مجله علمی- پژوهشی "پدافند الکترونیکی و سایبری"
ویژه‌نامه کنفرانس بین‌المللی ترکیبیات، رمزنگاری و محاسبات، ص ۱۵۰ -۱۴۳

درستی‌یابی پروتکل‌های رمزنگاری با استفاده از برنامه‌سازی منطق

مصطفی زارع خورمیزی*

استادیار، دانشکده ریاضی و علوم کامپیوتر دانشگاه دامغان

(دریافت: ۹۵/۰۷/۰۵، پذیرش: ۹۶/۰۳/۰۶)

چکیده

در [۱] برونو بلنچت روشی برای درستی‌یابی پروتکل‌های رمزنگاری براساس نمایش مجرد پروتکل‌ها با بندهای هورن ارائه داده است. این روش کاملاً خودکار و کارآمد است و توانایی بررسی تعداد نشست‌ها و فضای پیام نامحدود را دارد. این روش اولیه‌های رمزنگاری مختلف که با قواعد بازنویسی یا معادلات تعریف می‌شوند را پشتیبانی می‌کند. در این مقاله، این روش پیاده‌سازی شده است. اگرچه در این مقاله روی محرمانگی تمرکز می‌کنیم ولی این روش می‌تواند ویژگی‌های امنیتی دیگر نظیر احراز هویت و هم‌ارزی پردازه‌ها را نیز اثبات کند. آزمایش‌ها نشان می‌دهد با این ابزار بسیاری از پروتکل‌ها در زمان کم‌تر از یک ثانیه و با حافظه کمتر از دو مگابایت قابل درستی‌یابی است.

واژه‌های کلیدی: درستی‌یابی خودکار، پروتکل‌های رمزنگاری، بندهای هورن، محرمانگی

۱- مقدمه

پروتکل‌های رمزنگاری[۱] می‌توانند توسط یک رده‌یافت مبتنی بر بندهای هورن[۲] درستی‌یابی شوند؛ هدف اصلی این رده‌یافت اثبات خواص امنیتی پروتکل‌ها در مدل دولو-یائو[۳] به‌صورت کاملاً خودکار بدون محدودیت در تعداد نشست‌ها یا فضای پیام پروتکل‌هاست. برخلاف حالت تعداد نشست‌های محدود که در آن تصمیم‌پذیری قابل اثبات است، حالت تعداد نشست‌های نامحدود در مدل قابل قبولی از پروتکل‌ها تصمیم‌ناپذیر است [۲]. راه‌حل‌های ممکن برای حل این مشکل بر دخالت کاربر، مجاز دانستن عدم توقف و اعمال تقریب‌های دارای صحت تکیه دارند (در این حالت، روش تمامیت ندارد یعنی ویژگی‌های امنیتی درست همواره قابل اثبات نیستند). روش اثبات قضیه [۳] و روش‌های مبتنی بر منطق‌های شناختی نظیر منطق BAN [۴] بر دخالت کاربر یا اثبات‌های دستی تکیه دارند. روش تایپ[۴] [۵-۷] در حالت کلی بر تفسیر کاربر تکیه دارد و ناتمام است. روش فضاهای استرند[۵] [۸] و توابع رتبه [۹] نیز روش‌هایی را فراهم می‌آورند که می‌تواند تعداد نشست‌های نامحدود را پشتیبانی کند ولی ناتمام است.

روش‌های زیادی بر نمایش مجرد[۶] دارای صحت تکیه دارند [۱۰]: این روش‌ها معمولاً با محاسبه تقریب دست بالا از دانش مهاجم[۷] در بررسی امکان حمله‌ها دست بالا می‌گیرند. این کار دست‌یابی به روش‌های کاملاً خودکار ولی ناتمام را امکان‌پذیر می‌سازد. رده‌یافت بندهای هورن یکی از این روش‌هاست. این روش ابتدا توسط ویدن باخ[۸] [۱۱] معرفی شد. مقاله حاضر نسخه‌ای از این روش و توسیع‌های آن ارائه می‌کند که توسط بلنچت توسعه داده شده است. در این روش، پیام‌ها با عبارت M نشان داده می‌شوند؛ حقیقت attacker(M) به این معنی است که مهاجم می‌تواند پیام M را داشته باشد؛ بندهای هورن (یعنی قاعده‌های برنامه‌نویسی منطق) قواعد ایجابی بین این حقیقت‌ها را ارائه می‌کنند.

یک الگوریتم تجزیه کارآمد مشخص می‌کند که آیا یک حقیقت از بندها قابل استنتاج است یا نه، این می‌تواند برای اثبات خواص امنیتی به‌کار رود. به‌ویژه، هنگامی که attacker(M) از بندها قابل استنتاج نباشد، مهاجم نمی‌تواند M را داشته باشد، یعنی، M محرمانه است. این روش ناتمام است زیرا تعداد تکرارها از هر عمل در پروتکل را نادیده می‌گیرد. (بندهای هورن می‌توانند

* رایانامه نویسنده مسئول: mostafazaare@du.ac.ir

1- cryptographic protocols
2- Horn clause
3- Dolev-Yao
4- typing
5- strand space
6- abstrac representation
7- attacker
8- Weidenbach

هر تعداد از دفعات به‌کار برده شوند.) این نمایش مجرد کلید اجتناب از محدودیت در تعداد اجراهای پروتکل است. این روش دارای صحت است، به این معنی که اگر درست‌یاب رخنه‌ای در پروتکل پیدا نکند، آن‌گاه رخنه‌ای وجود ندارد. بنابراین، درست‌یاب تضمین امنیتی واقعی فراهم می‌سازد. در مقابل، ممکن است یک حمله نادرست علیه پروتکل ارائه دهد. اما، حمله‌های نادرست در عمل چنان‌که آزمایش‌ها نشان می‌دهد، نادر هستند. توقف در حالت کلی تضمین‌شده نیست، اما روی زیرکلاس خاصی از پروتکل‌ها تضمین می‌شود و با تقریب‌های اضافی می‌تواند در همه حالت‌ها به‌دست آید.

بدون این تقریب‌های اضافی، اگرچه این روش همواره متوقف نمی‌شود و ناتمام است، اما در عمل تعادل خوبی فراهم می‌آورد: این روش در بسیاری از حالت‌ها متوقف می‌شود و بسیار کارآمد و دقیق است. می‌تواند رده وسیعی از اولیه‌های رمزنگاری که با قواعد بازنویسی[1] یا معادلات تعریف می‌شوند نظیر رمزنگاری کلید- مشترک و کلید- عمومی (رمزگذاری و امضاءها)، توابع چکیده‌ساز و توافق کلید دفی-هلمن[2] را پشتیبانی کند. می‌تواند خواص امنیتی مختلف (محرمانگی، احراز هویت و هم‌ارزی پردازش‌ها) را ثابت کند. در این مقاله بر محرمانگی تمرکز می‌کنیم.

در بخش ۲ نمایش پروتکل با جزئیات ارائه می‌شود. در بخش ۳ الگوریتم تجزیه توصیف می‌شود. در بخش ۴ نتایج برخی آزمایش‌ها بیان می‌شود و بخش ۵، بخش نتیجه‌گیری است.

۲- نمایش مجرد پروتکل‌ها با بندهای هورن

یک پروتکل با مجموعه‌ای از بندهای هورن نمایش داده می‌شود؛ نحو این بندها در شکل (۱) نشان داده شده است.

عبارت‌ها	$M.N ::=$
متغیر	x
نام	
کاربرد تابع	$f(M_1. \dots . M_n)$
حقیقت	$F ::= p(M_1. \dots . M_n)$
بند هورن	$R ::= F_1 \wedge \dots \wedge F_n \Rightarrow F$

شکل (۱): نحو نمایش پروتکل

در این شکل، x روی متغیرها، a روی نام‌ها، fروی نمادهای تابعی و p روی نمادهای محمولی تغییر می‌کنند. عبارات M پیام‌هایی که بین شرکت‌کننده‌ها در پروتکل مبادله می‌شود را نمایش می‌دهند. یک متغیر می‌تواند هر عبارتی را نمایش دهد. نام‌ها مقدارهای اتمی نظیر کلیدها و نانس‌ها (اعداد تصادفی) را نمایش می‌دهند. هر شرکت‌کننده توانایی تولید نام‌های جدید را دارد: نام‌های جدید در هر اجرای پروتکل تولید می‌شوند. در این‌جا، نام‌های تولیدشده به‌عنوان تابع‌هایی از پیام‌هایی که قبلاً توسط شرکت‌کننده‌ای که نام را تولید می‌کند دریافت شده است، درنظر گرفته می‌شوند. بنابراین، نام‌ها تنها هنگامی‌که پیام‌های قبلی متفاوت باشند، متمایز هستند. کاربردهای تابع $f(M_1. \dots . M_n)$ عبارت‌ها را می‌سازد: مثال‌های تابع‌ها رمزگذاری و تابع‌های چکیده‌ساز هستند. یک حقیقت $F = p(M_1. \dots . M_n)$ یک خاصیت از پیام‌های M_1، ...، M_nرا بیان می‌کند. چندین محمول p می‌تواند به کار رود اما برای اولین مثال از محمول attacker استفاده می‌کنیم، به‌طوریکه حقیقت attacker(M) به این معنی است که «مهاجم می‌تواند پیام M را داشته باشد». بند $R = F_1 \wedge \dots \wedge F_n \Rightarrow F$ به این معنی است که، اگر همه حقیقت‌های F_1، ...، F_n درست باشند، آن‌گاه F نیز درست است. یک بند بدون هیچ فرضی $\Rightarrow F$ برای سادگی با F نشان داده می‌شود.

به‌عنوان مثال جاری در این مقاله، از پروتکل دست‌دادن[3] ساده زیر استفاده می‌کنیم:

پیام ۱. $A \to B: \{|[k]_{sk_A}|\}^a_{pk_B}$

پیام ۲. $B \to A: \{|s|\}^s_k$

sk_A کلید خصوصیA، pk_Aکلید عمومیA، sk_Bکلید خصوصی B و pk_B کلید عمومی B را نشان می‌دهد.

۲-۱- نمایش اولیه‌ها

اولیه‌های رمزنگاری با تابع‌ها نمایش داده می‌شوند. به‌عنوان مثال، رمزگذاری کلید عمومی را با تابع pencrypt$(m.pk)$ نشان می‌دهیم که دو شناسه دارد: پیام m که باید رمزگذاری شود و کلید عمومی.pk یک تابع pk وجود دارد که کلید عمومی را از روی کلید خصوصی می‌سازد. (هم‌چنین می‌توانیم دو تابع pk و sk داشته باشیم که از یک راز به‌ترتیب کلیدهای عمومی و خصوصی را می‌سازند.) کلید خصوصی با یک نام نمایش داده می‌شود که هیچ شناسه‌ای ندارد (یعنی، فقط یک نسخه از این نام

1- rewrite
2- Deffie-Hellman
3- handshake

وجود دارد)$sk_A[\,]$ برای A و $sk_B[\,]$ برایB. بنابراین $pk_A = \mathrm{pk}(sk_A[\,])$و $pk_B = \mathrm{pk}(sk_B[\,])$.

به‌طور کلی‌تر، دو نوع از تابع‌ها درنظر می‌گیریم: سازنده‌ها و مخرب‌ها. سازنده‌ها تابع‌هایی هستند که صریحاً در عبارت‌هایی که پیام‌ها را نمایش می‌دهند، ظاهر می‌شوند.

برای نمونه، pencryptو pk سازنده هستند. مخرب‌ها عبارت‌ها را دست‌کاری می‌کنند. یک مخرب g با یک مجموعه‌ی $\mathrm{def}(g)$ از قاعده‌های بازنویسی به‌صورت $g(M_1.\ldots.M_n) \to M$ تعریف می‌شود که M_1، ...، M_n و Mعبارت‌هایی هستند که تنها شامل متغیرها و سازنده‌ها هستند و متغیرهای M همگی در M_1، ...، M_n ظاهر می‌شوند. برای مثال، رمزگشایی pencrypt یک مخرب است که با

$$\mathrm{pencrypt}\big(\mathrm{pencrypt}\big(m.\mathrm{pk}(sk)\big).sk\big) \to m$$

تعریف می‌شود. این قاعده بازنویسی این‌که رمزگشایی یک متن رمز با کلید خصوصی متناظر، متن ساده را به‌دست می‌دهد، را مدل می‌کند.

تابع‌های دیگر به‌طور مشابه تعریف می‌شوند:

- برای امضاءها، از یک سازنده sign استفاده می‌کنیم و $\mathrm{sign}(m.sk)$ به معنای امضای پیام m با کلید خصوصی sk است. یک مخرب getmess که با $\mathrm{getmess}\big(\mathrm{sign}(m.sk)\big) \to m$ تعریف می‌شود پیام را بدون امضاء آن به‌دست می‌دهد، و $\mathrm{checksign}\big(\mathrm{sign}(m.sk).\mathrm{pk}(sk)\big) \to m$ پیام را برمی‌گرداند تنها اگر امضاء معتبر باشد.
- رمزگذاری کلید- مشترک یک سازنده sencrypt و رمزگشایی یک مخرب sdecrypt است که با $\mathrm{sdecrypt}(\mathrm{sencrypt}(m.k).k) \to m$

تعریف می‌شوند.

- یک تابع چکیده‌ساز یک- طرفه با یک سازنده h نمایش داده می‌شود (و هیچ مخربی ندارد).
- چندتایی‌های n موضعی با سازنده $(-.\ldots.-)$ نمایش داده می‌شوند و n مخرب $i\mathrm{th}_n$ با $i \in \{1.\ldots.n\}$ و $i\mathrm{th}_n((x_1.\ldots.x_n)) \to x_i$ تعریف می‌شوند. چندتایی‌ها می‌توانند برای نمایش ساختارهای داده مختلف در پروتکل‌ها به‌کار روند.

قواعد بازنویسی روش انعطاف‌پذیری برای تعریف اولیه‌های رمزنگاری بسیاری پیشنهاد می‌کنند که می‌تواند با استفاده از معادلات بیش‌تر گسترش یابد.

۲-۲- نمایش توانایی‌های مهاجم

فرض می‌کنیم که پروتکل در حضور یک مهاجم اجرا می‌شود که می‌تواند همه پیام‌ها را شنود کند، از پیام‌هایی که دریافت کرده است پیام‌های جدید محاسبه کند و هر پیامی که بتواند بسازد را ارسال کند. این مدل دولو-یائو [۱۲] نامیده می‌شود.

ابتدا توانایی‌های محاسباتی مهاجم را کدگذاری می‌کنیم. کدگذاری پروتکل در بخش ۲-۳ انجام می‌شود. مهاجم در طول محاسبات خود می‌تواند همه سازنده‌ها و مخرب‌ها را به‌کار برد. اگر f یک سازنده n موضعی باشد، بند زیر را خواهیم داشت:

$$\mathrm{attacker}(x_1) \wedge \ldots \wedge \mathrm{attacker}(x_n) \Rightarrow \mathrm{attacker}\big(f(x_1.\ldots.x_n)\big)$$

اگر g یک مخرب باشد، برای هر قاعده بازنویسی $g(M_1.\ldots.M_n) \to M$ در $\mathrm{def}(g)$، بند زیر را داریم:

$$\mathrm{attacker}(M_1) \wedge \ldots \wedge \mathrm{attacker}(M_n) \Rightarrow \mathrm{attacker}(M)$$

مخرب‌ها هرگز در بندها ظاهر نمی‌شوند، آن‌ها توسط تطابق الگو روی پارامترهایشان (در این‌جا M_1، ...، M_n) در فرضیات بند و تولید نتیجه آن‌ها در حکم کدگذاری می‌شوند. در حالت خاص رمزگذاری کلید- عمومی، به

$$\begin{aligned}&\mathrm{attacker}(m) \wedge \mathrm{attacker}(pk)\\ &\Rightarrow \mathrm{attacker}\big(\mathrm{pencrypt}(m.pk)\big).\\ &\mathrm{attacker}(sk)\\ &\Rightarrow \mathrm{attacker}(\mathrm{pk}(sk)).\\ &\mathrm{attacker}\Big(\mathrm{pencrypt}\big(m.\mathrm{pk}(sk)\big)\Big) \wedge \mathrm{attacker}(sk)\\ &\Rightarrow \mathrm{attacker}(m).\end{aligned} \qquad (۱)$$

منجر می‌شود که بندهای اول و دوم با سازنده‌های pencrypt و pk متناظر هستند و بند آخر با مخرب pdecrypt متناظر است. هنگامی‌که مهاجم پیام رمزگذاری‌شده $\mathrm{pencrypt}(m.pk)$و کلید رمزگشایی sk را دارد، آن‌گاه او هم‌چنین متن ساده m را دارد. (فرض می‌کنیم که رمزنگاری کامل است، بنابراین، مهاجم می‌تواند از پیام رمزشده به متن ساده دست یابد فقط اگر کلید را داشته باشد.)

بندهای مربوط به امضاءها (sign، getmess، checksign) و رمزگذاری کلید- مشترک (cencrypt، sdecrypt) در شکل (۲) ارائه شده است.

بندهای بالا توانایی‌های محاسباتی مهاجم را توصیف می‌کند. علاوه‌بر این، مهاجم در ابتدا کلیدهای عمومی شرکت‌کنندگان در

پروتکل را دارد. بنابراین، مهاجم بندهای (($\mathrm{attacker}(\mathrm{pk}(sk_A[\,]))$ و (($\mathrm{attacker}(\mathrm{pk}(sk_B[\,]))$را اضافه می‌کند. به‌ویژه، $a[\,]$کلید خصوصی هر شرکت‌کننده غیرامین را نمایش می‌دهد که کلید عمومی او $\mathrm{pk}(a[\,])$ است و مهاجم می‌تواند با استفاده از بندهای مربوط به سازنده pk آن را محاسبه کند.

۲-۳- نمایش پروتکل

حال، این‌که خود پروتکل چگونه نمایش داده می‌شود را توصیف می‌کنیم. فرض کنیم A و B قصد صحبت با یک شرکت‌کننده دلخواه A، Bیا هم‌چنین شرکت‌کنندگان بداندیش که با مهاجم نمایش داده می‌شوند، را دارند. بنابراین، اولین پیامی که توسط A ارسال می‌شود، می‌تواند

$$\mathrm{pencrypt}(\mathrm{sign}(k.sk_A[\,]).\mathrm{pk}(x))$$

برای هر x باشد. مهاجم مجاز است پروتکل را با هر شرکت‌کننده‌ای که بخواهد شروع کند، یعنی مهاجم یک پیام اولیه به A می‌فرستد که کلید عمومی شرکت‌کننده‌ای را مشخص می‌کند که A باید با او صحبت کند. این شرکت‌کننده می‌تواند B یا هر شرکت‌کننده دیگری باشد که با مهاجم نمایش داده می‌شود. بنابراین، اگر مهاجم یک کلید $\mathrm{pk}(x)$ را داشته باشد، می‌تواند پیام $\mathrm{pk}(x)$ را به A بفرستد؛ A با اولین پیام خود پاسخ می‌دهد که مهاجم می‌تواند آن‌را شنود کند، بنابراین، $\mathrm{pencrypt}(\mathrm{sign}(k.sk_A[\,]).\mathrm{pk}(x))$را به‌دست می‌آورد. در نتیجه، یک بند به صورت:

$$\mathrm{attacker}(\mathrm{pk}(x))$$
$$\Rightarrow \mathrm{attacker}\left(\mathrm{pencrypt}\left(\mathrm{sign}\left(k.sk_A[\,]\right).\mathrm{pk}(x)\right)\right)$$

خواهیم داشت. علاوه‌براین، در هر اجرای پروتکل یک کلید جدید تولید می‌شود. درنتیجه، اگر دو کلید متفاوت $\mathrm{pk}(x)$ توسط A دریافت شود، کلیدهای k تولید شده مطمئناً متفاوت هستند: k به $\mathrm{pk}(x)$ بستگی دارد. بند به

$$\mathrm{attacker}(\mathrm{pk}(x))$$
$$\Rightarrow \mathrm{attacker}\left(\mathrm{pencrypt}\left(\mathrm{sign}\left(k[\mathrm{pk}(x)].sk_A[\,]\right).\mathrm{pk}(x)\right)\right)$$

(۲)

تبدیل می‌شود.

هنگامی که B یک پیام دریافت می‌کند، آن را با کلید خصوصی خود sk_B رمزگشایی می‌کند، بنابراین B پیامی به صورت $\mathrm{pencrypt}(x'.\mathrm{pk}(sk_B[\,]))$ انتظار دارد. سپس، Bبررسی می‌کند که آیا A، x'را امضاء کرده است، یعنی، B، $\mathrm{checksign}(x'.pk_A)$را ارزیابی می‌کند و این تنها زمانی موفق است که$x' = \mathrm{sign}(y.sk_A[\,])$ اگر چنین باشد، او فرض می‌کند که کلید y را تنها A می‌داند، و یک راز s (یک ثابت که مهاجم آن را ندارد) که توسط y رمز شده است را می‌فرستد. فرض می‌کنیم مهاجم پیام فرستاده شده از طرف A را ذخیره می‌کند و پیام ارسال شده توسط B را شنود می‌کند. بنابراین بند حاصل به صورت

$$\mathrm{attacker}\left(\mathrm{pencrypt}\left(\mathrm{sign}(y.sk_A[\,]).\mathrm{pk}(sk_B[\,])\right)\right)$$
$$\Rightarrow \mathrm{attacker}\left(\mathrm{sencrypt}(s.y)\right)$$

است.

۲-۴- خلاصه

همان‌گونه که در شکل (۲) برای پروتکل جاری نشان داده شده است، یک پروتکل می‌تواند با سه مجموعه از بندهای هورن نمایش داده شود:

- بندهایی که توانایی‌های محاسباتی مهاجم را نمایش می‌دهند: سازنده‌ها، مخرب‌ها و تولید نام.
- حقیقت‌های متناظر با دانش اولیه مهاجم. در حالت کلی، حقیقت‌هایی وجود دارند که کلیدهای عمومی شرکت‌کنندگان و نام آن‌ها را به مهاجم می‌دهند.
- بندهایی که پیام‌های پروتکل را نمایش می‌دهند. برای هر پیام پروتکل یک مجموعه از بندها وجود دارد. در مجموعه‌ی متناظر با پیامi ام، ارسال شده توسط شرکت کنندهX، بندها به‌صورت

$$\mathrm{attacker}(M_{j_1}) \wedge \ldots \wedge \mathrm{attacker}(M_{j_n})$$
$$\Rightarrow \mathrm{attacker}(M_i)$$

است که M_{j_1}، ...، M_{j_n} الگوهای پیام‌های دریافت شده توسط X قبل از ارسالi امین پیام است و M_i الگوی پیام i است.

توانایی‌های محاسباتی مهاجم:

برای هر سازنده f با شناسه‌ی n:

$$\mathrm{attacker}(x_1) \wedge \ldots \wedge \mathrm{attacker}(x_n)$$
$$\Rightarrow \mathrm{attacker}\left(f(x_1.\ldots.x_n)\right)$$

برای هر مخرب g، برای هر قاعده بازنویسی $g(M_1.\ldots.M_n) \rightarrow M$ در$\mathrm{def}(g)$:

$$\mathrm{attacker}(M_1) \wedge \ldots \wedge \mathrm{attacker}(M_n) \Rightarrow \mathrm{attacker}(M)$$

یعنی

pencrypt	$\text{attacker}(m) \wedge \text{attacker}(pk)$ $\Rightarrow \text{attacker}(\text{pencrypt}(m.pk))$
pk	$\text{attacker}(sk) \Rightarrow \text{attacker}(\text{pk}(sk))$
pdecrypt	$\text{attacker}\left(\text{pencrypt}(m.\text{pk}(sk))\right) \wedge \text{attacker}(sk)$ $\Rightarrow \text{attacker}(m)$
sign	$\text{attacker}(m) \wedge \text{attacker}(sk)$ $\Rightarrow \text{attacker}(\text{sign}(m.sk))$
getmess	$\text{attacker}(\text{sign}(m.sk)) \Rightarrow \text{attacker}(m)$
checksign	$\text{attacker}(\text{sign}(m.sk)) \wedge \text{attacker}(sk)$ $\Rightarrow \text{attacker}(m)$
sencrypt	$\text{attacker}(m) \wedge \text{attacker}(k)$ $\Rightarrow \text{attacker}(\text{sencrypt}(m.k))$
sdecrypt	$\text{attacker}(\text{sencrypt}(m.k)) \wedge \text{attacker}(k)$ $\Rightarrow \text{attacker}(m)$

تولید نام:

$\text{attacker}(a[\,])$

دانش اولیه:

$\text{attacker}(\text{pk}(sk_A[\,])).\text{attacker}(\text{pk}(sk_B[\,]))$

پروتکل:

پیام اول:

$\text{attacker}(\text{pk}(x))$

$\Rightarrow \text{attacker}\left(\text{pencrypt}(\text{sign}(k[\text{pk}(x)].\, sk_A[\,]).\text{pk}(x))\right)$

پیام دوم:

$\text{attacker}\left(\text{pencrypt}(\text{sign}(y.\, sk_A[\,]).\text{pk}(sk_B[\,]))\right)$

$\Rightarrow \text{attacker}(\text{sencrypt}(s.y))$

شکل (۲). خلاصه نمایش پروتکل مثال جاری

۲-۵- محک محرمانگی

هدف ما مشخص کردن خواص محرمانگی است: برای مثال، آیا مهاجم می‌تواند راز s را به دست آورد؟ یعنی، آیا حقیقت $\text{attacker}(s)$از بندها قابل استنتاج است؟ اگر $\text{attacker}(s)$ قابل استنتاج باشد، دنباله‌ی بندهایی که برای استنتاج $\text{attacker}(s)$به‌کار می‌روند به توصیف یک حمله منجر می‌شوند. عبارت M محرمانه است اگر مهاجم نتواند با شنیدن و ارسال پیام‌ها و انجام محاسبات به آن دست یابد.

در مثال جاری، $\text{attacker}(s)$از بندها قابل استنتاج است. استنتاج به صورت زیر است. مهاجم نام جدید $a[\,]$ (به عنوان یک کلید خصوصی) را تولید می‌کند، سپس $\text{pk}(a[\,])$ را با استفاده از بند مربوط به pk محاسبه می‌کند و با استفاده از بند مربوط به پیام اول

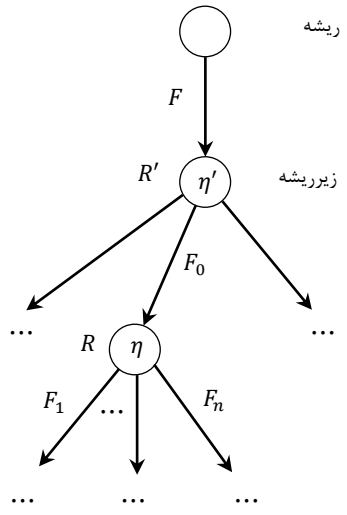
$$\text{pencrypt}(\text{sign}(k[\text{pk}(a[\,])].\, sk_A[\,]).\text{pk}(a[\,]))$$

را به دست می‌آورد. وی این پیام را با استفاده از بند مربوط به pdecrypt و آگاهی‌اش از $a[\,]$ رمزگشایی می‌کند و $\text{sign}(k[\text{pk}(a[\,])].\, sk_A[\,])$ را به دست می‌آورد. سپس امضاء را با استفاده از بند pencrypt و دانش اولیه‌اش از $\text{pk}(sk_B[\,])$ مجدداً رمزگذاری نموده و $\text{pencrypt}(\text{sign}(k[\text{pk}(a[\,])].\, sk_A[\,]).\text{pk}(a[\,]))$ را به‌دست می‌آورد. با استفاده از بند مربوط به پیام دوم، $\text{sencrypt}(s.k[\text{pk}(a[\,])])$ را به‌دست می‌آورد. از طرف دیگر، از $\text{sign}(k[\text{pk}(a[\,])].\, sk_A[\,])$ با استفاده از بند مربوط به getmess، $k[\text{pk}(a[\,])]$را به‌دست می‌آورد. بنابراین می‌تواند با استفاده از بند مربوط به sdecrypt، $\text{sencrypt}(s.k[\text{pk}(a[\,])])$را رمزگشایی و در نتیجه s را به‌دست آورد. به عبارت دیگر، مهاجم یک نشست بین A و یک شرکت‌کننده غیر امین با کلید خصوصی $a[\,]$ آغاز می‌کند. او اولین پیام $\text{pencrypt}(\text{sign}(k.\, sk_A[\,]).\text{pk}(a[\,]))$ را می‌گیرد، آن را رمزگشایی نموده و مجدداً با $\text{pk}(sk_B[\,])$ رمز نموده و آن را به B می‌فرستد. برایB، این پیام به صورت اولین پیام از یک نشست بین A و B به نظر می‌رسد، بنابراین B با $\text{sencrypt}(s.k)$ پاسخ می‌دهد، که مهاجم می‌تواند آن را رمزگشایی کند، زیرا او k را از اولین پیام به دست آورده است.

در نتیجه، استنتاج حاصل با حمله شناخته شده علیه این پروتکل متناظر است. در مقابل، اگر پروتکل را با افزودن کلید عمومی B در پیام اول $\left\{\left|\left[\langle \text{pk}_B.\, k\rangle\right]_{sk_A}\right|\right\}^{a}_{\text{pk}_B}$ اصلاح کنیم، $\text{attacker}(s)$ از بندها قابل استنتاج نیست. بنابراین، پروتکل اصلاح شده محرمانگی s را حفظ می‌کند.

در ادامه، این‌که چه هنگام یک حقیقت داده شده از یک مجموعه از بندها استنتاج می‌شود را به طور صوری تعریف می‌کنیم. فرض‌های یک بند به عنوان مجموعه‌های چندگانه درنظر گرفته می‌شوند. به این معنی که ترتیب فرض‌ها مهم نیست ولی تعداد تکرارهای یک فرض مهم است. از R برای بندها (قاعده‌های برنامه‌نویسی منطق)، H برای فرضیات و C برای حکم استفاده می‌کنیم.

تعریف ۱ (شمول) گوییم $H_1 \Rightarrow C_1$ شامل $H_2 \Rightarrow C_2$ است و می‌نویسیم$(H_1 \Rightarrow C_1) \supseteq (H_2 \Rightarrow C_2)$ ، اگر و تنها اگر جانشینی σ وجود داشته باشد به طوری که $\sigma C_1 = C_2$ و $\sigma H_1 \subseteq H_2$ (شمول مجموعه‌های چندگانه).



شکل (۳): استنتاج F

استنتاج به‌صورت زیر تعریف می‌شود، شکل (۳) را ببینید.

تعریف ۲ (استنتاج‌پذیری) فرض کنید F یک حقیقت بسته، یعنی، یک حقیقت بدون متغیر باشد. فرض کنید $\mathcal{R}$ یک مجموعه از بندها باشد. F از $\mathcal{R}$ استنتاج‌پذیر است اگر و تنها اگر یک استنتاج برای F از $\mathcal{R}$ وجود داشته باشد، یعنی، یک درخت متناهی که به صورت زیر تعریف می‌شود:

۱. رأس‌های (به‌جز ریشه) درخت با بندهای $R \in \mathcal{R}$ برچسب‌گذاری می‌شوند؛
۲. یال‌های درخت با حقیقت‌های بسته برچسب‌گذاری می‌شوند؛
۳. اگر درخت دارای رأسی باشد که با R برچسب‌گذاری شده است و یک یال وارد شونده به آن با F_0 و n یال خارج شونده از آن با F_1، ...، F_n نشانه‌گذاری شده‌اند، آن‌گاه $R \supseteq F_1 \wedge \cdots \wedge F_n \Rightarrow F_0$.
۴. ریشه یک یال خارج شونده دارد که با F نشانه‌گذاری شده است. تنها پسر ریشه، زیرریشه نامیده می‌شود.

۳- الگوریتم تجزیه

نمایش پروتکل، مجموعه‌ای از بندهای هورن و هدف تعیین این است که آیا یک حقیقت داده شده می‌تواند از این بندها استنتاج شود یا نه. این دقیقاً مسأله‌ای است که با سیستم‌های پرولوگ[1] معمول حل می‌شود. اما، در این‌جا، نمی‌توانیم از این سیستم‌ها استفاده کنیم، زیرا ممکن است متوقف نشوند. برای مثال، بند

$$\text{attacker}\left(\text{pencrypt}\left(m.\text{pk}(sk)\right)\right) \wedge \text{attacker}(sk) \Rightarrow \text{attacker}(m)$$

منجر به درنظر گرفتن بندهای پیچیده و پیچیده‌تر با تعداد نامحدودی رمزگذاری می‌شود. البته می‌توانیم برای حل مشکل، عمق عبارت‌ها را به دلخواه محدود کنیم، اما می‌توان بهتر از آن را انجام داد.

۳-۱. الگوریتم پایه

ابتدا تجزیه را تعریف می‌کنیم: هنگامی که حکم یک بند R با فرض بند دیگر (یا همان بند) R' متحد است، تجزیه، بند جدیدی استنتاج می‌کند که متناظر با کاربرد R و R' یکی پس از دیگری است. به‌طور صوری، تجزیه به صورت زیر تعریف می‌شود:

تعریف ۳ فرض کنید R و R' دو بند باشند، $R = H \Rightarrow C$ و $R' = H' \Rightarrow C'$. فرض کنید $F_0 \in H'$ موجود باشد به طوری‌که C و F_0 متحد باشند و σ کلی‌ترین متحدکننده[2] C و F_0 باشد. در این حالت، تعریف می‌کنیم

$$R \circ_{F_0} R' = \sigma\big(H \cup (H' \setminus \{F_0\})\big) \Rightarrow \sigma C'$$

برای مثال، اگر R بند (۲) و R' بند (۱) و حقیقت F_0 به‌صورت $F_0 = \text{attacker}\left(\text{pencrypt}\left(m.\text{pk}(sk)\right)\right)$ باشد، آن‌گاه $R \circ_{F_0} R'$،

$$\text{attacker}(\text{pk}(x)) \wedge \text{attacker}(x) \Rightarrow \text{attacker}\big(\text{sign}(k[\text{pk}(x)].sk_A[\,])\big)$$

است که در آن جانشینی

$$\sigma = \{sk \mapsto x.m \mapsto \text{sign}(k[\text{pk}(x)].sk_A[\,])\}$$

است. الگوریتم تجزیه در دو فاز کار می‌کند که در شکل (۴) توصیف شده است. فاز اول مجموعه‌ی اولیه از بندها را به یک بند جدید تبدیل می‌کند که حقیقت‌های یکسانی را استنتاج می‌کند. فاز دوم از یک جستجوی عمق-اول برای تشخیص این‌که یک حقیقت می‌تواند از بندها استنتاج شود یا نه استفاده می‌کند.

فاز اول: اشباع

$\text{saturate}(\mathcal{R}_0)=$

۱. $\mathcal{R} \leftarrow \emptyset$.

برای هر $R \in \mathcal{R}_0$، $\mathcal{R} \leftarrow \text{elim}(\{R\} \cup \mathcal{R})$.

۲. اعمال زیر را تا رسیدن به یک نقطه ثابت تکرار کن

برای هر $R \in \mathcal{R}$ به طوری که $sel(R) = \emptyset$،

برای هر $R' \in \mathcal{R}$، برای هر $F_0 \in sel(R')$ به طوری که $R \circ_{F_0} R'$ تعریف شده باشد،

1- Prolog

2- most general unifier

$\mathcal{R} \leftarrow \text{elim}(\{R \circ_{F_0} R'\} \cup \mathcal{R})$

۳. $\{R \in \mathcal{R} \mid sel(R) = \emptyset\}$ را برگردان.

فاز دوم: جستجوی عقب‌گرد عمق-اول

$$\text{deriv}\left(R, \mathcal{R}, \mathcal{R}_1\right) = \begin{cases} \emptyset & \text{اگر } R' \in \mathcal{R} \text{ وجود داشته باشد به طوری که } R' \supseteq R \\ \{R\} & \text{درغیر این صورت اگر } sel(R) = \emptyset \\ \cup\{\text{deriv}\left(R' \circ_{F_0} R, \{R\} \cup \mathcal{R}, \mathcal{R}_1\right) & \text{در غیر این‌صورت} \\ \mid R' \in \boldsymbol{R}, F_0 \in sel(R), & \{\text{به طوری که } R' \circ_{F_0} R \text{ تعریف شده باشد} \end{cases}$$

$$\text{derivable}\left(F, \mathcal{R}_1\right) = \text{deriv}(F \Rightarrow F, \emptyset, \mathcal{R}_1)$$

شکل (۴). الگوریتم تجزیه

۴- نتیجه آزمایش‌ها

این روش برای اثبات خواص محرمانگی و احراز هویت در بسیاری از پروتکل‌ها به‌کار رفته است، از جمله نسخه معیوب و صحیح پروتکل کلید عمومی نیدهام-شرودر[1] [۱۳، ۱۴]، پروتکل کلید مشترک نیدهام-شرودر [۱۳، ۴، ۱۵]، پروتکل کلید عمومی وو-لم[2] [۱۶، ۱۷]، پروتکل کلید مشترک وو-لم [۱۶، ۱۸، ۱۹، ۱۷، ۲۰]، پروتکل دنینگ-ساسکو[3] [۲۱، ۱۹]، پروتکل یالوم[4] [۴]، پروتکل اوتوی-ریس[5] [۲۲، ۴، ۳]، و پروتکل اسکیم[6] [۲۳]. هیچ حمله نادرستی در این آزمون‌ها رخ نداده است و تنها حالت عدم توقف در برخی نسخه‌های معیوب پروتکل کلید مشترک وو-لم بوده است. این پروتکل‌ها در کمتر از یک ثانیه روی یک کامپیوتر Intel Core i3-2310M 2.10GHz درستی‌یابی شده‌اند.

۵- نتیجه‌گیری

یک جنبه مهم رهیافت بندهای هورن توانایی اثبات خواص امنیتی پروتکل‌ها برای تعداد نامحدودی از نشست‌ها به روشی کاملاً خودکار است. این برای تصدیق پروتکل‌ها اساسی است. این رهیافت هم‌چنین رده‌ی وسیعی از اولیه‌های رمزنگاری را پشتیبانی می‌کند و می‌تواند رده‌ی وسیعی از خواص امنیتی را ثابت کند.

از طرف دیگر، مسأله درستی‌یابی پروتکل‌ها برای تعداد نامحدود از نشست‌ها تصمیم‌ناپذیر است، بنابراین این رهیافت ناتمام است یعنی همواره متوقف نمی‌شود و تقریب را به‌کار می‌گیرد، بنابراین پروتکل‌های امنی وجود دارند که امن بودن آن‌ها را نمی‌تواند اثبات کند، اگرچه در عمل بسیار دقیق و کارآ است.

1- Needham-Schroeder
2- Woo-Lam
3- Denning-Sacco
4- Yahalom
5- Otway-Rees
6- Skeme

۶- مراجع

[1] B. Blanchet, "An efficient cryptographic protocol verifier based on Prolog rules," In 14th IEEE Computer Security Foundations Workshop (CSFW-14), IEEE Computer Society, pp. 82–96, 2001.

[2] N. Durgin, P. Lincoln, J. C. Mitchell, and A. Scedrov, "Multiset rewriting and the complexity of bounded security protocols," Journal of Computer Security, vol. 12, no. 2, pp. 247–311, 2004.

[3] L. C. Paulson, "The inductive approach to verifying cryptographic protocols," Journal of Computer Security, vol. 6 (1–2), pp. 85–128, 1998.

[4] M. Burrows, M. Abadi, and R. Needham, "A Logic of Authentication," Proceedings of the Royal Society of London A, vol. 426, pp. 233–271, 1989.

[5] M. Abadi, "Secrecy by Typing in Security Protocols," Journal of the ACM, vol. 46, no. 5, pp. 749–786, 1999.

[6] L. Cardelli, G. Ghelli, and A. D. Gordon, "Secrecy and Group Creation," In C. Palamidessi, editor, CONCUR 2000: Concurrency Theory, volume 1877 of Lecture Notes on Computer Science, pp. 365–379. Springer Verlag, 2000.

[7] M. Hennessy and J. Riely, "Information Flow vs. Resource Access in the Asynchronous Pi-Calculus," In Proceedings of the 27th International Colloquium on Automata, Languages and Programming, Lecture Notes on Computer Science, pp. 415–427, Springer Verlag, 2000.

[8] F. J. Thayer Fabrega, J. C. Herzog, and J. D. Guttman, "Strand Spaces: Proving Security Protocols Correct," Journal of Computer Security, vol. 7, pp. 191-230, 1999.

[9] J. Heather and S. Schneider, "Towards automatic verification of authentication protocols on an unbounded network," In 13th IEEE Computer Security Foundations Workshop (CSFW-13), pp. 132–143, Cambridge, England, July 2000.

[10] P. Cousot and R. Cousot, "Systematic design of program analysis frameworks," In 6th Annual ACM Symposium on Principles of Programming Languages, pp. 269–282, 1979.

[11] C. Weidenbach, "Towards an automatic analysis of security protocols in first-order logic," In 16th International Conference on Automated Deduction (CADE-16), vol. 1632 of Lecture Notes in Artificial Intelligence, pp. 314–328, 1999.

[12] D. Dolev and A. C. Yao, "On the security of public key protocols," IEEE Transactions on Information Theory, IT, vol. 29, no. 12, pp. 198–208, 1983.

[13] R. M. Needham and M. D. Schroeder, "Using encryption for authentication in large networks of computers," Commun. ACM, vol. 21, no. 12, pp. 993–999, 1978.

[14] G. Lowe, "Breaking and fixing the Needham-Schroeder public-key protocol using FDR," In Tools and Algorithms for the Construction and Analysis of Systems, vol. 1055 of LNCS, pp. 147–166. Springer, 1996.

[15] R. M. Needham and M. D. Schroeder, "Authentication revisited," Operating Systems Review, vol. 21, no. 1, pp. 7, 1987.

[16] T. Y. C. Woo and S. S. Lam, "Authentication for distributed systems," Computer, vol. 25, no. 1, pp. 39–52, 1992.

[17] T. Y. C. Woo and S. S. Lam, "Authentication for distributed systems," In Internet Besieged: Countering Cyberspace Scofflaws, pp. 319–355, ACM Press and Addison-Wesley, 1997.

[18] R. Anderson and R. Needham, "Programming Satan's computer," In Computer Science Today: Recent Trends and Developments, vol. 1000 of LNCS, pp. 426–440, Springer, 1995.

[19] M. Abadi and R. Needham, "Prudent engineering practice for cryptographic protocols," IEEE Transactions on Software Engineering, vol. 22, no. 1, pp. 6–15, 1996.

[20] A. Gordon and A. Jeffrey, "Authenticity by typing for security protocols," Journal of Computer Security, vol. 11, no. 4, pp. 451–521, 2003.

[21] D. E. Denning, and G. M. Sacco, "Timestamps in key distribution protocols," Commun. ACM, vol. 24, no. 8, pp. 533–536, 1981.

[22] D. Otway and O. Rees, "Efficient and timely mutual authentication," Operating Systems Review, vol. 21, no. 1, pp. 8–10, 1987.

[23] H. Krawczyk, "SKEME: A versatile secure key exchange mechanism for Internet," In Internet Society Symposium on Network and Distributed Systems Security, 1996.