروض به‌وضوح یک نقطه‌ضعف جدی است. در اینجا رویکردی بر اساس توابع چکیده ساز مقاوم در برابر برخورد را نشان می‌دهیم که به امضاکننده اجازه می‌دهد که تعداد دلخواه پیام‌ها را امضا نماید با این هزینه که باید «حالتی» را حفظ کند که باید بعد از تولید هر امضا، به‌روزرسانی شود. در بخش ۱۲-۶-۳، یک نسخه‌ی کارآتر از این رویکرد را مورد بحث قرار می‌دهیم (که همچنان نیازمند حالت است) و سپس توضیح می‌دهیم که این

انتخاب تبدیل‌شده را چگونه می‌توان بدون حالت کرد. نتیجه نشان می‌دهد که طرح‌های امضای صادق به‌تعریف ۱۲-۲ را می‌توان بر اساس توابع چکیده ساز مقاوم در برابر برخورد، ایجاد نمود.

برای طرح‌های امضایی را تعریف می‌کنیم که به امضاکننده اجازه می‌دهند تا «حالتی» را حفظ کند که بعد از تولید هر امضا، به‌روزرسانی می‌شود.

تعریف ۱۲-۱۸

یک طرح امضای حالت‌مند یک چندگانه از الگوریتم‌های زمان-چندجمله‌ای تصادفی $(\text{Gen}, \text{Sign}, \text{Vrfy})$ است که در موارد زیر صدق می‌کند:

(۱) الگوریتم تولید کلید Gen به‌عنوان ورودی، یک پارامتر امنیتی 1^n را دریافت کرده و $(pk, sk, s.)$ را به‌عنوان خروجی ارائه می‌کند. این موارد را به ترتیب، کلید عمومی، کلید خصوصی و حالت اولیه می‌نامند. فرض می‌کنیم که هر کدام از pk و sk دارای طول حداقل n هستند و اینکه n را می‌توان از روی pk, sk تعیین نمود.

(۲) الگوریتم امضای Sign به‌عنوان ورودی، یک کلید خصوصی sk ، یک مقدار r و یک پیام $m \in \{0, 1\}^*$ را دریافت می‌کند. این الگوریتم، یک امضای σ و یک مقدر s را به‌عنوان خروجی ارائه می‌کند.

(۳) الگوریتم راست آزمایی قطعی Vrfy ، به‌عنوان ورودی، یک کلید عمومی pk ، یک پیام m و یک امضای σ را دریافت می‌کند. به‌عنوان خروجی، این الگوریتم یک بیت b را ارائه می‌کند. ملزم می‌کنیم که برای هر n ، هر $(pk, sk, s.)$ اعلام شده توسط $\text{Gen}(1^n)$ و هر پیام $m_1, \dots, m_t \in \{0, 1\}^*$ اگر به‌صورت بازگشتی $(\sigma_i, s_i) \leftarrow \text{Sign}_{sk, s_{i-1}}(m_i)$ را برای $i = 1, \dots, t$ محاسبه کنیم، آنگاه برای هر $i \in \{1, \dots, t\}$ خواهیم داشت $\text{Vrfy}_{pk}(m_i, \sigma_i) = 1$.

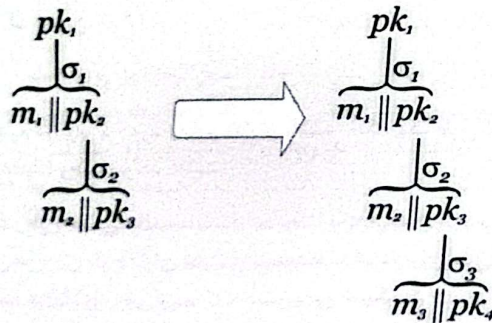
تأکید می‌کنیم که نیازی نیست واریسی کننده، حالت امضاکننده را بداند تا بتواند یک امضا را راست آزمایی کند؛ در واقع، در بعضی از طرح‌ها، حالت باید توسط امضاکننده مخفی نگه داشته شود تا امنیت برقرار بماند؛ طرح‌های امضایی که حالت را حفظ نمی‌کنند (مثل تعریف ۱۲-۱) را بدون حالت می‌نامیم تا آن‌ها را از طرح‌های حالت‌مند، متمایز کنیم. واضح است که طرح‌های بدون حالت، ترجیح داده می‌شوند (هر چند طرح‌های حالت‌مند همچنان می‌توانند به شکل بالقوه مفید باشند) امضاهای حالت‌مند را به‌عنوان یک گام اولیه برای رسیدن به یک ساختار بدون حالت نهایی، معرفی می‌کنیم.

امنیت برای طرح‌های امضای حالت‌مند دقیقاً مشابه با تعریف ۱۲-۲ است تنها با این جزئیات که پیشگوی امضاکننده تنها امضا را بازمی‌گرداند (و نه حالت را) و اینکه پیشگوی امضاکننده، حالت را در هر بار فراخوانی، به‌روزرسانی می‌کند.

برای هر چندجمله‌ای $I(n) = t$ ، به‌سادگی می‌توانیم یک طرح امضای «یک-بار-امن» حالت‌مند را ایجاد کنیم. (تعریف امنیت در اینجا عبارت خواهد بود از تعمیم واضح تعریف ۱۲-۱۲) می‌توانیم این کار را

بنحیله از یک طرح «زنجیره‌ای» می‌توانیم عملکرد بهتری داشته باشیم که در آن، امضاکننده پیام‌های عمومی اضافی را به هنگام نیاز به شکل فوری، تولید می‌کند.

در طرح زنجیره‌ای، کلید عمومی تنها شامل یک کلید عمومی یکتای pk_1 تولیدشده با استفاده از Gen است و کلید خصوصی درواقع کلید خصوصی مرتبط sk_1 است. به‌منظور امضای پیام اول m_1 امضاکننده در ابتدا یک زوج-کلید جدید (pk_2, sk_2) را با استفاده از Gen تولید می‌کند و سپس هم m_1 و هم pk_2 را با استفاده از sk_1 امضا می‌کند تا $\sigma_1 \leftarrow \text{Sign}_{sk_1}(m_1 \parallel pk_2)$ را به دست آورد. امضایی که اعلام می‌شود شامل هر دو pk_2 و σ_1 است و امضاکننده، $(m_1, pk_2, sk_2, \sigma_1)$ را به حالت جاری خود اضافه می‌کند. به‌طور کلی، وقتی زمان آن می‌رسد که پیام نام امضا شود، امضاکننده $\{(m_j, pk_{j+1}, sk_{j+1}, \sigma_j)\}_{j=1}^{t-1}$ را به‌عنوان بخشی از حالت خود، ذخیره کرده است. به‌منظور امضای پیام نام m_t امضاکننده در ابتدا یک زوج-کلید جدید (pk_{t+1}, sk_{t+1}) را با استفاده از Gen تولید می‌کند و سپس، m_t و pk_{t+1} را با استفاده از sk_t امضا می‌کند تا یک امضای $\sigma_t \leftarrow \text{Sign}_{sk_t}(m_t \parallel pk_{t+1})$ را به دست آورد. امضای واقعی که اعلام می‌شود شامل σ_t, pk_{t+1} و همین‌طور مقادیر $\{(m_j, pk_{j+1}, sk_{j+1}, \sigma_j)\}_{j=1}^{t-1}$ است. در ادامه، امضاکننده، $(m_t, pk_{t+1}, sk_{t+1}, \sigma_t)$ را به حالت خود اضافه می‌کند. برای ترسیم گرافیکی این فرآیند ن.ک. تصویر ۱۲-۴.



تصویر ۱۲-۴: امضاهای زنجیره‌ای: وضعیت قبل و بعد از امضای پیام سوم m_3

به‌منظور راست آزمایی یک امضای $\{(m_j, pk_{j+1}, sk_{j+1}, \sigma_j)\}_{j=1}^{t-1}$ بر روی یک پیام $m = m_t$ بر اساس کلید عمومی pk_1 ، گیرنده هر لینک بین یک کلید عمومی pk_j و کلید عمومی بعدی pk_{j+1} در زنجیره و همین‌طور لینک بین آخرین کلید عمومی pk_t و m را راست آزمایی می‌کند. یعنی، واریسی دارای خروجی ۱ خواهد بود اگر و تنها اگر برای تمام $i \in \{1, \dots, t\}$ داشته باشیم $\text{Vrfy}_{pk_i}(m_i \parallel \text{...}) = 1$ (ن.ک. تصویر ۱۲-۴).

با در نظر گرفتن کلید عمومی (به ترتیب، کلید خصوصی) شامل t کلید عمومی (به ترتیب، کلید خصوصی) تولیدشده به شکل مستقل برای هر طرح امضای یک-بار-امن انجام دهیم؛ یعنی، قرار دادن $pk := (pk_1, \dots, pk_t)$ و $sk := (sk_1, \dots, sk_t)$ که در آن، هر (pk_i, sk_i) یک زوج-کلید تولیدشده مستقل برای یک طرح امضای یک-بار-امن است. حالت یک شمارنده‌ی i است که در ابتدا برابر با ۱ قرار داده می‌شود. به‌منظور امضای یک پیام m با استفاده از کلید خصوصی sk و حالت جاری i ، مقدار $\sigma \leftarrow \text{Sign}_{sk_i}(m)$ را محاسبه می‌کنیم (یعنی، یک امضا را بر روی m با استفاده از کلید خصوصی sk_i تولید می‌کنیم) و خروجی (σ, i) را ارائه می‌کنیم؛ حالت به $i := i + 1$ به‌روزرسانی می‌شود. از آنجاکه حالت از ۱ شروع می‌شود، این امر به این معنا است که پیام نام با استفاده از sk_i امضا می‌شود. راست آزمایی یک امضای (σ, i) بر روی یک پیام m با بررسی این مسئله انجام می‌شود که آیا σ یک امضای معتبر بر روی m بر اساس pk_i هست یا خیر. این طرح، امن است اگر برای امضای t پیام استفاده شود چراکه هر کلید خصوصی طرح یک-بار-امن مبنا برای امضای تنها یک پیام یکتا استفاده می‌شود.

به شکل توصیف‌شده، امضاها دارای طول ثابت (یعنی مستقل از t) هستند ولی کلید عمومی دارای طول خطی بر اساس t است. می‌توان طول کلید عمومی و امضا را به این صورت مورد مصالحه قرار داد که امضاکننده را وارد کنیم یک درخت مرکب $h := \text{MT}_t(pk_1, \dots, pk_t)$ (ن.ک. بخش ۵-۲) را بر روی t کلید عمومی مبنا از طرح یک-بار-امن، محاسبه نماید. یعنی، کلید عمومی اکنون برابر با (t, h) خواهد بود و امضاها بر روی پیام نام شامل (σ, i) به‌مانند قبل به همراه نامین مقدار pk_i و یک اثبات π_i خواهند بود که مقدار درست متناظر با h است. (راست آزمایی به‌صورت طبیعی انجام می‌شود). اکنون کلید عمومی دارای اندازه‌ی ثابت است و طول امضا تنها به‌صورت لگاریتمی با t افزایش می‌یابد.

از آنجاکه t می‌تواند یک چندجمله‌ای دلخواه باشد، چرا طرح‌های قبلی راه‌حلی را که به دنبال آن هستیم به دست نمی‌دهند؟ اصلی‌ترین نقطه‌ضعف این است که این طرح‌ها نیازمند آن هستند که کران بالای t بر روی تعداد پیام‌هایی که می‌توانند امضا شوند، «از قبل» و در زمان تولید کلید، مشخص شود. این امر یک محدودیت بالقوه شدید است چراکه زمانی که به کران بالا برسیم، یک کلید عمومی جدید را باید تولید و توزیع نمود. به‌جای این کار، ترجیح می‌دهیم که یک کلید عمومی ثابت یکتا داشته باشیم که بتوان از آن برای امضای تعداد نامحدودی پیام استفاده نمود.

فرض کنید $\Pi = (Gen, Sign, Vrfy)$ یک طرح امضای یک-بار-امن باشد. در طرحی که هم‌اکنون توصیف کردیم (با نادیده‌گیری بهینه‌سازی درخت مرکب)، امضاکننده، t فراخوانی از Gen را انجام می‌دهد تا کلیدهای عمومی $pk_1, \dots, pk_t$ را به دست آورد و هرکدام از این‌ها را در کلید عمومی pk واقعی خود در نظر می‌گیرد. در ادامه، امضاکننده محدود به امضای حداکثر t پیام خواهد بود. با

نشانده در درخت از یک برگ تا ریشه خواهد بود؛ مادامی که ریشه دارای عمق چندجمله‌ای باشد بنی اگر دارای اندازه‌ی نمایی باشد)، راست آزمایی را می‌توان در زمان چندجمله‌ای انجام داد.

بشکل عینی، به‌منظور امضای پیام‌های با طول n ، ما با یک درخت دودویی به عمق n و دارای 2^n برگ کار می‌کنیم. به‌مانند قبل، امضاکننده، گره‌ها را در لحظه و به شکل موردنیاز به درخت اضافه می‌کند. باین‌حال، برخلاف طرح زنجیره‌ای، تنها برگ‌ها (و نه گره‌های داخلی) برای امضای پیام‌ها به‌دستگاه قرار خواهند گرفت. هر برگ این درخت متناظر با یکی از پیام‌های بالقوه با طول n خواهد بود.

باجزئیات دقیق‌تر، ما یک درخت دودویی با عمق n را تصور می‌کنیم که در آن، ریشه توسط E (یعنی رشته‌ی تهی) برچسب شده است و یک گره با رشته‌ی دودویی w (با طول کمتر از n) برچسب زده شده است که فرزند سمت چپ آن با $w0$ و فرزند سمت راست آن با $w1$ برچسب زده می‌شوند. این درخت هیچ‌گاه به‌طور کامل ساخته نمی‌شود (توجه کنید که این درخت دارای اندازه‌ی نمایی است)، بلکه به شکل موردنیاز توسط امضاکننده ساخته خواهد شد.

برای هر گره w ، یک زوج کلید pk_w, sk_w را برای یک طرح امضای یک-بار-امن Π مرتبط می‌کنیم. کلید عمومی ریشه، pk_E برابر است با کلید عمومی واقعی امضاکننده. به‌منظور امضای یک پیام $m \in \{0,1\}^*$ ، امضاکننده موارد زیر را انجام می‌دهد:

- (۱) در ابتدا، کلیدها (بر اساس نیاز) را برای تمام گره‌ها بر روی مسیر از ریشه تا برگ دارای برچسب m تولید می‌کند. (بعضی از این کلیدهای عمومی ممکن است در فرآیند امضای پیام‌های قبلی تولید شده باشند و در این حالت، دوباره تولید نمی‌شوند.)
- (۲) در ادامه، امضاکننده، مسیر از ریشه تا برگ برچسب خورده توسط m را با محاسبه‌ی یک امضا بر روی $pk_w \parallel sk_w$ با استفاده از کلید خصوصی sk_w برای هر رشته‌ی w که یک پیشوند مناسب برای m است، «گواهی» می‌کند.
- (۳) درنهایت، خود m را با محاسبه‌ی یک امضا بر روی m با استفاده از کلید خصوصی sk_w «گواهی» می‌کند.

امضای نهایی بر روی m شامل امضا بر روی m بر اساس pk_m و همین‌طور تمام اطلاعات موردنیاز برای واریسی مسیر از برگ دارای برچسب m تا ریشه است؛ ن.ک. تصویر ۱۲-۵. به‌علاوه، امضاکننده، حالت خود را با ذخیره‌ی تمام کلیدهای تولیدشده به‌عنوان بخشی از فرآیند امضا کردن بالا، به‌روزرسانی می‌کند. یک توصیف رسمی از این طرح به‌عنوان ساختار ۱۲-۲۰ ارائه شده است.

سخت نیست قانع شویم حداقل در سطح شهودی- که این طرح امضا جعل ناپذیر وجودی تحت یک حمله‌ی پیام منتخب وفقی است (بدون توجه به تعداد پیام‌هایی که امضا می‌شوند). به شکل غیررسمی، این امر بازهم به دلیل این واقعیت است که هر زوج-کلید (pk_i, sk_i) تنها برای امضای یک «پیام» یکتا مورد استفاده قرار می‌گیرد که در این حالت، «پیام» درواقع یک زوج پیام/کلید عمومی $pk_{i+1} \parallel m_i$ است. از آنجاکه ما امنیت یک طرح کارتر را در بخش بعدی اثبات خواهیم کرد، امنیت را برای طرح زنجیره‌ای در اینجا اثبات نمی‌کنیم.

در طرح زنجیره‌ای، هر کلید عمومی pk_i برای امضا کردن یک پیام و همین‌طور یک کلید عمومی دیگر استفاده می‌شود. بنابراین، ضروری است که طرح امضای یک-بار-امن Π «قادر به امضای پیام‌های طولانی‌تر از کلید عمومی» باشد. طرح لمپورت ارائه‌شده در بخش ۱۲-۶-۱ دارای این ویژگی نیست. باین‌حال، اگر الگوی چکیده سازی-و-امضا از بخش ۱۲-۳ را برای طرح لمپورت اعمال کنیم، یک طرح امضای یک-بار-امن را به دست می‌آوریم که می‌تواند پیام‌های با طول دلخواه را امضا کند. (هرچند قضیه‌ی ۱۲-۴ تنها در رابطه با طرح‌های امضایی بیان شد که در تعریف ۱۲-۲ صدق می‌کنند، مشاهده‌ی اینکه یک اثبات مشابه برای طرح‌های امضای یک-بار-امن مؤثر است، اصلاً سخت نیست.) به این دلیل که این نتیجه برای بخش بعدی ضروری است، آن را به‌صورت رسمی بیان می‌کنیم. (توجه کنید که وجود توابع چکیده ساز مقاوم در برابر برخورد به‌صورت ضمنی نشان‌دهنده‌ی وجود توابع یک‌طرفه است؛ ن.ک. تمرین ۷-۴.)

لم ۱۲-۱۹

اگر توابع چکیده ساز مقاوم در برابر برخورد وجود داشته باشند، آنگاه یک طرح امضای یک-بار-امن (برای پیام‌های با طول دلخواه) وجود خواهد داشت.

طرح امضای زنجیره‌ای یک طرح امضای حالتمند است که تحت یک حمله‌ی پیام منتخب وفقی، جعل ناپذیر وجودی است. باین‌حال، این طرح دارای تعدادی نقطه‌ضعف است. برای مثال، هیچ روش فوری برای حذف حالت وجود ندارد (یادآوری می‌کنیم که هدف نهایی ما یک طرح بدون حالت صادق در تعریف ۱۲-۲ است). همچنین خیلی کارا نیست به این معنا که طول امضا، اندازه‌ی حالت و زمان راست آزمایی همگی خطی بر اساس تعداد پیام‌های مورد امضا هستند. درنهایت، هر امضا تمام پیام‌های امضا شده‌ی قبلی را افشا می‌کند و این امر ممکن است در بعضی موقعیت‌ها، نامطلوب باشد.

۱۲-۶-۳- امضاهای مبتنی بر درخت

امضاکننده در طرح زنجیره‌ای بخش قبلی را می‌توان حفظ‌کننده‌ی یک «درخت» از درجه‌ی ۱ به شمار آورد که ریشه‌ی آن در کلید عمومی pk_1 با عمق برابر با تعداد پیام‌های امضا شده تاکنون است (ن.ک. تصویر ۱۲-۴). یک روش طبیعی برای بهبود کارایی عبارت است از استفاده از یک درخت «دودویی» که در آن، هر گره دارای مرتبه‌ی ۲ است. به‌مانند قبل، یک امضا متناظر با یک مسیر

یک طرح امضای «مبتنی بر درخت»

ساختار ۲۰-۱۲

فرض کنید $\Pi = (\text{Gen}, \text{Sign}, \text{Vrfy})$ یک طرح امضا باشد. برای یک رشته دودویی m ، فرض کنید $m|_i \stackrel{\text{def}}{=} m_1 \dots m_i$ نشان دهنده پیشوند i -بیتی m باشد (با $\varepsilon \stackrel{\text{def}}{=} m|_0$ ، یعنی رشته خالی). طرح $\Pi^* = (\text{Gen}^*, \text{Sign}^*, \text{Vrfy}^*)$ را به صورت زیر ایجاد می‌کنیم:

• Gen^* : با دریافت 1^n به عنوان ورودی، $(pk_e, sk_e) \leftarrow \text{Gen}(1^n)$ را محاسبه کرده و کلید عمومی pk_e را به عنوان خروجی ارائه می‌کنیم. کلید خصوصی و حالت اولیه برابر با sk_e هستند.

• Sign^* : با دریافت یک پیام $m \in \{0, 1\}^n$ به عنوان ورودی، عملیات زیر را انجام می‌دهیم:
(۱) برای $i = 0$ to $n-1$

- اگر $pk_{m|i-1}$ و $sk_{m|i-1}$ در حالت نباشند، $(pk_{m|i}, sk_{m|i}) \leftarrow \text{Gen}(1^n)$ و $(pk_{m|i}, sk_{m|i}) \leftarrow \text{Gen}(1^n)$ و $pk_{m|i-1} \parallel pk_{m|i}$ را محاسبه می‌کنیم. به علاوه، تمام این مقادیر را به حالت اضافه می‌کنیم.

(۲) اگر sk_m هنوز در حالت قرار نگرفته است، $sk_m \leftarrow \text{Sign}_{sk_m}(m)$ را محاسبه کرده و آن را به عنوان بخشی از حالت، ذخیره می‌کنیم.

(۳) امضای $(\{sk_{m|i}, pk_{m|i}\}_{i=0}^{n-1}, sk_m)$ را به عنوان خروجی ارائه می‌کنیم.
• Vrfy^* : با دریافت یک کلید عمومی pk_e ، پیام m و امضای $(\{sk_{m|i}, pk_{m|i}\}_{i=0}^{n-1}, sk_m)$ ، مقدار ۱ را به عنوان خروجی ارائه می‌کنیم اگر و تنها اگر:

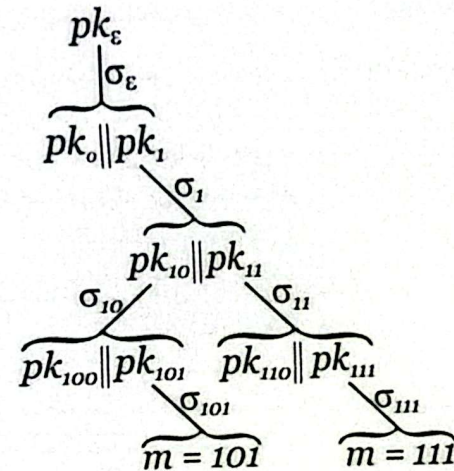
(۱) برای همه $i \in \{0, \dots, n-1\}$ داشته باشیم $\text{Vrfy}_{pk_{m|i}}(pk_{m|i} \parallel sk_{m|i}) = 1$
(۲) $\text{Vrfy}_{pk_m}(m, sk_m) = 1$

قضیه ۲۱-۱۲

فرض کنید Π یک طرح امضای یک-بار-امن باشد. آنگاه ساختار ۲۰-۱۲ یک طرح امضای امن است.

اثبات

فرض کنید Π^* نشان دهنده ساختار ۲۰-۱۲ باشد. فرض کنید $\mathcal{A}^*$ یک متخاصم زمان چندجمله‌ای تصادفی باشد، $\ell^* = \ell^*(n)$ یک کران بالا (چندجمله‌ای) بر روی تعداد پرسمان‌های امضای انجام شده توسط $\mathcal{A}^*$ باشد و قرار دهید $1 + \ell^*(n) \stackrel{\text{def}}{=} \ell$.



تصویر ۵-۱۲: امضاهای مبتنی بر درخت (به شکل مفهومی)

توجه کنید که هر کدام از کلیدهای مبنا در این طرح برای امضای تنها یک پیام «یکتا» مورد استفاده قرار می‌گیرد. هر کلید مرتبط با یک گره داخلی یک زوج کلید عمومی را امضا می‌کند و یک کلید در یک برگ برای امضای تنها یک پیام یکتا استفاده می‌شود. از آنجاکه هر کلید برای امضای یک «زوج» کلید دیگر استفاده می‌شود، بازهم نیازمند آن هستیم که طرح امضای یک-بار-امن Π قادر به امضای پیام‌های طولانی‌تر از کلید عمومی باشد. لم ۱۲-۱۹ نشان می‌دهد که چنین طرح‌هایی را می‌توان بر اساس توابع چکیده ساز مقاوم در برابر برخورد، ایجاد نمود.

قبل از اثبات امنیت این رویکرد مبتنی بر درخت، توجه کنید که این طرح از چند وجه طرح زنجیره‌ای را بهبود می‌بخشد. این طرح همچنان اجازه‌ی امضای تعداد نامحدودی پیام را می‌دهد. (هرچند تنها 2^n برگ وجود دارد، فضای پیام تنها شامل 2^n پیام است. در هر صورت، 2^n در نهایت بزرگ‌تر از هر تابع چندجمله‌ای از n خواهد بود.) از لحاظ کارایی، طول امضا و زمان واریسی اکنون متناسب با طول پیام n ولی مستقل از تعداد پیام‌های امضا شده، هستند. این طرح همچنان حالتمند است، ولی بعد از اثبات نتیجه‌ی زیر، نشان خواهیم داد که چگونه می‌توان از این امر اجتناب کرد.

که $pk^i = pk_{m|j} = pk'_{m|j}$ (چنین i بر اساس حداقل بودن j وجود دارد). اگر $i = i^*$ خروجی $(pk'_{m|j} \parallel pk'_{m|j}, \sigma'_{m|j})$ را ارائه می‌کنیم.

حالت دوم: اگر حالت اول برقرار نباشد، آنگاه $pk'_m = pk_m$. فرض کنید i به صورتی باشد که $pk^i = pk_m$. اگر $i = i^*$ ، خروجی (m, σ'_m) را ارائه می‌کنیم.

توجه کنید که l کران بالایی برای تعداد کلیدهای عمومی از Π تعیین می‌کند که برای تولید l^* امضا با استفاده از Π^* موردنیاز هستند. این امر به این دلیل است که هر امضا در Π^* نیازمند حداکثر $2n$ کلید جدید از Π (در بدترین حالت) است و یک کلید اضافی از Π به‌عنوان کلید عمومی واقعی pk_E استفاده می‌شود.

که $pk'_{m|l} = pk_{m|l}$ (چنین i بر اساس حداقل بودن i وجود دارد). اگر $i = i^*$ ، i خروجی $(pk'_{m|l}, \sigma_{m|l})$ را ارائه می‌کنیم.
حالت دوم: اگر حالت اول برقرار نباشد، آنگاه $pk'_m = pk_m$ فرض کنید i به صورتی باشد که $pk'_i = pk_m$ ، اگر $i = i^*$ ، i خروجی (m, σ_m) را ارائه می‌کنیم.

در آرایش $\Pi^{1-time}(n)$ $forge_{\mathcal{H}}(n)$ دیدگاه $\mathcal{H}^*$ اجرانده به‌عنوان یک زیررول توسط $\mathcal{H}$ داری و نیز یکسان با دیدگاه $\mathcal{H}^*$ در آرایش $Sig - forge_{\mathcal{H}, \Pi}(n)$ است.^{۱۲}

پسین احتمال اینکه $\mathcal{H}^*$ یک چل را خروجی دهد هنگامی که به‌عنوان یک زیررول توسط $\mathcal{H}$ در آن آرایش اجرا می‌شود، دقیقاً برابر با $\delta(n)$ است. با توجه به اینکه $\mathcal{H}^*$ یک چل را خروجی می‌دهد و کاملاً دو حالت ممکن توصیف‌شده در بالا را در نظر بگیرید:

حالت اول

زبانکه i^* یکخواست و مستقل از دیدگاه $\mathcal{H}^*$ است، احتمال اینکه $i = i^*$ باشد دقیقاً برابر با $1/1$ است. اگر $i = i^*$ باشد، آنگاه $\mathcal{H}$ یک امضا بر روی پیام $pk_{m|l}$ ، $pk_{m|l} \parallel pk'_{m|l}$ بر اساس کلید عمومی $pk'_i = pk'_i$ که با آن داده شده است، درخواست کرده است (و هیچ امضای دیگری درخواست نکرده است). به‌علاوه،

$$pk'_{m|l} \parallel pk'_{m|l} \neq pk_{m|l}, \quad pk'_{m|l} \parallel pk_{m|l}$$

و این حال، $\sigma'_{m|l}$ یک امضای معتبر بر روی $pk'_{m|l}$ ، $pk_{m|l}$ بر اساس pk است. بنابراین، $\mathcal{H}^*$ یک چل را در این حالت، خروجی می‌دهد.

حالت دوم

بازهم از آنجاکه i^* به‌صورت یکخواست و تصادفی انتخاب شده و مستقل از دیدگاه $\mathcal{H}^*$ است، احتمال اینکه $i = i^*$ باشد دقیقاً برابر با $1/1$ است. اگر $i = i^*$ ، آنگاه $\mathcal{H}$ هیچ امضایی بر اساس کلید عمومی $pk'_i = pk'_i$ درخواست نکرده است و این حال، σ'_m یک امضای معتبر بر روی m بر اساس $pk = pk'_i$ است.

^{۱۲} همان‌طور که اشاره کردیم، $\mathcal{H}$ هیچ‌گاه کلیدهای خصوصی را تمام نمی‌کند. یک پرسان امضای $\mathcal{H}^*$ از m کلید عمومی استفاده می‌کند؛ بنابراین، حتی اگر کلیدهای عمومی جدید برای پاسخ به تمام پرسانهایی امضای $\mathcal{H}^*$ مورد نیاز می‌بود (که عموماً به این صورت نیست)، تنها $\gamma m^1(n)$ کلید عمومی توسط $\mathcal{H}$ علاوه بر کلید عمومی اولیه pk ، مورد نیاز خواهد بود.

توجه کنید که اگران بالایی برای تعداد کلیدهای عمومی از Π تعیین می‌کند که برای تولید i^* امضا با استفاده از Π^* مورد نیاز هستند. این امر به این دلیل است که هر امضا در Π^* نیازمند حداکثر $2m$ کلید جدید از Π (در بدترین حالت) است و یک کلید اضافی از Π به‌عنوان کلید عمومی واقعی pk_E استفاده می‌شود.

متخلف PPT زیر، یعنی $\mathcal{H}$ را در نظر بگیرید که به طرح امضای یک-بار-امن Π حمله می‌کند.

متخلف $\mathcal{H}$

متخلف $\mathcal{H}$ به‌عنوان ورودی، یک کلید عمومی pk (پارامتر امنیتی n به‌صورت ضمنی است) را دریافت می‌کند.

- یک نمایه یکخواست $\{i^1, \dots, i^t\}$ را انتخاب می‌کنیم. یک فهرست $pk^1, \dots, pk^t$ از کلیدها را به‌صورت زیر ایجاد می‌کنیم:

- قرار می‌دهیم $pk^{i^*} := pk$

- برای $i^* \neq i$ ، $i \in \text{Gen}(1^n)$ را محاسبه می‌کنیم.

- $\mathcal{H}^*$ را بر روی کلید عمومی ورودی $pk = pk_E$ اجرا می‌کنیم. هنگامی که $\mathcal{H}^*$ یک امضا بر روی یک پیام m را درخواست می‌کند، موارد زیر را انجام می‌دهیم:

(۱) برای $1 \leq t \leq n$:

- اگر مقادیر $pk_{m|l}^{i^*}, pk_{m|l}^{i^*}$ هنوز تعریف نشده‌اند، آنگاه، $pk_{m|l}^{i^*}$ و $pk_{m|l}^{i^*}$ را برابر با دو کلید عمومی بعدی استفاده‌شده pk^1 و pk^{i^*+1} قرار می‌دهیم و یک امضای $\sigma_{m|l}$ را بر روی $pk_{m|l}^{i^*} \parallel pk_{m|l}^{i^*}$ بر اساس $pk_{m|l}^{i^*}$ محاسبه می‌کنیم.^{۱۳}

(۲) اگر σ_m هنوز تعریف‌نشده باشد، یک امضای σ_m را بر روی m بر اساس pk_m محاسبه می‌کنیم. آن‌ک پانوس قبلی.

(۳) $(\sigma_m, pk_{m|l}^{i^*}, pk_{m|l}^{i^*})$ را به $\mathcal{H}^*$ می‌دهیم.

- فرض کنید $\mathcal{H}^*$ یک پیام m (که برای آن، قبلاً درخواست یک امضا نکرده است) و یک امضا $(\sigma_m, pk_{m|l}^{i^*}, pk_{m|l}^{i^*})$ را خروجی می‌دهد. اگر این امضا یک امضای معتبر بر روی m باشد، آنگاه:

حالت اول: فرض کنید یک $i \in \{0, \dots, n-1\}$ وجود دارد که برای آن، $pk_{m|l}^{i^*} \neq pk_{m|l}^{i^*}$ یا $pk_{m|l}^{i^*} \neq pk_{m|l}^{i^*}$ این امر شامل حالتی می‌شود که $pk_{m|l}^{i^*}$ یا $pk_{m|l}^{i^*}$ هیچ‌گاه توسط $\mathcal{H}$ تعریف‌نشده باشند. حداقل مقدار چنین i را در نظر گرفته و i را به‌گونه‌ای در نظر بگیرید

^{۱۳} اگر $i^* \neq i$ ، آنگاه $\mathcal{H}$ می‌تواند یک امضا را بر اساس pk به‌تنهایی محاسبه کند. $\mathcal{H}$ همچنین می‌تواند یک امضای (یکتا) را بر اساس pk^{i^*} به دست آورد به این صورت که پرسان مناسب را از پیشگویی امضاکننده خود ببرد. منظور ما در اینجا همین است.

- ۱) محاسبه $F_x(w)^{1/2}$
- ۲) محاسبه $Gen(1^n, r_w) = Gen(1^n, r_w, sk_w)$ (بمعانی قبل).

به علاوه، کلید k' برای تولید مقدار r'_w استفاده می‌شود که برای محاسبه امضای m استفاده می‌شود. این شیوه یک طرح «بدون حالت» را می‌دهد که در آن، تولید کلید (و همین‌طور امضا کردن و راست زنجیره کردن) را می‌توان در زمان چندجمله‌ای انجام داد. به شکل شهودی، این روش، امن است به این دلیل که ذخیره کردن یک تابع تصادفی برابر با ذخیره تمام مقادیر r_w و r'_w مورد نیاز است و ذخیره یک تابع شبه تصادفی هم به همین اندازه خوب است. از آنجایی که اثبات رسمی برای اینکه این طرح تبدیل‌شده، امن باقی می‌ماند را به‌عنوان یک تمرین باقی می‌گذاریم.

از آنجاکه وجود توابع چکیده ساز مقاوم در برابر برخورد به شکل ضمنی نشان‌دهنده وجود توابع یک‌طرفه است (ن. ک. تمرین ۷-۴) و مورد دوم به شکل ضمنی نشان‌دهنده وجود توابع شبه تصادفی است (ن. ک. فصل هفتم)، داریم:

فصل ۱۲-۲۲

اگر توابع چکیده ساز مقاوم در برابر برخورد وجود داشته باشند، آنگاه یک طرح امضای امن (بدون حالت) وجود خواهد داشت.

بیان می‌کنیم که می‌توان طرح‌های امضای صادق در تعریف ۱۲-۲ را از روی این فرض (حادثی) ایجاد نمود که توابع یک‌طرفه وجود دارند؛ اثبات این نتیجه فراتر از گستره این کتاب است.

۷-۱۲ گواهی‌ها و زیرساخت‌های کلید عمومی

در این بخش، به‌صورت مختصر به بررسی یکی از کاربردهای اصلی امضاهای دیجیتال می‌پردازیم. توزیع امن کلیدهای عمومی، این مطلب، بحث ما در خصوص رمزنگاری کلید عمومی را کامل می‌کند. در این فصل و فصل قبلی، مشاهده کردیم که چگونه می‌توان از رمزنگاری کلید عمومی به‌دست آورد توزیع امن کلیدهای عمومی، استفاده کرد. اکنون نشان می‌دهیم که خود رمزنگاری کلید عمومی را چگونه می‌توان برای توزیع امن کلیدهای عمومی استفاده نمود. این میحت ممکن است دایره‌وار به نظر برسد ولی این گونه نیست. چیزی که نشان می‌دهیم این است که زمانی که یک کلید عمومی «یک‌تار» متعلق به یک طرف مورد اعتماد، به شکلی امن توزیع می‌شود، آن کلید را می‌توان برای «راندازی» توزیع

^{۱۲} فرض می‌کنیم که طول خروجی F به اندازه کافی طولانی است و اینکه w تا رسیدن به یک رشته طول ثابت به یکسان یک‌پارگی لایه گذاری می‌شود. این موارد نمی‌تواند در اینجا نادیده می‌گیریم.

مشاهده می‌کنیم که مشروط به اینکه m یک جمل را خروجی دهد، m یک جمل را با احتمالاً دقیقاً برابر با $1/n$ خروجی می‌دهد. این امر به این معنا است که

$$\Pr[\text{Sig} - \text{forge}_{\Pi, n}(n) = 1] = \Pr[\text{Sig} - \text{forge}_{\Pi, n}(n) = 1] / (n).$$

به این دلیل که Π یک طرح امضای یک‌پار-امن است، یک تابع ناچیز $negl$ وجود دارد که برای آن

$$\Pr[\text{Sig} - \text{forge}_{\Pi, n}(n) = 1] \leq negl(n).$$

چون Π چندجمله‌ای است، این امر به این معنا است که $\Pr[\text{Sig} - \text{forge}_{\Pi, n}(n) = 1] \leq negl(n)$ ناچیز است. ■

یک راه‌حل بدون حالت

همان‌طور که توصیف شد، امضاکننده، حالت را به‌صورت لحظه‌ای و در هنگام نیاز تولید می‌کند. با این حال، می‌توانیم تصور کنیم که امضاکننده قادر به تولید اطلاعات لازم برای تمام گروه‌های درخت «زیر قلم» و در زمان تولید کلید، شود. (یعنی، در زمان تولید کلید، امضاکننده می‌تواند کلیدهای $\{(pk_w, sk_w)\}$ و امضاهای $\{m\}$ را برای تمام رشته‌های دودویی w با طول حداکثر n تولید کند.) اگر تولید کلید به این شیوه انجام شود، امضاکننده اجباری برای به‌روزرسانی حالت خود به‌هیچ‌وجه نخواهد داشت. این مقادیر را همگی می‌توان به‌عنوان بخشی از یک کلید خصوصی (عظیم) ذخیره نمود و آنگاه، یک طرح بدون حالت را به دست می‌آوریم. البته، مشکل این رویکرد آن است که تولید همگی این مقادیر نیازمند زمان «حسابی» خواهد بود و ذخیره کردن تمام آن‌ها هم نیازمند حافظه‌ی نمایی است.

یک جایگزین برای این روش، عبارت است از ذخیره یک «مقدار تصادفی» که می‌توان از آن برای تولید مقادیر $\{(pk_w, sk_w)\}$ و $\{m\}$ در هنگام نیاز استفاده نمود به‌جای اینکه خود این مقادیر را ذخیره کنیم. یعنی، امضاکننده می‌تواند یک رشته تصادفی r_w را برای هر w ذخیره کند و هر بار که مقادیر pk_w, sk_w مورد نیاز بودند، امضاکننده می‌تواند $Gen(1^n, r_w) = Gen(1^n, r_w, sk_w)$ را محاسبه کند که نشان‌دهنده تولید یک کلید با طول n با استفاده از سکه‌های تصادفی r_w است. به شکل مشابه اگر رویه‌ی امضا کردن، احتمالاتی باشد، امضاکننده می‌تواند r'_w را ذخیره کرده و سپس قرار دهد $(pk_w, r'_w) \parallel (pk_w, r_w) = \text{Sig}_w(m)$ (با این فرض که در اینجا $n < |w|$). با این حال، تولید و ذخیره تعداد کافی رشته‌ی تصادفی همچنان نیازمند زمان و حافظه‌ی نمایی است.

یک تغییر ساده در این روش جایگزین، یک راه‌حل چندجمله‌ای را می‌دهد. به‌جای ذخیره r_w و r'_w تصادفی که در بالا پیشنهاد شد، امضاکننده می‌تواند دو کلید k, k' را برای یک تابع شبه تصادفی F ذخیره کند. در هنگام نیاز، در این حالت مقادیر pk_w, sk_w را می‌توان از طریق فرآیند دوم حلای ذیل، تولید نمود:

بیمه‌ای از مدل‌های متفاوت PKI پیشنهاد شده‌اند و ما اکنون به تعدادی از محبوب‌ترین آن‌ها نگاهی می‌کنیم. بحث ما در اینجا نسبتاً در سطح بالا خواهد بود و خوانندگان علاقه‌مند به جزئیات بیشتر می‌توانند به مراجع انتهایی این فصل مراجعه کنند.

یک مجوز دهنده گواهی یکتا

بهترین PKI یک «مجوز دهنده گواهی» (CA) یکتا را فرض می‌کند که مورد اعتماد همه است و گواهی‌ها را برای کلید عمومی همه، منتشر می‌کند. یک مجوز دهنده گواهی معمولاً یک فرد نخواهد بود بلکه با احتمال زیاد شرکتی است که فعالیت اصلی آن گواهی کردن کلیدهای عمومی است. یک سازمان دولتی است یا شاید یک بخش در یک سازمان است (هرچند در حالت آخر، CA با احتمال زیاد تنها توسط افراد درون سازمان مورد استفاده قرار خواهد گرفت). هر شخصی که می‌خواهد به خدمات CA تکیه کند، مجبور خواهد بود یک کپی قانونی از کلید عمومی CA یعنی pk_{CA} را به دست آورد. مشخص است که این گام باید به شکلی امن انجام شود چرا که اگر یک طرف مفروض یک شخصی نامرست از pk_{CA} را دریافت کند، آنگاه آن طرف ممکن است بتواند یک کپی اصلی از کلید عمومی دیگر افراد را به دست آورد. این امر به این معنا است که pk_{CA} باید بر روی یک کانال «حراز اصالت شده» توزیع شود. ساده‌ترین راه برای انجام این کار از طریق شیوه‌های فیزیکی است: برای مثال، اگر CA درون یک سازمان باشد، آنگاه هر کدام از کارمندان می‌توانند یک کپی اصلی از pk_{CA} را به صورت مستقیم از CA در اولین روز کاری خود دریافت نمایند. اگر CA یک شرکت باشد، آنگاه دیگر کاربران مجبور به رفتن به این شرکت و مثلاً دریافت یک درایو USB هستند که شامل کلید عمومی CA باشد. این گام نامناسب تنها باید یکبار انجام شود.

یک روش معمول برای یک CA برای توزیع کلید عمومی خود در عمل عبارت است از «اضافه کردن» این کلید عمومی به همراه یک نرم‌افزار دیگر برای مثال، این امر امروزه در بسیاری از مرورگرهای وب رخ می‌دهد: کلید عمومی یک CA به همراه مرورگر ارائه می‌شود و مرورگر برنامه‌ریزی می‌شود تا به صورت خودکار، گواهی‌ها را هنگام دریافت آن‌ها، واریسی کند (درواقع، مرورگرهای وب مدرن دارای کلیدهای عمومی «چندین» CA هستند که درون آن‌ها قرار گرفته است و بنابراین به شکلی دقیق‌تر در مدل «چندگانه» مورد بحث در ذیل، قرار می‌گیرند).

مکانیسمی که از طریق آن، یک CA یک گواهی را برای یک طرف مثلاً باب صادر می‌کند نیز باید با دقت بالا کنترل شود هرچند جزئیات ممکن است برای هر CA متفاوت باشند به عنوان یک مثال، باب ممکن است مجبور باشد به صورت حضوری به همراه یک کپی از کلید عمومی خود، pk_B و به همراه مدرکی که اثبات می‌کند نام وی (یا آدرس ایمل وی) همان چیزی است که ادعا می‌کند به CA مراجعه کند. تنها در این صورت است که CA، گواهی را صادر خواهد کرد.

امن تعداد داخواهی کلید عمومی دیگر، مورد استفاده قرار داد. بنابراین، حداقل در اصل، تنها نیاز است مسئله‌ی توزیع کلید امن را یکبار حل کنیم.

مفهوم کلیدی در اینجا یک «گواهی دیجیتال» است که درواقع یک امضای پیوند زنده‌ی یک هزار (موجود) به یک کلید عمومی مفروض است. به بیان عینی، فرض کنید یک طرف به نام چارلی یک زوج-کلید (pk_C, sk_C) را برای یک طرح امضای دیجیتال امن، تولید کرده است (در این بخش، تنها به طرح‌های امضای می‌پردازیم که در ترفند ۱۲-۲ صدق می‌کنند). همچنین فرض کنید که یک طرف دیگر به نام باب نیز یک زوج-کلید (pk_B, sk_B) را تولید کرده است (در بحث کنونی، این کلیدها می‌توانند کلیدهایی برای یک طرح امضا یا یک طرح رمزگذاری کلید عمومی باشند) و اینکه چارلی می‌داند که pk_B کلید عمومی باب است. آنگاه، چارلی می‌تواند امضای زیر را محاسبه کند

$$\text{cert}_{C-B} \stackrel{\text{def}}{=} \text{Sign}_{sk_C}(\text{Bob's key is } pk_B)$$

و این امضا را به باب بدهد. ما cert_{C-B} را یک «گواهی» برای کلید باب می‌نامیم که توسط چارلی صادر شده است. در عمل، یک گواهی باید بدون ابهام طرف دارای یک کلید عمومی خاصی را شناسایی کند و بنابراین یک عبارت توصیف‌کننده‌ی متحصربه‌فردتر از «باب» مورد استفاده قرار می‌گیرد مثلاً نام کامل و ایمل باب یا URL وسایت باب.

اکنون فرض کنید باب می‌خواهد با یک طرف دیگر مثلاً آلیس ارتباط برقرار کند که از قبل pk_C را می‌داند. باب می‌تواند (pk_B, cert_{C-B}) را برای آلیس ارسال کند که در ادامه می‌تواند راست آزمایی کند که cert_{C-B} واقعاً یک امضای معتبر بر روی پیام 'Bob's key is pk_B ' بر اساس pk_C است. با فرض موفقیت واریسی، آلیس اکنون می‌داند که چارلی پیام مشخص شده را امضا کرده است. آلیس به چارلی اعتماد داشته باشد، می‌تواند بپذیرد که pk_B کلید عمومی قانونی باب است.

تمام ارتباطات بین باب و آلیس می‌تواند بر روی یک کانال «نامن» و «حراز اصالت نشده» رخ دهد. اگر یک متخاصم فعال در ارسال (pk_B, cert_{C-B}) از باب به آلیس دخالت کند، آن متخاصم قادر به تولید یک گواهی معتبر متصل‌کننده‌ی باب به هیچ کلید عمومی pk_B دیگری نخواهد بود مگر اینکه چارلی قبلاً یک گواهی دیگر را امضا کرده باشد که باب را با pk_B مرتبط نماید (که در چنین حالتی، این حمله چنان حمله‌ی جدی نخواهد بود). همین‌اين موارد فرض می‌کنند که چارلی، متقلب نیست و اینکه کلید خصوصی وی به خطر نیفتاده است.

ما جزئیات فراوانی را در توصیف بالا حذف کرده‌ایم. برجسته‌ترین مورد این است که ما این مسئله را مورد بحث قرار نداده‌ایم که آلیس به چه صورت در ابتدا pk_C را می‌فهمد؛ چارلی چگونه می‌تواند اطمینان حاصل کند که pk_B کلید عمومی باب است و اینکه آلیس چگونه تصمیم می‌گیرد که آیا به چارلی اعتماد کند یا خیر. مشخص کردن کامل چنین جزئیاتی (و دیگر جزئیات) یک «زیرساخت کلید عمومی» (PKI) را تعریف می‌کند که اجازه‌ی توزیع گسترده‌ی کلیدهای عمومی را می‌دهد.

تعارف و تثبیت‌شده به همراه دیگر شرکت‌های جدیدتر است که قابلیت اعتماد به آن‌ها را به نام شخص نمود. این امر بر عهده کاربر است که به‌صورت دستی تنظیمات خود را به نام آن‌ها تایید کند یا نه.

پیکان و زنجیره‌های گواهی

پیکان دیگر به بار وارد شده به یک CA یکتا را تقلیل می‌دهد (ولی به نگرانی‌های امنیتی داشتن پیکان یکتاست یکتای نامی پردازش) استفاده از «زنجیره‌های گواهی» است. ما ایده‌ی زنجیره‌های گواهی را از ۲ راه می‌کنیم هرچند به‌سادگی می‌توان مشاهده نمود که هر چه می‌گوئیم به زنجیره‌های پیکان دایره‌ای تعمیم می‌یابد.

نوع کد چارلی که به‌عنوان یک CA عمل می‌کند، یک گواهی را برای باب مثل بحث اول ما صادر می‌کند همچنین فرض کنید که کلید باب، یعنی pk_B یک کلید عمومی برای یک طرح امضا است. باب نیز به‌نوبت خود می‌تواند گواهی‌های خود را برای طرفین دیگر صادر کند. برای مثال، باب ممکن است یک گواهی برای آلینس به شکل زیر صادر کند

$$\text{cert}_{B \rightarrow A} \stackrel{\text{def}}{=} \text{Sign}_{pk_B}(\text{'Alice's key is } pk_A \text{'})$$

اکنون اگر آلینس بخواهد با یک طرف چهارم، مثلاً دیو، که کلید عمومی چارلی را می‌داند (ولی کلید عمومی باب را نمی‌داند) ارتباط برقرار کند، آلینس می‌تواند موارد زیر را برای دیو ارسال کند.

$$pk_A, \text{cert}_{B \rightarrow A}, pk_B, \text{cert}_{C \rightarrow B}$$

دیو چه نتیجه‌ای از این موارد بگیرد؟ خب، اول اینکه می‌تواند واریسی کند که چارلی، که دیو به او اعتماد دارد و کلید عمومی او اکنون در اختیار دیو است، یک گواهی pk_B را امضا کرده است که مشخص می‌کند pk_B واقعاً متعلق به آلینس است. اگر دیو به چارلی اعتماد دارد که گواهی‌ها را تنها برای افراد مورد اعتماد صادر می‌کند، آنگاه دیو می‌تواند pk_A را به‌عنوان کلید اصلی آلینس قبول کند.

تأکید می‌کنیم که در این مثال، معانی قوی‌تری با گواهی $\text{cert}_{C \rightarrow B}$ مرتبط هستند. در بحث قبلی ما، یک گواهی به این شکل تنها بیان می‌کرد که باب کلید عمومی pk_B را در اختیار دارد. اکنون، یک گواهی بیان می‌کند که باب کلید عمومی pk_B را در اختیار دارد و همین‌طور باب برای صدور گواهی‌های دیگر، مورد اعتماد است. هنگامی که چارلی یک گواهی را برای باب صادر می‌کند که دارای این معانی قوی‌تر است، چارلی در اصل در حال «موکالت» توانایی خود برای صدور گواهی‌ها به باب است. باب اکنون می‌تواند به‌عنوان یک وکیل برای چارلی عمل کند و گواهی‌ها را به‌جای چارلی صادر کند.

با بازگشت به یک PKI مبتنی بر CA می‌توانیم یک CA «ریشه» و n مورد CA «سطح دوم» $CA_1, \dots, CA_n$ را تصور کنیم. CA ریشه می‌تواند گواهی‌ها را برای هر کدام از CA های سطح دوم صادر

در مدلی که در آن تنها یک CA یکتا وجود دارد، طرفین به‌طور کامل به این CA اعتماد دارند که گواهی‌ها را تنها در حالت مناسب صادر می‌کند؛ به همین دلیل است که ضروری است یک فرآیند راست آزمایی دقیق قبل از صدور یک گواهی مورد استفاده قرار گیرد. در نتیجه، اگر آلینس یک گواهی $\text{cert}_{C \rightarrow B}$ گواهی کننده‌ی این امر که pk_B کلید عمومی باب است را دریافت کند، آلینس این مطلب را به‌عنوان یک مطلب معتبر می‌پذیرد و از pk_B به‌عنوان کلید عمومی باب استفاده می‌کند.

مجاز دهنده‌های گواهی چندگانه

درحالی که مدلی که در آن تنها یک CA وجود دارد، ساده و جذاب است، ولی خیلی عملی نیست. یک دلیل این امر این است که خارج از یک سازمان یکتا، این احتمال کم است که «همه» به یک CA یکسان اعتماد کنند. نیازی نیست که این امر به این معنا باشد که همه فکر می‌کنند که CA معتقب است؛ این امر می‌تواند به این معنا باشد که فردی فکر می‌کند که فرآیند راست آزمایی CA ، ناکافی است (مثلاً هنگام تولید یک گواهی، CA تنها درخواست یک نوع مدرک شناسایی می‌کند درحالی که آلینس ترجیح می‌دهد که دو مدرک شناسایی مورد استفاده قرار گیرند). به‌علاوه، CA یک نقضی شکست یکتا برای کل سیستم است. اگر CA معتقب باشد یا بتوان به او رشوه داد یا حتی CA به شکلی درست از کلید خصوصی خود محافظت نکند، مشروعیت گواهی‌های صادرشده ممکن است مورد سؤال واقع شود. همچنین برای همه طرفینی که گواهی می‌خواهند داشتن ارتباط با این CA آسان نخواهد بود.

یک رویکرد برای حل تمام این مسائل آن است که به CA های متعدد تکیه کنیم. یک طرف مثلاً باب که می‌خواهد یک گواهی را بر روی کلید عمومی خود دریافت کند می‌تواند انتخاب کند که کدام CA یا CA ها یک گواهی را صادر کنند و یک طرف مثلاً آلینس که می‌خواهد یک گواهی را مشاهده کند یا حتی چندین گواهی صادرشده توسط CA های متفاوت را مشاهده نماید می‌تواند انتخاب کند که به گواهی‌های کدام CA اعتماد دارد. اینکه باب مجبور به دریافت یک گواهی از بیش از یک CA شود (به‌غیر از اینکه اندکی دردسر و هزینه برای باب دارد) ضروری نخواهد داشت ولی آلینس باید مراقبت بیشتری داشته باشد چراکه امنیت ارتباط وی در نهایت تنها به‌خوبی CA با کمترین امنیتی است که آلینس به آن اعتماد کرده است. یعنی، فرض کنید آلینس به دو CA مثلاً CA_1 و CA_2 اعتماد دارد و CA_1 توسط یک متخاصم، فاسد شده است. آنگاه هرچند این متخاصم قادر به جعل گواهی‌های صادرشده توسط CA_1 نخواهد بود ولی قادر به صدور گواهی‌های جعلی به اسم CA_1 برای هر هویت/کلید عمومی دایره‌ای خود خواهد بود. این مسئله یک مشکل واقعی در سیستم‌های کنونی است. همان‌طور که قبلاً اشاره شد، سیستم‌های عامل امروز گروهای وب معمولاً از قبل با تعداد زیادی کلید عمومی CA های مختلف تنظیم شده‌اند و تنظیمات پیش‌فرض این است که تمام این CA ها به شکل برابر مورد اعتماد قرار گیرند. با این حال، اصولاً هر شرکتی که مایل به پرداخت پول باشد می‌تواند به‌عنوان یک CA در این فهرست قرار گیرد. بنابراین، فهرست CA های از قبل تنظیم‌شده شامل تعدادی

به، برای باب ارسال کند، می‌تواند در این پایگاه داده به دنبال کلید عمومی باب بگردد. همراه با یک عمومی باب، این پایگاه داده فهرستی از تمام گواهی‌های موجود صادرشده برای کلید عمومی باب را هم ارائه می‌کند. همچنین این امکان وجود دارد که چندین کلید عمومی برای باب در این پایگاه پیدا شوند و هر کدام از این کلیدها ممکن است توسط گواهی‌های صادرشده توسط مجموعه متفاوتی از طرفین، گواهی شده باشند. بازهم در این حالت آلیس باید تصمیم بگیرد که به چه کدام از این کلیدهای عمومی اعتماد دارد و سپس از آن‌ها استفاده کند.

مثل تار اعتماد به این دلیل جذاب است که نیازمند اعتماد به هیچ مجوز دهنده مرکزی نیست. از سوی دیگر، درحالی که این مدل ممکن است برای کاربران عادی رمزگذاری کننده‌ی ایمیل خود به‌خوبی عمل کند، به نظر برای موقعیت‌هایی که در آن‌ها امنیت دارای اهمیت بیشتری است یا برای توزیع کلیدهای عمومی سازمانی (برای مثال برای تجارت الکترونیک بر روی وب) مناسب نیست. اگر کاربری بخواهد برای مثال با بانک خود ارتباط برقرار کند، این احتمال کم است که به افرادی برای گواهی کلید عمومی بانک خود اعتماد کند که آن‌ها را در یک اجلاس ملاقات کرده است و همچنین این احتمال نیز کم است که یک نماینده‌ی بانک به یک مهمانی امضای کلید ورود تا کلید بانک را مورد گواهی قرار دهد.

باطل کردن گواهی‌ها

یک مسئله‌ی مهم که هنوز به آن نپرداخته‌ایم این واقعیت است که گواهی‌ها معمولاً نباید همیشه معتبر باقی بمانند. یک کارمند ممکن است شرکت را ترک کند که در این حالت وی دیگر اجازه‌ی دریافت ارتباط رمز شده از سوی افراد درون شرکت را ندارد. کلید خصوصی یک کاربر نیز ممکن است سرتف شود که در این حالت، کاربر (با این فرض که در مورد این سرتف اطلاع دارد) می‌خواهد که یک زوج-کلید جدید را تولید کرده و کلید عمومی قبلی را از گردش خارج کند. در هر کدام از این سناریوها، نیازمند روشی برای باطل کردن گواهی‌های صادرشده‌ی قبلی هستیم.

رویکردهای استفاده‌شده برای مدیریت این مسائل، متفاوت و پیچیده هستند و ما تنها به دو ایده‌ی نسبتاً ساده اشاره می‌کنیم که تا حدودی نشان‌دهنده‌ی دو نقطه‌ی متقابل هستند (بهبود این روش‌ها یک حوزه‌ی فعال در تحقیقات امنیت شبکه در جهان واقعی است).

انتضا

یک روش برای جلوگیری از استفاده از گواهی‌ها به‌طور نامحدود این است که یک «تاریخ انتضا» را به‌عنوان بخشی از گواهی، اضافه کنیم. یک گواهی صادرشده توسط یک CA چارلی، برای کلید عمومی باب ممکن است دارای شکل زیر باشد

$$\text{cert}_{C_A-B} \stackrel{\text{def}}{=} \text{Sign}_{K_C} ('Bob's\ key\ is\ pk_B',\ date)$$

توانید که آن‌ها هم به‌نوبه‌ی خود می‌توانند گواهی‌هایی برای دیگر افراد دارای کلیدهای عمومی، صادر نمایند. این امر باعث کاهش فشار به CA ریشه خواهد شد و همین‌طور کار دیگر طرفین برای دریافت گواهی‌ها را راحت‌تر خواهد کرد (چراکه برای مثال اکنون می‌توانند با آن CA سلاح دومی ارتباط برقرار کنند که به آن‌ها نزدیک‌تر است). از سوی دیگر، مدیریت این CA های سطح دوم ممکن است دشوار باشد و حضور آن‌ها به این معنا است که اکنون نقاط حمله‌ی بیشتری در سیستم وجود دارد.

مدل «تار اعتماد»

آخرین مثال از یک PKI که موردبحث قرار می‌دهیم یک مدل کاملاً توزیعی است که هیچ نقطه‌ی اعتماد مرکزی ندارد و مدل «تار اعتماد» نامیده می‌شود. یک نسخه از این مدل توسط نرم‌افزار رمزگذاری ایمیل PGP («محرماتی نسبتاً خوب») برای توزیع کلیدهای عمومی استفاده می‌شود.

در مدل «تار اعتماد»، هر شخصی می‌تواند گواهی‌ها را برای هر فرد دیگر صادر کند و هر کاربر باید تصمیم خود در خصوص میزان اعتماد به گواهی‌های صادرشده توسط دیگر کاربران را بگیرد. به‌عنوان مثالی از چگونگی عملکرد این روش، فرض کنید که یک کاربر به نام آلیس اکنون کلیدهای خصوصی pk_A ، pk_B را برای کاربران C_A ، C_B در اختیار دارد. (در ادامه این مسئله را موردبحث قرار می‌دهیم که این کلیدها چگونه در ابتدا می‌توانند توسط آلیس به دست آیند) یک کاربر دیگر به نام باب که می‌خواهد با آلیس ارتباط برقرار کند ممکن است دارای گواهی‌های pk_A ، pk_B ، $cert_{C_A-B}$ و $cert_{C_B-B}$ باشد و این گواهی‌ها را (به همراه کلید عمومی خود، یعنی pk_B) برای آلیس ارسال خواهد کرد. آلیس نمی‌تواند $cert_{C_A-B}$ را واریسی کند (چراکه کلید عمومی C_A را در اختیار ندارد) ولی می‌تواند دو گواهی دیگر را راست آزمایی نماید. اکنون وی باید تصمیم بگیرد که تا چه حد به C_A و C_B اعتماد دارد. وی ممکن است تصمیم بگیرد pk_B را بپذیرد اگر به وضوح به C_A اعتماد داشته باشد یا اگر به حدی کمتر به C_A و همین‌طور C_B اعتماد داشته باشد. (برای مثال، وی ممکن است این امر را محتمل بداند که C_A یا C_B متقلب باشند، ولی احتمال متقلب بودن هر دو آن‌ها را کم بداند).

در این مدل، همان‌طور که توصیف شد، از کاربران انتظار می‌رود تا هر دو کلید عمومی دیگر طرفین و همین‌طور گواهی‌های بر روی کلید عمومی خود را جمع‌آوری کنند. در رابطه با PGP ، این امر در «مهمانی‌های امنای کلید» انجام می‌شد که در آن، کاربران PGP دورهم (مثلاً در یک اجلاس) جمع می‌شدند، کی‌های اصلی از کلیدهای عمومی خود را به هم می‌دادند و برای یکدیگر گواهی صادر می‌کردند. به‌طور کلی، کاربران در یک مهمانی امنای کلید ممکن است همدیگر را نشناسند ولی می‌توانند برای مثال گواهی‌نامه‌ی رانندگی یک کاربر را مشاهده کرده و سپس یک گواهی برای کلید عمومی وی را بپذیرند یا صادر کنند.

کلیدهای عمومی و گواهی را همچنین می‌توان در یک پایگاه داده‌ی مرکزی ذخیره نمود و این امر برای PGP انجام می‌شود (ن. ک. <http://pgp.mit.edu>) هنگامی که آلیس می‌خواهد یک پیام رمز

پایه‌های باطل‌شده، تولید می‌کند و *CRL* و تاریخ جاری را امضا می‌نماید. در ادامه، *CRL* امضا شده به صورت گسترده توزیع می‌شود یا به هر صورت در دسترس واریس کنندگان بالقوه قرار می‌گیرد. اکنون، راست آزمایی یک گواهی نیازمند بررسی معتبر بودن امضای موجود در گواهی، بررسی عدم وجود شماره‌سرال آن گواهی بر روی جدیدترین فهرست ابطال و واریس امضای *CA* بر روی خود فهرست ابطال خواهد بود.

در این رویکرد به شیوه‌ی توصیف‌شده در اینجا، یک تأخیر حداکثر یک‌روزه قبل از ابطال یک گواهی وجود دارد. این امر انعطاف‌پذیری بیشتری را در مقایسه با رویکردی که تنها مبتنی بر تاریخ‌های انقضا است، فراهم می‌کند.

۸-۱۲- قرار دادن همه‌چیز در کنار هم- SSL/TLS

به عنوان جمع‌بندی آنچه تاکنون در این کتاب بررسی کرده‌ایم، پروتکل «امنیت لایه‌ی انتقال» (*TLS*) را مورد بحث قرار می‌دهیم که به صورت گسترده برای امن نمودن ارتباطات بر روی وب استفاده می‌شود. *TLS* پروتکل استفاده‌شده توسط مرورگر شما در زمان اتصال به یک وبسایت یا استفاده از *https* به جای *http* است. باید بیان کنیم که هدف ما در اینجا تمرکز بر رمزنگاری مبتنی استفاده‌شده در هسته‌ی پروتکل *TLS* است و نه دیگر ابهامی که هرچند از دیدگاه امنیت شبکه دارای جذابیت هستند ولی برای اهمیت‌های ما در اینجا ابهامی ندارند. به شکل معمول، ما بخش‌هایی از این پروتکل را اندکی ساده و مجرد نموده‌ایم تا نکته‌ی اصلی را بیان کنیم و توصیف ما نباید برای پیاده‌سازی مورد نظر قرار گیرد. در نهایت، ما به صورت رسمی امنیت را برای این پروتکل اثبات یا ادعا نمی‌کنیم؛ در واقع، یک تحلیل رسمی از *TLS* موضوعی برای تحقیقات فعال است.

TLS یک پروتکل استانداردشده بر اساس یک پیش‌آگهی با نام *SSL* (یا لایه‌ی درجه‌ی امن) است که توسط نت اسکپت در اواسط دهه‌ی ۱۹۹۰ توسعه یافت؛ آخرین نسخه‌ی در دسترس آن *SSL 3.0* است. *TLS* نسخه‌ی ۱.۰ در سال ۱۹۹۹ منتشر شد، در سال ۲۰۰۰ به نسخه‌ی ۱.۱ به‌روزرسانی شد و دوباره در سال ۲۰۰۸ به نسخه‌ی ۱.۲ (نسخه‌ی جاری) به‌روزرسانی گردید. در زمان نگارش این کتاب، تقریباً ۵۰ درصد از وبسایت‌ها همچنان از *TLS 1.0* به‌جای نسخه‌ی جدیدتر استفاده می‌کنند؛ تمامی مرورگرهای وب اصلی از *TLS 1.2* پشتیبانی می‌کنند، هرچند در بعضی موارد، نسخه‌های قدیمی‌تر *TLS* به صورت پیش‌فرض استفاده می‌شوند. بخش اعظمی از توصیف ما در ادامه در یک سطح بالا قرار دارد که در آن، تفاوت‌های بین نسخه‌ها اهمیت‌های برای اهداف ما ندارند. باین حال، هشدار می‌دهیم که چندین حمله‌ی شناخته‌شده علیه نسخه‌های قدیمی‌تر وجود دارند.

TLS به یک کارخواه (برای مثال، یک مرورگر وب) و یک کارکنار (برای مثال، یک وبسایت) اجازه می‌دهد تا یک مجموعه کلیدهای تشهیم شده را مورد توافق قرار دهند و سپس از این کلیدها برای رمزگذاری و احراز اصالت ارتباطات بعدی خود استفاده کنند. این پروتکل شامل دو قسمت است: یک

که در آن، *date* تاریخی در آینده است که در آن، گواهی باطل می‌شود. (برای مثال، یک سال بعد از تاریخ صدور گواهی)، هنگامی که یک کاربر دیگر این گواهی را راست آزمایی می‌کند، وی منتظر باید *pkix* را بداند بلکه باید تاریخ انقضا را هم بداند و اکنون نه تنها باید بررسی کند که امضا معتبر است بلکه باید این مسئله را هم بررسی کند که تاریخ انقضا نگذشته باشد. کاربری که یک گواهی را در اختیار دارد باید در زمان انقضای یک گواهی، با *CA* تماس بگیرد تا یک گواهی جدید برای او صادر شود. در این نقطه، *CA* هویت اشخاص کاربر را دوباره واریس کرده و یک گواهی دیگر را صادر می‌کند.

استفاده از تاریخ‌های انقضا یک راه حل بسیار کلی را برای مسائل بیان‌شده در بالا ارائه می‌کند. اگر یک کارمند شرکتی را در روز بعد از دریافت یک گواهی ترک کند و گواهی یک سال بعد از تاریخ صدور منقضی شود، آنگاه این کارمند می‌تواند از کلید عمومی خود به صورت غیرقانونی به مدت یک سال تا زمان تاریخ انقضا استفاده کند. به همین دلیل، این رویکرد معمولاً همراه با روش‌های دیگری از قبیل روش توصیف‌شده‌ی بعدی استفاده می‌شود.

فسخ

هنگامی که کارمند سازمان را ترک می‌کند یا کلید خصوصی یک کاربر سرت می‌شود، می‌خواهیم گواهی‌های صادرشده برای کلیدهای عمومی آن‌ها فوراً یا حداقل به محض امکان باطل شود. این کار را می‌توان به این صورت انجام داد که *CA* به صورت آشکار، گواهی را «فسخ» کند. برای ساده‌گی یک *CA* یکتا را فرض می‌کنیم ولی هر چه می‌گوئیم به صورت کلی تر نیز اعمال خواهد شد اگر کاربر دارای گواهی‌های صادرشده توسط چندین *CA* باشد.

روش‌های متعدد متفاوتی برای مدیریت فسخ وجود دارد. یک روش ممکن (تنها روشی که مورد بحث قرار می‌دهیم) این است که *CA* یک شماره‌سرال را در هر گواهی که صادر می‌کند در نظر بگیرد؛ یعنی، یک گواهی اکنون به شکل زیر خواهد بود:

```
cert-c-g = Signpkg ('Bob's key is pkg', ###)
```

که در آن «###» نشان‌دهنده‌ی شماره‌سرال این گواهی است. هر گواهی باید دارای یک شماره‌سرال منحصر به‌فرد باشد و *CA* اطلاعات (*Bob.pk_g*, ###) را برای هر گواهی تولیدی خود ذخیره می‌کند.

اگر کلید خصوصی یک کاربر، باب، که متناظر با یک کلید عمومی *pk_g* است، سرت شود، باب می‌تواند این مسئله را به *CA* اطلاع دهد. *CA* باید در اینجا هویت باب را راست آزمایی کند تا طالع از آن شود که یک کاربر دیگر، گواهی صادرشده برای باب را به‌دروغ باطل کند. در ادامه، *CA* پایگاه داده‌ی خود را جستجو می‌کند تا شماره‌سرال مربوط به گواهی صادرشده برای باب و *pk_g* را بیابد. مثلاً، در انتهای هر روز، یک «فهرست ابطال گواهی» (*CRL*) را با شماره‌سرال‌های تمامی

در نهایت، C مقدار $(transcript, \tau_C) \leftarrow Mac_{mk}(transcript)$ را محاسبه می‌کند که در آن، $transcript$ نشان‌دهنده‌ی تمامی پیام‌های مبادله شده بین S و τ_C تاکنون است. در ادامه، کارخواه τ_C را برای S ارسال می‌کند. (درواقع، خود τ_C رمزگذاری و احراز اصالت می‌شود به صورتی که برای ارتباط در لایه‌ی ثابت، توصیف‌شده در ذیل، انجام می‌شود.)

۴ S مقدار $(c, Decaps_{sk}(c))$ را محاسبه می‌کند که از روی آن، می‌تواند mk و k_C, k_C', k_C, k_C' را به دست آورد همان‌طور که کارخواه به دست آورد. اگر

۱ $\neq 1$ $Verf_{mk}(transcript, \tau_C)$ آنگاه S فرآیند را متوقف می‌کند. در غیر این صورت،

قرار می‌دهد $(transcript', \tau_C) \leftarrow Mac_{mk}(transcript, \tau_C)$ که در آن، $transcript'$ نشان‌دهنده‌ی تمام

پیام‌های مبادله شده بین S و τ_C تاکنون است (یعنی، شامل آخرین پیام دریافتی از C). در

ادامه، S مقدار τ_C را برای C ارسال می‌کند. (بازهم τ_C درواقع بهمانند ترافیک لایه‌ی ثبت،

مورد رمزگذاری و احراز اصالت قرار می‌گیرد.)

۵ اگر ۱ $\neq 1$ $Verf_{mk}(transcript', \tau_C)$ کارخواه فرآیند را متوقف می‌کند.

در نهایت یک اجرای موفق پروتکل پروتکل دستداد، C و S یک مجموعه از چهار کلید متقارن k_C, k_C', k_C, k_C' را تسهیم می‌کنند.

۱۲.۱۲ KEM از دو پشتیبانی می‌کند: یک KEM مبتنی بر CDH/DDH بهمانند ساختار ۱۱-۱۹ با طرح رمزگذاری مبتنی بر RSA یعنی $PKCS_{15}$ بهمنظور حملات از نوع بلاژیاکو علیه طرح در حالت دوم، کارگذار نباید یک خطای رمزگشایی را ارائه کند حتی اگر $Decaps$ در گام چهارم، شکست بخورد. به‌جای این کار، اگر رمزگشایی شکست خورد، باید یک pmk یکپارخت را انتخاب کرده و سپس با استفاده از آن مقدار به اجرای پروتکل ادامه دهد. (البته در این حالت مقدار τ_C راست آزمایی نخواهد شد و S پروتکل را متوقف می‌کند، لکنه آن است که مهاجم نمی‌تواند تشخیص دهد آیا این امر ناشی از شکست رمزگشایی بوده است یا خیر.)

برداشت شهودی برای امنیت پروتکل دستداد این است که از آنجایی که C گواهی را راست آزمایی می‌کند، می‌داند که تنها کارگذار قانونی S می‌تواند pmk و در نتیجه mk را بفهمد. بنابراین، اگر پروتکل با موفقیت به پایان برسد، C می‌داند که کلیدهای k_C, k_C', k_C, k_C' را یا S قانونی تسهیم می‌کند و اینکه هیچ متخاصمی هیچ چیزی در مورد این کلیدها نفهمیده است. فرض بر این است که این امر برقرار خواهد ماند حتی در حضور یک متخاصم فعال، هرچند یک اثبات رسمی برای این مقوله، نسبتاً پیچیده است. بااین حال، بیان می‌کنیم که کد احراز اصالت پیام بر روی رونوشت برای مطابقت از مهاجم فرد-در-سیانه برای تغییر نسخه‌ی پروتکل و بسته رمزهای (رسالی در ابتدای پروتکل دستداد استفاده می‌شود. این امر مانع مهاجم برای وادار کردن S یا C به استفاده از نسخه‌های قدیمی و ضعیف‌تر این پروتکل یا استفاده از رمزهای قانونی ضعیف یا کلیدهای کوتاه خواهد شد.

«پروتکل دستداد» که تبادل کلید توأم با احراز اصالت را برای تعیین کلیدهای تسهیم شده انجام می‌دهد و یک «پروتکل لایه‌ی ثبت» که از این کلیدهای تسهیم شده برای رمزگذاری احراز اصالت ارتباط طرفین استفاده می‌کند. هرچند TLS اجازه‌ی احراز اصالت کارخواهان نزد کارگذار را می‌دهد، این پروتکل معمولاً تنها برای احراز اصالت کارگذاران نزد کارخواهان استفاده می‌می‌شود به این دلیل که تنها کارگذاران معمولاً دارای گواهی هستند. (بعد از ایجاد یک نشست TLS ، احراز اصالت کاربر-به-کاربر- در صورت نیاز- را می‌توان در لایه‌ی کاربری پشت‌پشت شبکه برای مثال با ارسال یک گذرواژه انجام داد.)

پروتکل دستداد

اکنون روند اساسی پروتکل دستداد را توصیف می‌کنیم. در ابتدای پروتکل، کارخواه C یک مجموعه از کلیدهای عمومی CA ، یعنی $\{pk_1, \dots, pk_n\}$ را در اختیار دارد و کارگذار S دارای یک زوج-کلید (pk_C, sk_C) برای یک KEM «به همراه یک گواهی $cert_C$ صادرشده توسط یکی از CA هایی است که C کلید عمومی آن‌ها را می‌داند. برای اتصال به S طرفین گام‌های زیر را اجرا می‌کنند.

- ۱ C با ارسال یک پیام به S روند را آغاز می‌کند که شامل اطلاعاتی در مورد نسخه‌های پروتکل پشتیبانی شده توسط کارخواه، «بسته رمزهای» پشتیبانی شده توسط کارخواه (برای مثال، اینکه کارخواه اجازه‌ی کدام توابع چکیده ساز یا رمزهای قالبی را می‌دهد) و یک مقدار یکپارخت (یک «تک‌شمار» N_C) است.
- ۲ S با انتخاب آخرین نسخه‌ی پروتکلی که پشتیبانی می‌کند و همین‌طور یک بسته رمز مناسب، پاسخ می‌دهد. به‌علاوه، کلید عمومی خود pk_C و گواهی خود $cert_C$ و مقدار یکپارخت خود N_C را ارسال می‌کند.
- ۳ C بررسی می‌کند که آیا یکی از کلیدهای عمومی CA ، مثلاً pk_C ، که در اختیار دارد با CA که گواهی S را صادر کرده است متعلق است یا خیر. اگر متعلق باشد، C گواهی را راست آزمایی می‌کند (و همچنین بررسی می‌کند که متلفی با ابطال نشده باشد) و در صورت موفقیت، می‌فهمد که pk_C کلید عمومی S است. در ادامه، $Encaps_{pk_C}(c, pmk) \leftarrow$ را اجرا می‌کند. این بخش ۱۱-۱۳ تا یک متن رمز c و چیزی که یک «کلید پیش-اصلی» pmk نامیده می‌شود را به دست آورد. مقدار c را برای کارگذار ارسال می‌کند.
- ۴ pmk برای به دست آوردن شاه‌کلید mk با استفاده از یک تابع برگشتن کلید (ن.ک. بخش ۴-۶-۵) اعمال شده به pmk و N_C استفاده می‌شود. در ادامه، کارخواه یک مولد شبه تصادفی را بر mk اعمال می‌کند تا چهار کلید k_C, k_C', k_C, k_C' را به دست آورد.

«همچنین این امکان برای S وجود دارد که دارای یک کلید عمومی برای یک طرح امضا باشد و از یک کلید عمومی KEM موفق، در پروتکل زیر استفاده کند. ما این گزینه‌ی را مورد بحث قرار نمی‌دهیم.

یکای وجود دارد و به‌اشتباه فکر می‌کند که A پیام m را برای وی ارسال کرده است. اگر R به A پاسخ دهد یا به شکلی نسبت به A رفتار کند که به محتوای پیام بستگی دارد، آنگاه به شکل بالقوه می‌تواند پیام ناسلوم m را بفهمد.

یک مشکل دیگر در این طرح، هرچند تا حدودی مستقل از بحث ما در اینجا است، این است که بگر «نگارناپذیری» را فراهم نمی‌کند. یعنی، R نمی‌تواند به‌سادگی برای یک طرف ثالث اثبات کند که A پیام m را امضا کرده است. حداقل بدون افشای کلید رمزگشایی خود، (dk_A)

به‌جای این کار می‌توان یک رویکرد «احراز اصالت-سپس رمزگذاری» را امتحان کرد. در اینجا، S در ابتدا یک امضای (m) $\text{Sign}_{sk_A}(m)$ را محاسبه کرده و سپس

$$(S, \text{Enc}_{ek_R}(m \parallel \sigma))$$

را ارسال می‌کند. (توجه کنید که این شیوه، مسئله‌ای انکارناپذیری اشاره‌شده در بالا را حل می‌کند) اگر طرح رمزگذاری تنها CPA -امن باشد، آنگاه مشکلاتی از قبیل مشکلات اشاره‌شده در بخش ۵-۴ وجود خواهند داشت. بنابراین اجازه دهید فرض کنیم که یک طرح رمزگذاری CCA -امن به‌جای آن استفاده می‌شود. حتی در این حالت هم یک حمله وجود دارد که می‌تواند توسط یک R مغرب انجام شود. بعد از دریافت $((S, \text{Enc}_{ek_R}(m \parallel \sigma)))$ از S یک R مغرب می‌تواند رمزگشایی نماید تا $m \parallel \sigma$ را به دست آورد و سپس رمزگذاری دوباره انجام داده و $((S, \text{Enc}_{ek_R}(m \parallel \sigma)))$ را برای یک گیرنده‌ی دیگر R' ارسال نماید. این گیرنده‌ی R' در ادامه فکر می‌کند که S پیام m را برای او ارسال کرده است. این امر می‌تواند نتایج جدی داشته باشد برای مثال اگر m این پیام باشد که همین صد دلار به تو بدهکارم» (I owe you $\$100$).

اگر طرفین دقت بیشتری درباره‌ی چگونگی مدیریت شناسه‌ها داشته باشند می‌توان از این حملات جلوگیری کرد. هنگام رمزگذاری، فرستنده باید هویت خود را به همراه پیام در نظر بگیرد؛ هنگام امضا، طرف مغروض باید هویت گیرنده‌ی هدف را به همراه آنچه امضا می‌شود، امضا کند. برای مثال، رویکرد دوم به‌گونه‌ای تغییر داده می‌شود که S در ابتدا، $(m \parallel R)$ $\text{sign}_{sk_A}(m \parallel R) \leftarrow \sigma$ را محاسبه کرده و سپس $((S, \text{Enc}_{ek_R}(S \parallel m \parallel R)))$ را برای R ارسال می‌کند. هنگام رمزگشایی، گیرنده باید بررسی کند که مقدار رمزگشایی‌شده‌ی به‌دست‌آمده شامل هویت (فرضی) فرستنده هست یا خیر؛ هنگام واریس، گیرنده باید بررسی کند که آنچه امضا شده است شامل هویت خود او باشد. هنگام اضافه کردن هویت‌ها به این شیوه، هر دو رویکرد احراز اصالت-سپس رمزگذاری و رمزگذاری-سپس احراز اصالت، امن خواهند بود اگر یک طرح رمزگذاری CCA -امن و یک طرح امضای امن قوی (که تعریف آن موازی با تعریف ۳-۴ برای MAC ها است) استفاده شوند.

پروتکل لایه‌ی ثبت

هنگامی که کلیدها توسط C و D موردتوافق قرار گرفتند، طرفین از این کلیدها برای رمزگذاری و احراز اصالت تمام ارتباطات بعدی خود استفاده می‌کنند. C از ek_A (به ترتیب، pk_A) برای رمزگذاری (به ترتیب، احراز اصالت) تمام پیام‌هایی که برای S می‌فرستد، استفاده خواهد کرد؛ به شکل مشابه، S از pk_A و dk_A برای رمزگذاری و احراز اصالت تمام پیام‌هایی که می‌فرستد، استفاده خواهد کرد. اعداد دنباله‌ای برای جلوگیری از حملات تکرار، به‌صورت بحث شده در بخش ۳-۵-۴ استفاده می‌شوند. ۱.۲.۱ TS از یک رویکرد احراز اصالت-سپس رمزگذاری استفاده می‌کند که همان‌طور که در بخش ۳-۵-۴ مشاهده کردیم می‌تواند مشکل ساز باشد.

۹-۹-۱۲ امضای رمز

در انتهای این فصل، به شکل مختصر و غیررسمی به بحث در خصوص مسئله‌ی محرمانگی و یکپارچگی هم‌زمان در شرایط کلید عمومی می‌پردازیم. در حالی که این بحث موازی با بحث ما در بخش ۵-۴ است، این واقعیت که اکنون در شرایط کلید عمومی هستیم، چندین پیچیدگی جدید را ایجاد می‌کند.

برای سادگی، شبکه‌ای را در نظر می‌گیریم که در آن، تمام طرفین مربوطه دارای زوج-کلیدهای عمومی-خصوصی برای رمزگذاری و امضا هستند. فرض می‌کنیم که (ek_A, dk_A) نشان‌دهنده‌ی کلید رمزگذاری (عمومی) و کلید رمزگشایی (خصوصی) باشد و از (pk_A, sk_A) برای یک کلید واریس (عمومی) و کلید امضا (خصوصی) استفاده می‌کنیم. فرض می‌کنیم که تمامی طرفین، کلیدهای عمومی همه‌ی دیگر افراد را می‌دانند.

به شکل غیررسمی، هدف ما طراحی مکانیسمی است که به یک فرستنده S اجازه دهد یک پیام m را برای یک گیرنده R ارسال نماید در حالی که تضمین می‌کند که (۱) هیچ طرف دیگری در شبکه نمی‌تواند هیچ اطلاعاتی در مورد m بفهمد (یعنی، محرمانگی) و (۲) به R اطمینان داده می‌شود که پیام از سوی S آمده است (یعنی، یکپارچگی). می‌خواهیم هر دو این ویژگی‌های امنیتی را حتی علیه حملات فعال (برای مثال، حملات متن رمزمنتخب) توسط دیگر طرفین در شبکه، در نظر بگیریم.

بر اساس بحث ما در بخش ۵-۴، یک ایده‌ی طبیعی استفاده از یک رویکرد «رمزگذاری-سپس احراز اصالت» است که در آن، S مقدار $((c), \text{Sign}_{sk_S}(c))$ را برای R ارسال می‌کند، که در آن، c یک رمزگذاری از m با استفاده از کلید رمزگذاری R یعنی ek_R است. (به‌صورت آشکار هویت فرستنده را در اینجا برای راحتی بیشتر اضافه می‌کنیم) بااین‌حال، یک حمله‌ی هوشمندانه‌ی متن رمزمنتخب بدون توجه به طرح رمزگذاری استفاده‌شده، در اینجا وجود دارد. بعد از مشاهده‌ی یک ارسال به شکل بالا، یک طرف دیگر (متخاصم) A می‌تواند امضای S را جدا کرده و آن را با امضای خود جایگزین نماید، یعنی $((c), \text{Sign}_{sk_A}(A, c, \text{Sign}_{sk_S}(c)))$ را برای R ارسال کند. در این حالت، R تشخیص نخواهد داد که

پید بررسی رسمی محرمانگی و یکپارچگی ترکیبی در شرایط کلید عمومی توسط آن و همکاران [۱۰] ارائه شده است.

نبرین‌ها

۱۲-: نشان دهید که ساختار ۴-۷ برای ساختن یک MAC با طول متغیر از روی هر MAC با طول ثابت را همچنین می‌توان (با تغییرات مناسب) برای ساختن یک طرح امضا برای پیام‌های با طول دلخواه از روی هر طرح امضا برای پیام‌های با طول ثابت $n \geq n$ استفاده نمود.

۱۳-: اثبات کنید که وجود یک طرح امضای یک-بار-امن برای پیام‌های ایستی به شکل ضمنی نشان‌دهنده وجود توابع یک‌طرفه است.

۱۴-: در بخش ۱۲-۴، ۱-۴، حمله‌ای علیه طرح امضای RSA ساده نشان دادیم که در آن، مهاجم یک امضا را بر روی یک پیام دلخواه با استفاده از دو پرسنل امضا، جعل می‌کند. نشان دهید که مهاجم چگونه می‌تواند یک امضا را بر روی یک پیام دلخواه با استفاده از یک پرسنل امضای «یکتا» جعل کند.

۱۴-۴: فرض کنید که مسئله RSA ، سخت است. نشان دهید که طرح امضای RSA در تعریف ضعیف زیر برای امنیت صدق می‌کند: یک مهاجم، کلید عمومی (N, e) و یک پیام یکپارچه $m \in \mathbb{Z}_N^*$ را دریافت می‌کند. متخاصم موفق می‌شود اگر بتواند یک امضای معتبر بر روی m را بدون انجام هیچ پرسنل امضایی، خروجی دهد.

۱۴-۵: یک رویکرد دیگر (بغیر از چکیده‌سازی) که برای ساختن امضاهای مبتنی بر RSA امن امتحان شده است عبارت است از «کدگذاری» پیام قبل از اعمال جایگشت RSA در اینجا، امضاکننده یک تابع کدگذاری عمومی $Z_N^* \rightarrow \mathbb{Z}_N^*$ را به عنوان بخشی از کلید عمومی خود، مشخص می‌کند و امضا بر روی یک پیام m برابر با $[enc(m)^d \bmod N]$ است.

(الف) راست آزمایی در RSA کدگذاری شده به چه صورت انجام می‌شود؟

(ب) این مسئله را مورد بحث قرار دهید که چرا انتخاب مناسب تابع کدگذاری برای $\|N\|$ مانع «حمله‌ای بدون پیام» توصیف‌شده در بخش ۱۲-۴ نخواهد شد.

(ج) نشان دهید که RSA کدگذاری شده، ناامن خواهد بود اگر $\|m\| \leq x \cdot 10^{-6}$.

باشد (که در آن، $\|N\| \leq x \cdot 10^{-6}$ و $\|m\| \leq x \cdot 10^{-6}$ و $\|m\| \leq x \cdot 10^{-6}$ فرض کنید $x = 10^{-6}$).

(د) نشان دهید که RSA کدگذاری شده برای $m \in \mathbb{Z}_N^*$ و $\|m\| \leq x \cdot 10^{-6}$ (که در آن، $\|N\| \leq x \cdot 10^{-6}$) ناامن خواهد بود. فرض کنید $x = 10^{-6}$.

مراجع و مطالعه بیشتر

مطالعات اولیه‌ای برجسته در خصوص امضاها شامل مطالعات دینی و هلمن [۵۸]، رابین [۱۴۴، ۱۴۵]، ریوست، شامیر و ادلمن [۱۴۸] و گلدواسر، میکالی و یائو [۸۲] است. برای بحث گسترده در خصوص طرح‌های امضا فراتر از آنچه در اینجا پوشش داده شد، ن. ک. [۹۸].

گلدواسر، میکالی و ریوست [۸۱] مفهوم جعل ناپذیری وجودی را تحت یک جمله‌ی پیام منتخب وقتی تعریف کرده و همچنین اولین ساختار از یک طرح امضای حالتمند صادق در این تعریف را ارائه کردند. گلدیش [۷۴] رویکردی را برای بی‌حالت نمودن طرح گلدواسر-میکالی-ریوست پیشنهاد داد و ما اصولاً از ایده‌های گلدیش در بخش ۱۲-۶-۳ استفاده کرده‌ایم.

امضاهای RSA ساده به اولین مقاله‌ی RSA بازمی‌گردند [۱۴۸]. $RSA - FDH$ توسط بل ابر و روگاری در مقاله‌ی اصلی خود پیرلومن مدل پیشگوی تصادفی [۲۱] پیشنهاد شد. هرچند ریشه‌ی ایده‌ی (بدون اثبات) استفاده از یک تابع چکیده ساز رمزنگاری برای جلوگیری از حملات جبری را می‌توان در رابین [۱۴۵] یافت. یک بهبود [۲۳] بعداً به عنوان بخشی از $PKCS\#1$ استانداردسازی شد که در <http://www.emc.com/emc-plus/rsa-labs> در دسترس است.

تبدیل فیات-شامیر [۶۵] و طرح امضای اشنور [۱۵۲] هر دو به اواخر دهه‌ی ۱۹۸۰ بازمی‌گردند. هرچند اثبات ما برای قضیه‌ی ۱۲-۱۰ از [۱۱] گرفته شده و اثبات ما برای قضیه‌ی ۱۲-۱۱ ملهم از [۲۰] بوده است. استانداردهای DSA و $ECDSA$ در [۱۳۲] توصیف شده‌اند.

طرح امضای لمپورت در ۱۷ منتشر شد [۱۱۱]، هرچند قبلاً در [۵۸] توصیف شده بود. یک ساختار مبتنی بر درخت که دارای ماهیت مشابه با ساختار ۱۲-۲۰ است توسط مرکل [۱۲۲، ۱۲۳] پیشنهاد شد. هرچند یک رویکرد مبتنی بر درخت در [۸۱] هم استفاده شده است. ناائو و یونگ [۱۲۸] نشان دادند که جایگشت‌های یک‌طرفه برای ساختن امضاهای یک-بار-امن که می‌توانند پیام‌های با طول دلخواه را امضا کنند، کافی هستند و این امر توسط رابین [۱۵۱] بهبود یافت که نشان داد توابع یک‌طرفه کافی هستند (همچنین ن. ک. [۹۹]) همان‌طور که در بخش ۱۲-۶-۳ مشاهده کردیم. امضاهای یک-بار-امن از این دست را می‌توان برای ساختن طرح‌های امضای امن مورد استفاده قرار داد که به شکل ضمنی نشان می‌دهد که توابع یک‌طرفه برای وجود امضاهای امن (بدون حالت) کافی هستند.

مفهوم گواهی‌ها اولین بار توسط کوئنفلتر [۱۰۷] در رساله‌ی کارشناسی او توصیف شدند. زیرساخت‌های کلید عمومی با جزئیات دقیق‌تر در [۱۰۲، فصل ۱۵] مورد بحث قرار گرفته‌اند. همچنین ن. ک. [۶۲، ۳] جزئیات بیشتر در خصوص پروتکل TLS را می‌توانید برای مثال در [۱۴۵، ۱۰۲]

(ه) بخش‌های (ج) و (د) را برای e دلخواه حل کنید.

۱۲-۹: یک گونه از تبدیل فیات-شامیر را در نظر بگیرید که در آن، امضا به جای (r, s) برابر با (I, S) است و راست آزمایی به شکل طبیعی تغییر می‌کند. نشان دهید که اگر طرح شناسایی مبنا، امن باشد، آنگاه طرح امضای به‌دست‌آمده در اینجا هم امن خواهد بود.

۱۲-۷: یک گونه از DSA را در نظر بگیرید که در آن، فضای پیام برابر با $\mathbb{Z}_q$ بوده و H حذف شده است. (شماره‌ای، مؤلفه‌ی دوم امضا اکنون برابر با $[k^{-1} \cdot (m + xr) \bmod q]$ است) s نشان دهید که این گونه، امن نیست.

۱۲-۸: فرض کنید f یک جایگشت یک‌طرفه باشد. طرح امضای زیر برای پیام‌های موجود در مجموعه‌ی $\{1, \dots, n\}$ را در نظر بگیرید:

- برای تولید کلیدها، $x \in \{0, 1\}^n$ را بخواهت را در نظر گرفته و قرار دهید $y: y = f^{(n)}(x)$ در آن، $f^{(0)}(x)$ به تکرار i -همه‌ای f اشاره می‌کند و $x = f^{(1)}(x)$ کلید عمومی برابر با y و کلید خصوصی برابر با x است.
- برای امضای پیام $m \in \{1, \dots, n\}$ ، خروجی $f^{(n-m)}(x)$ را ارائه کنید.
- برای راست آزمایی امضای σ بر روی پیام i بر اساس کلید عمومی y بررسی کنید که آیا $y = f^{(0)}(\sigma)$ ؟

(الف) نشان دهید که طرح بالا یک طرح امضای یک-بار-امن نیست. با داشتن یک امضا بر روی یک پیام i ، برای کدام پیام‌های j متخاصم می‌تواند یک جعل را خروجی دهد.
(ب) اثبات کنید که هیچ متخاصم PPT با داشتن یک امضا بر روی i قادر به اعلام یک جعل بر روی یک پیام مغرض $j > i$ مگر با احتمال ناچیز نیست.
(ج) پیشنهاد دهید این طرح را چگونه می‌توان تغییر داد تا یک طرح امضای یک-بار-امن به دست آید.

راهتمانی: دو مقدار y را در کلید عمومی در نظر بگیرید.

۱۲-۹: یک طرح امضای یک-بار-امن در شروط زیر (به‌صورت غیررسمی) صدق می‌کند: با داشتن یک امضای σ بر روی یک پیام m' ، اعلام $(m', \sigma) \neq (m, \sigma)$ عملی است که برای آن، σ یک امضای معتبر بر روی m است (وجه کنید که $m' = m$ اجازه داده می‌شود).

(الف) تعریفی رسمی از امضاهای یک-بار-امن قوی ارائه کنید.

(ب) با فرض وجود توابع یک‌طرفه، یک تابع یک‌طرفه را نشان دهید که برای آن، طرح لمبورت یک طرح امضای یک-بار-امن قوی، «نیست».

(ج) یک طرح امضای یک-بار-امن قوی را بر اساس یک فرض استفاده‌شده در این کتاب، ایجاد کنید.

پسای: از یک تابع یک‌طرفه‌ی خاص در طرح لمبورت استفاده کنید.

۱۱-۱۱: طرح امضای لمبورت را در نظر بگیرید. متخاصمی را توصیف کنید که امضاهایی بر دو پر خطی خود را به دست می‌آورد و سپس امضاهایی بر هر پیامی که مایل باشد، جعل می‌کند.

۱۱-۱۱: طرح لمبورت از $2l$ مقدار در کلید عمومی برای امضای پیام‌های با طول l استفاده می‌کند. پیامی را در نظر بگیرید که در آن، کلید خصوصی شامل $2l$ مقدار $x_1, \dots, x_{2l}$ و کلید خصوصی $2l$ مقدار $y_1, \dots, y_{2l}$ یا $y_i = f(x_i)$ است. یک پیام $m \in \{0, 1\}^l$ به شکل یک-به-یک به یک $\{x_i\}$ را امضا می‌کند. اثبات کنید که این روش، یک طرح امضای یک-بار-امن را می‌دهد. طول پیام ماگزیمم، l' ، که این طرح پشتیبانی می‌کند، کدام است؟

۱۲-۱۲: در انتهای بخش ۱۲-۶، ۳ نشان دادیم که چگونه یک تابع شبه تصادفی را می‌توان برای بدون جات نمونه‌ی ساختار ۱۲-۲ استفاده نمود. آیا رویکردی مشابه برای طرح رمزنگاری توصیف‌شده در بخش ۱۲-۶ عمل خواهد کرد؟ اگر نه، یک ساختار و اثبات ارائه کنید. اگر خیر، توضیح دهید که چرا و طرح را تغییر دهید تا یک گونه‌ی بدون حالت به دست آید.

۱۲-۱۳: قضیه‌ی ۱۲-۲۲ را اثبات کنید.

۱۲-۱۳: فرض کنید ابطال گواهی‌ها به شکل زیر مدیریت می‌شود: هنگامی که یک کارور، باب، ادعا می‌کند که کلید خصوصی متناظر با کلید عمومی وی، pk_B ، سرقت شده است، کارور بیان این امر را که بر اساس pk_B امضا شده است برای CA ارسال می‌کند. بعد از دریافت چنین پیام امضاشده‌ای، CA گواهی مربوطه را باطل می‌کند.

توضیح دهید که چرا ضروری نیست که CA هویت باب را در این حالت، بررسی کند. علی‌الخصوص، توضیح دهید که چرا اینکه یک کلید خصوصی باب را سرقت است بتواند امضاها را بر اساس pk_B جعل کند، مایه‌ی نگرانی نیست.