مدلی برای توصیف پروتکل‌های رمزنگاری و تفسیر اجرایی مدل با رویکرد جبری

سعید جلیلی[1*]، بهروز ترک لادانی[2]

۱- استادیار مهندسی کامپیوتر، بخش مهندسی برق، دانشکدهٔ فنی و مهندسی، دانشگاه تربیت مدرس

۲- دانشجوی دکترای کامپیوتر، بخش مهندسی برق، دانشکدهٔ فنی و مهندسی، دانشگاه تربیت مدرس

*تهران، صندوق پستی: ۱۴۳-۱۴۱۱۵

sjalili@modares.ac.ir , Torkla_b@modares.ac.ir

(دریافت مقاله: اسفند ۱۳۸۰، پذیرش مقاله: شهریور ۱۳۸۲)

چکیده ـ اگرچه روشهای تحلیل منطقی و روشهای ساخت حمله، دو رویکرد کلی در زمینهٔ تحلیل پروتکل‌های رمزنگاری است، اما اغلب برای تحلیل پروتکل‌ها، به کارگیری توأم آنها مناسبتر است. در این مقاله، یک چارچوب عمومی برای یکپارچه سازی تحلیل امنیت پروتکل‌های رمزنگاری ارائه شده است. بر اساس این چارچوب می‌توان روشهایی تحلیلی ارائه کرد که در آنها وارسی وجود ویژگیهای امنیتی در پروتکل، همراه با وارسی وجود سناریوهای نقض ویژگیهای امنیتی، به صورت یکجا انجام می‌شود. در این مقاله همچنین یک مدل محاسباتی برای پروتکل‌های رمزنگاری، به عنوان جزئی از چارچوب عمومی مذکور همراه با یک تفسیر اجرایی از آن، بر اساس سیستم‌های بازنویسی جبری ارائه شده است. با توصیف پروتکل در قالب مجموعه‌ای از گام‌های منفرد و به کمک یک الگوریتم تبدیل، مجموعه‌ای پایان‌پذیر و همگرا از قوانین بازنویسی مولد مسیر اجرای پروتکل ایجاد می‌شود. توصیف یک پروتکل رمزنگاری در چارچوب ارائه شده نیز به عنوان نمونه بیان شده است.

کلید واژگان: پروتکل‌های رمزنگاری، وارسی صوری پروتکل‌های رمزنگاری، مدل محاسباتی پروتکل، تحلیل منطقی.

۱- مقدمه

پروتکل‌های رمزنگاری اگر چه ساختاری ساده دارند، اما خواص امنیتی که از خود ارائه می‌کنند چندان واضح نیستند. به کارگیری روشهای مختلف رمزنگاری برای حصول اهداف مختلف در پروتکل و آثاری که این روشها بر یکدیگر می‌گذارند، به ایجاد یک سری خواص جبری در پروتکل‌ها منجر می‌شود که یا مانع رسیدن پروتکل به تمامی اهداف مورد نظر و ضروری در آن می‌شود یا راه را برای سوء استفادهٔ عوامل مخرب و نفوذی هموار می‌کند. تعدّد پروتکل‌هایی که خطادار بودن آنها نشان داده شده، شاهدی بر این مدعا است[۱].

رابطهٔ دو طرفه‌ای بین دو عامل وجود رخنه در پروتکل و بروز حملات علیه آنها وجود دارد. وجود رخنه در پروتکل مولد سناریوهای حمله است و حمله به پروتکل، مبین وجود رخنه‌ای کشف نشده در طراحی پروتکل است. بر این اساس تلاش‌های انجام شده برای تحلیل و وارسی پروتکل‌های رمز نگاری نیز در این دو جهت انجام شده است. بعضی از روشهای تحلیل بر صوری‌سازی مفاهیم، تکنیک‌های به کار رفته و اهداف مد نظر از طراحی پروتکل، به منظور وارسی صوری عملکرد پروتکل تأکید دارند . هدف از این روشها کشف خطا و نقص در وصول به اهداف مد نظر پروتکل از طریق ایجاد اثبات‌های صوری در مورد این اهداف است. از اینگونه روشها با عنوان روشهای

تحلیل منطقی[1] [۲]، یا روشهای ساخت استنتاج[2] [۳] یا روشهای مبتنی بر مفهوم باور و دانش[3] [٤] یاد می‌شود.

از طرف دیگر بعضی از روشها مستقیماً به سمت کشف سناریوهای حمله علیه پروتکل‌ها از طریق مدلسازی پروتکل همراه با عامل نفوذی و ایجاد فضای حملات امکان پذیر پیش رفته اند. در این روشها با تعریف قابلیت‌ها و امکانات نفوذی سعی می‌شود با بررسی تمامی‌حالات ممکن، به حالتی دست پیدا کنیم که در آن عامل نفوذی بتواند اطلاعات مخفی را کشف کند یا به نحوی عوامل پروتکل را فریب دهد، مثلاً خود را جانشین یکی از طرفین مجاز جلوه دهد. از اینگونه روشها با نام روشهای جبری[٤] ، یا روشهای مبتنی بر ماشین حالت[4] [۳] ، یا روشهای ساخت حمله[5] [٥] یاد می‌شود. منطق BAN [٦] و دنباله روهای آن (GNY [۷]، SVO[۸] و ...) پیرو رویکرد تحلیل منطقی هستند در حالی که Dolev-Yao [۹] و دنباله‌روهای آن (NRL [۱۰]، Interrogator [۱۱] و ...) رویکرد ساخت حمله را انتخاب کرده اند.

می‌توان دو روش ذکر شده را به ترتیب روشهای تحلیل ایستا و پویای پروتکل نامید. با انجام تحلیل ایستا، صحت طراحی پروتکل در حصول ویژگی‌های امنیتی مورد انتظار به اثبات می‌رسد و با انجام تحلیل پویا می‌توان میزان مقاومت پروتکل در محیط متخاصم را ارزیابی نمود. تحلیل ایستا را می‌توان یک فعالیت مربوط به زمان طراحی و تحلیل پویا را فعالیتی برای ارزیابی پروتکل طراحی شده دانست. برای دستیابی به پروتکل‌هایی با امنیت بالا، هر دو تحلیل ضروری است. علاوه بر این، پروتکل‌های رمزنگاری اهداف متفاوتی داشتند و از مکانیزم‌های مختلفی استفاده می‌کنند. روشهای تحلیل ارائه شده، معمولاً توانایی تحلیل دسته‌های خاصی از پروتکل‌ها را دارند و روشهایی نیز که سعی در ارائهٔ ویژگی‌های گوناگون پروتکل‌ها به صورت یکجا داشته‌اند در عمل بسیار پیچیده و حجیم شده‌اند.

به جای استفاده از روشهای خاص منظوره برای انجام انواع تحلیل روی پروتکل‌ها، با اتکا بر یک چارچوب کلی تحلیل، می‌توان روش تحلیل مورد نیاز و مناسب برای هر دسته از پروتکل‌ها را ارائه کرد. به کمک چنین چارچوبی، می‌توان هم پروتکل را مدل کرد و در مورد ویژگی‌های امنیتی که پدید می‌آورد استنتاج کرد، و هم می‌توان عامل نفوذی را وارد کرد و توانایی وی را در ایجاد وضعیت‌هایی که به نقض ویژگیهای امنیتی منجر می‌شود وارسی نمود. به عبارت دیگر، تحلیل ایستا و پویای پروتکل بر اساس این چارچوب امکان پذیر خواهد بود. علاوه بر این، با استفاده از این چارچوب می‌توان برای دسته‌های مختلفی از پروتکل‌ها با اهداف متفاوت و سازوکارهای مختلف مورد استفاده، روشهای ویژهٔ تحلیل منطبق و مناسب با نیاز آنها فراهم ساخت. روشهایی که بر این اساس ارائه می‌شوند چون از جزئیات و پیچیدگی‌های سایر پروتکل‌ها فارغ بوده و بر ویژگی‌های همان دسته از پروتکل‌ها تأکید دارند، قابل فهم‌تر و کاراتر از روشهای کلی خواهند بود.

ما در این مقاله به سمت چنین هدفی پیش رفته‌ایم. درادامه، ابتدا ضمن مقایسهٔ دو روش تحلیل ایستا و پویا، ملزومات و نیازمندی‌های یک چارچوب عام تحلیل پروتکل‌های رمزنگاری را بررسی و اجزای اساسی آن را مشخص می‌کنیم. سپس یک زبان توصیف و یک مدل محاسباتی از پروتکل به عنوان عناصر ثابت این چارچوب ارائه می‌شود. در ادامه، یک تفسیر اجرایی از نحوهٔ پیاده‌سازی مدل محاسباتی پروتکل بر اساس یک سیستم بازنویسی ارائه شده است. پس از آن، برای تبیین چارچوب ارائه شده و مدل محاسباتی آن، یک پروتکل نمونه به کمک آن توصیف شده و در نهایت جمع‌بندی مطالب ارائه شده است. لازم است ذکر شود که فکر اولیهٔ این کار و نسخهٔ اولیه‌ای از مطالب این مقاله در [۱۲] ارائه شده است.

1. Logic Analysis Methods
2. Inference Construction Methods
3. Knowledge and Belief Methods
4. State Machine
5. Attack Construction Methods

۲- تحلیل ایستا در مقابل تحلیل پویا

هدف از تحلیل ایستا، وارسی ایجاد اهداف امنیتی مورد نظر از پروتکل است. برای این کار، منطق حاکم بر اجزای پروتکل (توابع رمزنگاری و روشهای طراحی پروتکل) و چگونگی حصول دانش، باور و اعتماد در اثر اجرای پروتکل توسط عوامل، در قالب مجموعه‌ای از اصول منطقی ارائه می‌شود. پروتکل و فرضیات اولیهٔ آن نیز در قالب ایده‌آل شده[1] برای منطق، ورودی روش تحلیل هستند. ایده‌آل سازی این فرضیات و پروتکل از طریق یک گام نحوی[2] انجام می‌شود. سپس با انجام یک گام معنایی[3]، دانش، باور یا اعتماد قابل حصول از روی پروتکل از طریق انجام استنتاج به‌دست می‌آید. برای اثبات وجود هر ویژگی امنیتی در پروتکل، از استنتاج رو به جلو یا روبه‌عقب می‌توان استفاده کرد. شکل ۱ مدل کلی انجام تحلیل ایستا را نشان می‌دهد.

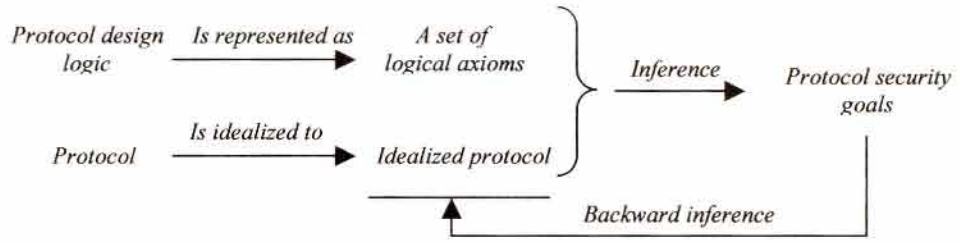
در رویکرد تحلیل پویا (شکل ۲)، تأکید اصلی بر نقشهای عوامل پروتکل به عنوان عوامل مجاز و نفوذی به عنوان عامل غیر مجاز است. عوامل پروتکل مجموعه‌ای از دانش‌ها و قابلیت‌ها را دارند (که محدود به پروتکل مورد تحلیل است) و نفوذی نیز دارای مجموعه‌ای اولیه از دانش‌ها وقابلیت‌ها است. نفوذی مجموعهٔ دانش و قابلیت خود را به هر طریقی که مناسب باشد به کار می‌برد اما عوامل مجاز مجبورند از دانش و قابلیت خود در راستای اجرای پروتکل استفاده کنند. نقض ویژگی امنیتی به این معنا است که نفوذی به کمک دانش و قابلیت‌های خود بتواند یک مثال نقض برای آن ویژگی ایجاد کند. یافتن سناریوهای نقض ویژگی‌های امنیتی بر اساس نوع روش می‌تواند به صورت رو به جلو یا رو به عقب و با کمک استنتاج، بررسی مدل یا استقرا صورت گیرد. در این رویکرد، به تغییرات در مجموعه دانش یا باور و اعتماد عوامل مجاز توجه نمی‌شود و فقط به دنبال پیدا کردن یک دانش خاص درمجموعه دانش نفوذی یا ایجاد یک وضعیت غیر امن در اجرای پروتکل توسط عوامل هستیم[۱۳].

۳- چارچوبی برای تحلیل پروتکل

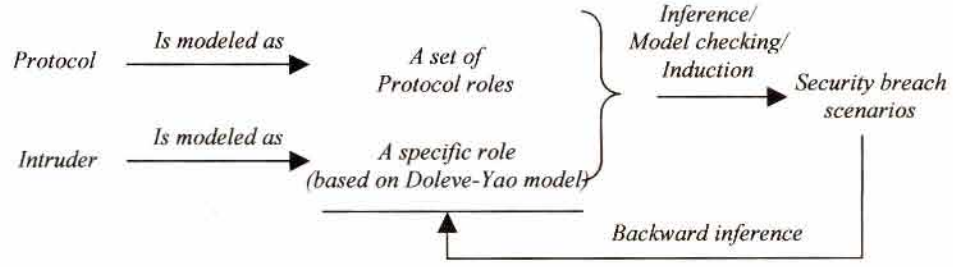
هدف ما از ارائهٔ چارچوب تحلیل پروتکل، فراهم سازی امکان انجام تحلیل‌های پویا و ایستا به صورت یکپارچه است. بر این اساس، در این مدل باید به صورت همزمان، توصیف پروتکل مورد تحلیل، منطق طراحی پروتکل و قابلیت‌های نفوذی را در نظرگرفت. برای این منظور به چهار مؤلفهٔ زیر نیازمندیم.

- **زبانی برای بیان توصیف پروتکل و ویژگی‌های امنیتی**
  با استفاده از این زبان می‌توان پیام‌های پروتکل را همراه با دانش ایجاد شده در طی اجرای پروتکل توصیف کرد. این دانش می‌تواند شامل دانش محلی مربوط به عوامل پروتکل، دانش سراسری مربوط به کل پروتکل یا دانش مربوط به نفوذی باشد. از زبان بیان ویژگی‌ها برای توصیف اهداف پروتکل نیز استفاده می‌شود. هدف، در واقع دانشی است که باید در نتیجهٔ اجرای پروتکل به‌دست آید یا توصیف ویژگی خاصی است که باید بر اثر اجرای پروتکل، نحوهٔ تغییر آن وارسی گردد.
- **مدلی محاسباتی برای پروتکل و نحوهٔ اجرای آن**
  مدل محاسباتی پروتکل اهمیت زیادی در چارچوب مورد بحث دارد. این مدل محاسباتی باید هم قابلیت مدلسازی عوامل مجاز پروتکل و نحوهٔ اعمال تغییرات در متغیرهای وضعیت محلی آنها و هم امکان وارد شدن نفوذی در اجرای پروتکل، به منظور اعمال قابلیت‌هایش را فراهم کند. توصیف پروتکل در مدل محاسباتی به

1. Idealized
2. Syntactical
3. Semantical



شکل ۱ تحلیل ایستا



شکل ۲ تحلیل پویا

کمک زبان توصیف ویژگی‌ها انجام می‌شود.

- **مجموعه‌ای از اصول[1] برای نشان دادن چگونگی اعمال تغییرات در وضعیت محلی عوامل پروتکل**

برای تحلیل ایستای پروتکل نیاز است که چگونگی اعمال تغییرات در وضعیت‌های محلی عوامل پروتکل بر اثر دریافت پیام‌های پروتکل، انجام محاسبات و یا استنتاج توسط این عوامل تعیین شود. برای این کار، از مجموعه‌ای از اصول استفاده می‌کنیم. این مجموعه نشان‌دهندهٔ چگونگی دستیابی عوامل پروتکل به باورها ودانش‌های مورد نیاز در طی اجرا است. برای بیان این مجموعه اصول از زبان توصیف ویژگی‌ها استفاده می‌شود.

- **مجموعه‌ای از اصول برای توصیف قابلیت‌های نفوذی**

در تحلیل پویای پروتکل، نفوذی دارای قابلیتهایی مانند ترکیب و تفکیک پیام‌ها، رمزنگاری و رمزگشایی با کلیدهای در دست خود ، انجام برخی محاسبات و استنتاج‌ها است. برای نشان دادن این قابلیت‌ها نیز از یک مجموعه اصول مجزا استفاده می‌کنیم. بیان این اصول نیز با زبان توصیف ویژگی‌ها صورت می‌گیرد.

به این ترتیب می‌توان چارچوب تحلیل پروتکل را به صورت $Analyze(\mathcal{L},\mathcal{M},\mathcal{PIR},\mathcal{EIR})$ نشان داد. $\mathcal{L}$ زبانی است که از طریق آن پیام‌های پروتکل و متغیرهای وضعیت اجرای پروتکل توصیف می‌شود. $\mathcal{M}$ یک مدل محاسباتی از پروتکل ونحوهٔ اجرای آن است. $\mathcal{PIR}$[2] و $\mathcal{EIR}$[3] نیز به‌ترتیب مجموعه اصول برای انجام استنتاج‌های عوامل مجاز و نفوذی است. توصیف پروتکل در مدل محاسباتی $\mathcal{M}$ و اصول لازم در مجموعه‌های $\mathcal{PIR}$ و $\mathcal{EIR}$ به زبان $\mathcal{L}$ بیان می‌شود.

مدل محاسباتی پروتکل ($\mathcal{M}$) در چارچوب مورد بحث حائز اهمیت بسیاری است. در بخش بعد به معرفی مدلی محاسباتی از پروتکل پرداخته‌ایم. در این مدل محاسباتی، پروتکل به عنوان مجموعه‌ای از نقش‌ها در نظر گرفته می‌شود

1. Axioms

2. Principal Inference Rules

3. Eavesdropper Inference Rules

و نفوذی نیز دارای نقش ویژه‌ای است. ترکیب خاصی از نقش‌های پروتکل از طریق انجام یک گام نحوی، یک مسیر اجرای پروتکل را می‌سازد و استنتاج روی این مسیر اجرا (به عنوان یک گام معنایی) موجب گسترش دانش یا باور عوامل مجاز پروتکل می‌شود و از این رو می‌توان بر اساس رویکرد تحلیل ایستا، به اثبات وجود ویژگی‌های امنیتی در پروتکل پرداخت. از طرف دیگر، نفوذی نیز به عنوان یک نقش با دانش و قابلیت‌های خود می‌تواند به انجام استنتاج وگسترش مجموعه دانش خود دست بزند. به این ترتیب امکان دستیابی به سناریوهای نقض ویژگی‌های امنیتی توسط نفوذی یعنی تحلیل پویا نیز فراهم می‌شود. رسیدن به این سناریوها از طریق تعریف بعضی شرایط روی مسیر اجرا و دانش و قابلیت‌های نفوذی و بررسی درست بودن یا نبودن آن شرایط انجام می‌شود. شکل ۳ چگونگی انجام تحلیل را به کمک چارچوب مورد نظر نشان می‌دهد.

با تعریف اجزای چهارگانهٔ فوق، روشهای تحلیل مختلفی را می‌توان ارائه کرد. به عنوان مثال روش تحلیل GNY، مجموعه‌ای از نمادها برای توصیف پیام‌ها و دانش پروتکل دارد (معادل زبان L)، همچنین این روش، مدلی محاسباتی را از پروتکل ارائه کرده است (معادل $\mathcal{M}$). توصیفی صوری از این مدل محاسباتی به کمک زبان توصیف Z در [۱٤] ارائه شده است. مجموعه اصول تحلیل پروتکل در GNY معادل $\mathcal{PIR}$ است و $\mathcal{EIR}$ نیز در این روش تهی در نظر گرفته می‌شود.

٤- زبان توصیف $\mathfrak{L}$

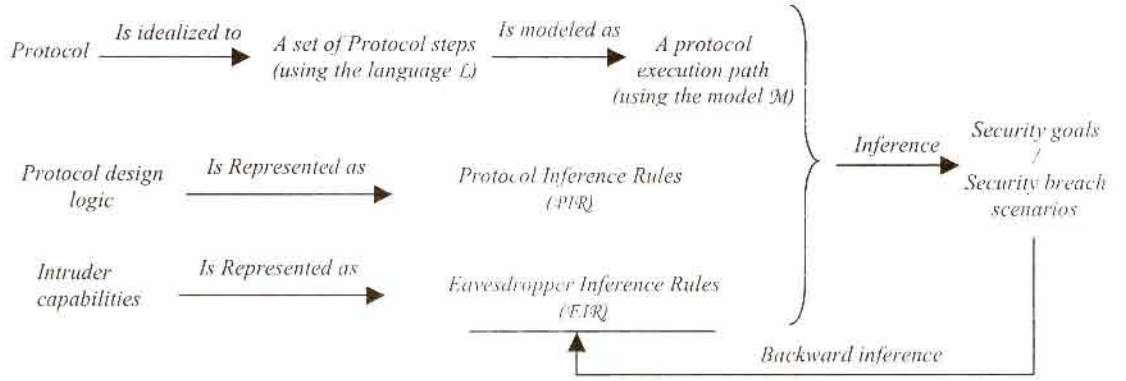
به‌کارگیری یک زبان مرتبه‌اول[1] برای بیشتر روشهای تحلیل کفایت می‌کند، هرچند بعضی از روشهای تحلیل مانند [۱۵] از زبان‌های مرتبه بالاتر[2] نیز استفاده کرده‌اند.

ما در اینجا زبان $\mathcal{L}$ را به عنوان زیر مجموعه‌ای از زبان منطق مسندات مرتبه‌اول[3] در نظر می‌گیریم. زبان $\mathfrak{L}$ شامل کلیهٔ جملاتی است که می‌توان با استفاده از گرامر زیر تولید کرد:

- الفبای زبان $\mathfrak{L}$ شامل نمادهای ثابت (مانند a,b,c,....)، نمادهای تابع (مانند id,nonce,cons,encrypt,...)، نمادهای متغیر (مانند X,Y,Z,...) و نمادهای مسند (مانند net, poss, believe,...) است. از حروف کوچک برای نشان دادن ثوابت و از حروف بزرگ برای نشان دادن متغیرها استفاده می‌کنیم.
- **پیام**[4] نمادی ثابت است، یا نمادی متغیر یا یک نماد تابع که روی یک چند تایی از پیام‌ها اعمال شده است. مانند a، X، id(a) یا encrypt(nonce(a),key(a,b)) .
- **دانش‌اتمیک**[5] . نمادی مسند است که روی یک چندتایی از پیام‌ها اعمال شده است. به عنوان مثال، poss(a,X) ، net(I,R,Y) یا believe(b,secret(a,b,key(a,b))) ، هر یک، دانش‌اتمیک هستند. فرض می‌کنیم که مقداردهی متغیرهای مورد استفاده، همگی به صورت مقداردهی عام[6] صورت می‌گیرد.
- **دانش‌خوش‌ساخت**[7]، یا دانش اتمیک است یا به یکی از اشکال (W) ، W1&W2 ، W1 or W2 ، یا ¬W ساخته می‌شود، که در آن W، W1 و W2 هر یک، دانشی خوش ساخت هستند. & ، or ، و ¬ به ترتیب، همان معانی and ، or و not در منطق را دارند.

در هر روش تحلیل باید مجموعه‌ای از نمادها به عنوان الفبای زبان $\mathcal{L}$ تعریف شود. توصیف پروتکل بر اساس مدل محاسباتی $\mathcal{M}$ و به کمک مجموعه پیام‌ها و دانش‌های خوش‌ساخت قابل تعریف توسط گرامر فوق انجام می‌شود.

1. First Order
2. Higher Order
3. First Order Predicate Logic Language
4. Message
5. Atomic knowledge
6. Universal quantification
7. Well-formed knowledge



شکل ۳ تحلیل ایستا و پویا به کمک چارچوب پیشنهادی

۵- مدل محاسباتی پروتکل

پروتکل مجموعه‌ای از نقشها است. هر نقش مشتمل بر چندین گام پی‌درپی است و هر گام رشته‌ای از توانایی‌ها است. توانایی نشان دهندهٔ ریزترین عملیات پایه‌ای[1] است که باید توسط نقش انجام شود. توانایی دارای دو مؤلفه است : مجموعه پیش‌شرطهای آن توانایی و مجموعه پس‌شرطهای آن توانایی. در صورت فراهم بودن پیش شرطهای توانایی، آن توانایی قابل اجرا است و با اجرای آن توانایی، پس‌شرطهای آن نیز فراهم می‌شود.

برای هر پروتکل یک مسیر اجرا تعریف می‌شود. مسیر اجرای پروتکل، حاصل تلفیق نحوی گامهای نقشهای پروتکل است. ساخت مسیر اجرای پروتکل، فرایندی نحوی است به این معنا که برای ایجاد مسیر اجرا، گامهای مختلف، براساس ضوابطی نحوی با یکدیگر مرتبط می‌شوند و در این مرحله، معنای پیش‌شرط و پس‌شرط توانایی‌ها دخالتی ندارند. در این بخش، به توصیف دقیق‌تر این مدل محاسباتی از پروتکل می‌پردازیم.

هر عامل پروتکل، برای انجام صحیح پروتکل، به انجام مجموعه‌ای از عملیات پایه نیاز دارد. هر یک از این عملیات پایه را توانایی می‌نامیم. هر توانایی برای قابل انجام بودن، مستلزم وجود بعضی شرایط در وضعیت پروتکل است و در صورت انجام، ممکن است وضعیت پروتکل را تغییر دهد.

تعریف ۱ (توانایی) : دوتایی $<pre , post>$ که در آن pre و post هر یک مجموعه‌ای از دانش‌ها با پارامترهای متغیر[2] در زبان $\mathfrak{L}$ هستند، تشکیل توانایی α را می‌دهند. توانایی α را با علامت $L_\alpha : \frac{pre}{post}$ نشان می‌دهیم که L_α برچسب نام توانایی است. Pre و post به ترتیب مجموعه پیش‌شرطها و پس‌شرطهای توانایی است و اگر α توانایی باشد، از $pre(\alpha)$ و $post(\alpha)$ به ترتیب برای نشان دادن این دو مجموعه استفاده می‌کنیم.

Send و Receive دو توانایی پایه است که دلالت بر ارسال یا دریافت یک پیام از/ به شبکه دارند. این دو توانایی، توانایی‌های اصلی در توصیف هر پروتکل بوده و برای تعریف آنها، از دو دانش خاص زیر استفاده می‌شود:

- دانش وضعیت به صورت $S_1(...),S_2(...),...$: نشان دهندهٔ به وجود آمدن وضعیت خاصی در پروتکل است.
- دانش انتقال پیام به صورت $\lhd(I,R,M)$: نشان دهندهٔ انتقال پیام خاصی مانند M از فرستندهٔ I به گیرندهٔ R است.

1. Primitive Operation

2. Non-ground

توانایی‌های Send و Receive را به صورت زیر می‌توان نشان داد:

$$Receive: \frac{\{\lhd(I',R',M'),S_j(\cdots)\}}{\{\cdots\}}$$

$$\text{Send}: \frac{\{\cdots\}}{\{\lhd(\text{I},\text{R},\text{M}),\text{S}_\text{i}(\cdots)\}}$$

پس‌شرط توانایی Send و پیش‌شرط توانایی Receive شامل دانش وضعیت و دانش انتقال پیام است. توانایی Send نشان می‌دهد که در صورت وجود پیش‌شرط‌های لازم، وضعیت جدیدی به وجود خواهد آمد و پیام خاصی روی شبکه قرار خواهد گرفت. در مقابل، توانایی Receive به این معنا است که در صورت به وجود آمدن وضعیت مناسب و به شرط انتقال پیام روی شبکه، توانایی Receive فعال می‌شود، می‌توان آن پیام را دریافت کرده و پس‌شرط‌های مناسب آن را ایجاد نمود.

لازم است ذکر شود که علاوه بر دو توانایی Send و Receive توانایی‌های دیگری را نیز می‌توان به عنوان عملیات پایهٔ قابل انجام در پروتکل در نظر گرفت. عملیاتی مانند تولید نانس، بررسی نانس، محاسبهٔ برخی مقادیر، رمزگذاری و رمزگشایی، تفکیک یا ترکیب پیام‌ها و موارد مشابه را می‌توان جزو توانایی‌های قابل انجام توسط عوامل پروتکل در نظر گرفت. تعیین نوع توانایی‌ها و میزان ریزدانگی آنها در توصیف پروتکل، به نوع پروتکل و هدف تحلیل بستگی دارد اما به هر حال وجود دو توانایی Send و Receive، حداقل نیازمندی در توصیف هر پروتکل است.

تعریف ۲ (گام): رشته‌ای از توانایی‌ها، نشان‌دهندهٔ یک گام اجرای پروتکل است. گام S را به صورت $\alpha_1.\alpha_2...\alpha_n$ نشان می‌دهیم که در آن α_1 تا α_n توانایی‌های تشکیل دهندهٔ گام S است.

اگر S یک گام باشد، از $\alpha_i(S)$ برای رجوع به توانایی i ام گام S استفاده می‌کنیم و |S| طول گام S را برحسب تعداد توانایی‌های آن نشان می‌دهد. پیش‌شرط اولین توانایی و پس‌شرط آخرین توانایی گام S به ترتیب نشان‌دهندهٔ ورودی و خروجی گام S است. ورودی گام S را با in(S) و خروجی این گام را با out(S) نشان می‌دهیم:

$$in(S) = pre(\alpha_1(S)) \ , \ out(S) = post(\alpha_{|S|}(S))$$

توانایی‌های اول و آخر گام S را به ترتیب با first(S) و last(S) نشان می‌دهیم. هر گام پروتکل، یا یک گام مقداردهی اولیه (S_{INIT}) است یا بر حسب ویژگیهای توانایی‌هایش، یکی از سه نوع گام زیر است:

- گام شروع (S_{START}) گامی است که با توانایی Receive شروع نشود.
- گام پایان (S_{STOP}) گامی است که با توانایی Send تمام نشود.
- گام واسط ($S_{INTERMED}$) گامی است که با توانایی Receive شروع و با توانایی Send تمام شود.

در هر گام واسط، دانش وضعیت در پس‌شرط توانایی Send ، فقط باید با دانش وضعیت در پیش‌شرط توانایی Receive گام دیگری (بجز خود این گام) قابل انطباق باشد. در بخشهای بعد، اهمیت این مسأله را در پایان‌پذیر بودن اجرای پروتکل ضمن قضیه‌ای نشان خواهیم داد.

تعریف ۳ (قابلیت دسترسی گام‌ها به یکدیگر): اگر S_i و S_j دو گام باشند ، S_j از S_i قابل دسترسی است اگر تابع جایگذاری[1] θ وجود داشته باشد به صورتی که $out(S_i)\theta = in(S_j)\theta$ ، یعنی $out(S_i)$ و $in(S_j)$ قابل یکسان‌سازی[2] باشند. در این حالت S_j گام بعد از S_i است، یعنی $S_j=succ(S_i)$.

تعریف ٤ (نقش) : رشتهٔ محدودی از گامها، یک نقش را تشکیل می‌دهند. اگر ρ نقشی به صورت $S_1.S_2...S_m$

1. Substitution Function
2. Unifiable

باشد، $S_i(\rho)$ نشان دهندهٔ گام i ام نقش ρ است.

تعریف ۵ (پروتکل) : پروتکل P سه تایی $<\Lambda, S, A>$ است که در آن Λ مجموعهٔ نقشهای پروتکل، S مجموعهٔ گامهای تشکیل دهنده این نقش‌ها و A مجموعهٔ توانایی‌های تشکیل دهندهٔ گامها است. اگر گامهای مقداردهی اولیه (S_{INIT}) و شروع (S_{START}) در مجموعه گامهای پروتکل P، به کمک تابع جایگذاری σ،به شکل پایه[1] درآیند، به ماحصل، نمونهٔ پروتکل P به ازاء σ می‌گوییم و آن را با P_σ نشان می‌دهیم.

تعریف ٦ (مسیر اجرای پروتکل): اگر $P=<\Lambda, S, A>$ پروتکل باشد، R(P) مسیر اجرای پروتکل P ، رشته محدودی از گامهای درهم شده از نقش‌های مختلف P است که به صورت زیر ساخته می‌شود:

- گام اول رشته، گام مقداردهی اولیهٔ پروتکل است (S_{INIT}) .
- گام دوم رشته، گام شروع پروتکل است (S_{START}) .
- گامهای سوم تا آخر این مسیر اجرا، سایر گامهای نقشهای Λ هستند، به صورتی که هر گام از گام بلافاصله قبل از آن قابل دسترسی است و تمام توابع جایگذاری اعمال شده روی عناصر زیر رشتهٔ سمت چپ آنها نیز روی آنها اعمال شده است.
- هیچ دو گام پی‌درپی در مسیر اجرا متعلق به یک نقش نیستند.
- رعایت تقدم و تأخر گامهای هر نقش، در رشتهٔ اجرا نیز به قوت خود باقی است.

مسیر اجرایی R(P) را به طور دقیق‌تری چنین می‌توان توصیف کرد:

$$R(P) = t_0.t_1.(t_2\delta_1).(t_3\delta_1\delta_2). \ \ldots \ .(t_n\delta_1..\delta_{n-1})$$

$$t_0 = S_{INIT}$$

$$t_1 = S_{START}$$

$$\forall i \geq 1, \exists \rho', \rho'' \in \Lambda,$$
$$\exists k', k'', \delta_i \bullet S_{k'}(\rho') = t_i, S_{k''}(\rho'') = t_{i+1},$$
$$out(t_i\delta_1..\delta_i) = in(t_{i+1}\delta_1..\delta_i),$$
$$\rho' \neq \rho''$$

$$\forall i, j \exists \rho' \in \Lambda, \exists k', k'' \bullet S_{k'}(\rho') = t_i, S_{k''}(\rho') = t_j,$$
$$k' < k'' \Rightarrow i < j$$

٦- تفسیراجرایی از مدل محاسباتی پروتکل

برای اینکه بتوان از مدل محاسباتی ارائه شده در بخش قبل به منظور تحلیل خودکار پروتکل استفاده کرد، لازم است تفسیری اجرایی از این مدل ارائه شود. یک روش برای انجام این کار، نگاشت مدل محاسباتی ارائه شده به سیستم‌هایی مانند منطق یا جبر است. با این کار می‌توان پروتکل‌ها را بر اساس مدل محاسباتی مورد نظر اما به کمک زبان توصیف منطق یا جبر توصیف کرد و از این‌رو می‌توان مستقیماً از ابزارهای خودکار اثبات قضیه[2] برای انجام تحلیل‌های مورد نظر بر اساس مدل محاسباتی استفاده کرد.

در این بخش به ارائهٔ تفسیری اجرایی از مدل محاسباتی مطرح شده در بخش قبل بر اساس سیستم‌های بازنویسی جبری[3] می‌پردازیم. توصیف پروتکل در قالب ارائهٔ مجموعه‌ای از گام‌های مختلف هر نقش از پروتکل صورت می‌گیرد. سپس این مجموعه از گامهای پروتکل بر اساس الگوریتم ارائه شده، به مجموعه‌ای از قوانین بازنویسی نگاشت می‌شود. با استفاده از این قوانین بازنویسی و بر اساس سازوکار محدودسازی[4] ، مسیر اجرای پروتکل را می‌توان ساخت. این تفسیر اجرایی به گونه‌ای است که مستقیما توسط ابزارهای خودکار قابل اجرا است. تولید و

1. Ground

2. Theorem Prover

3. Algebraic Term Rewriting Systems

4. Narrowing

اجرای قوانین بازنویسی در عمل به کمک قابلیت‌های بازنویسی موجود در ابزار اثبات قضیهٔ Otter [۱۶] انجام شده است.

در ادامهٔ این بخش، نخست مفاهیم اصلی در سیستم‌های بازنویسی جبری مرور می‌شود، سپس چگونگی توصیف پروتکل، الگوریتم ساخت قوانین بازنویسی و نحوهٔ ساخت مسیر اجرا به کمک این قوانین بیان می‌شود. چگونگی نگاشت توصیف پروتکل نمونهٔ Woo-Lam به مجموعه قوانین بازنویسی نشان داده شده است.

۶-۱- سیستم بازنویسی جبری

فرض کنید F مجموعه‌ای محدود از نمادها است که به هریک از این نمادها، تابعی معرف تعداد آرگومانهای آن اختصاص داده شده است. همچنین فرض کنید که V مجموعه‌ای قابل شمارش از متغیرها است. مجموعهٔ T(F,V) مجموعه عبارتهای[1] قابل تعریف روی F و V است. T(F) مجموعه عبارتهای پایه (بدون متغیر) است. موقعیت[2] زیر عبارتها در هر عبارت به صورت رشته‌ای از اعداد صحیح نشان داده می‌شود. اگر عبارت t به صورت $t=f(t_1,...t_n) \notin V$ باشد،مجموعه موقعیتهای عبارت t، به صورت زیر تعریف می‌شود:

$$O(t) = \{\varepsilon\} \cup \{ i.p \mid 1 \le i \le n \text{ and } p \in O(t_i)\}$$

که در آن ε رشته‌ای تهی و نشان دهندهٔ بالاترین موقعیت در عبارت t است. اگر $p \in O(t_i)$ ، آنگاه $t_{|p}$ نشان‌دهندهٔ زیر عبارت t در موقعیت p است و $t[s]_p$ نشان‌دهندهٔ عبارتی است که از جایگزین کردن زیر عبارت $t_{|p}$ در موقعیت p با عبارت s به‌دست می‌آید. مجموعهٔ متغیرهای عبارت t را به صورت Var(t) نشان می‌دهیم.

سیستم بازنویسی عبارت R ، مجموعه‌ای از قوانین بازنویسی به صورت $l \to r$ است که درآن، $l,r \in T(F,V)$ ، $l \notin V$ و $Var(l) \supseteq Var(r)$. رابطهٔ بازنویسی تحت R ($\to_R$) به صورت زیر تعریف می‌شود: برای هر $s,t \in (F,V)$ ، $s \to_R t$ اگر قانونی مانند $l \to r$ در R، موقعیتی مانند $p \in O(s)$ و یک تابع جایگذاری مانند σ وجود داشته باشد، به قسمی که $l\sigma = s_{|p}$ و $t=s[r\sigma]_p$. $s \to_R t$ یک گام بازنویسی نامیده می‌شود. اگر $s \to^*_R t$ ($\to^*_R$ بستار تراگذری[3] $\to_R$ است) و s غیر قابل بازنویسی تحت R باشد، به s شکل نهایی[4] t با توجه به R گوئیم و با $t\downarrow_R$ یا به صورت ساده‌تر $t\downarrow$ نشان می‌دهیم و R با توجه به مضمون، قابل تشخیص خواهد بود.

عبارت t قابل محدودشدن[5] به t' است اگر و فقط اگر موقعیتی مانند $p \in O(t)$ ، یک نسخه از $l \to r$ با متغیرهای تغییرنام یافته از قوانین R و یک تابع جایگذاری σ وجود داشته باشد، به قسمی که σ عمومی‌ترین یکسان‌ساز[6] برای $t_{|p}$ و l باشد و $t'=\sigma(t[r]_p)$. $t \leadsto t'$ نشان دهندهٔ محدود شدن t به t' و رابطهٔ $\leadsto$، محدودسازی نامیده می‌شود.

یک مسألهٔ مهم در مورد قوانین بازنویسی، اثبات همگرایی[7] رابطهٔ بازنویسی تحت مجموعه‌ای از قوانین است. رابطهٔ $\to_R$ همگرا است اگر به ازای هر $a,b,c \in T(F,V)$ که $a \to_R b$ و $a \to^*_R c$ ، یک $d \in T(F,V)$ وجود داشته باشد، به قسمی که $b \to^*_R d$ و $c \to^*_R d$. اثبات همگرایی رابطهٔ بازنویسی را می‌توان به اثبات وجود رابطهٔ ساده‌تر همگرایی محلی[8] و ویژگی پایان پذیری[9] برای رابطه مورد نظر تبدیل کرد [۱۷]. روش دیگر اثبات همگرایی، استفاده از ویژگی

1. Terms
2. Position
3. Transitive Closure
4. Normal Form
5. Narrow
6. Most General Unifier
7. Confluent
8. Local confluent
9. Termination

متعامد[1] بودن سیستم بازنویسی است. هر سیستم بازنویسی متعامد، همگرا است[۱۸] .

۶-۲- سیستم بازنویسی توصیف پروتکل

همانطور که پیشتر نیز گفته شد، مدل محاسباتی $\mathcal{M}$ در چارچوب تحلیل پروتکل برای توصیف پیام‌های پروتکل و متغیرهای وضعیت اجرای آن (دانش محلی ایجاد شده برای عوامل پروتکل)، از زبان توصیف $\mathcal{L}$ استفاده می‌کند. بر این اساس در اینجا مجموعه نمادهای مورد استفاده در سیستم بازنویسی یعنی مجموعهٔ F، شامل کلیهٔ نمادهای تعریف شده در زبان $\mathcal{L}$ است. کلیهٔ مسندات در زبان $\mathcal{L}$ در اینجا به صورت نمادهای تابع در نظر گرفته می‌شود. اگر مجموعهٔ تمام عبارتهای قابل تعریف در زبان $\mathcal{L}$ را با T_L نشان دهیم، در این صورت $T(F,V) \supseteq T_L$. مجموعهٔ متغیرهای V نیز به ازای هر عبارت $t \in T(F)$ شامل یک متغیر x_i خواهد بود.

بر طبق تعریف ۵، پروتکل شامل مجموعه‌ای از نقش‌ها، مجموعه‌ای از گامها و مجموعه‌ای از توانایی‌ها است. بر این اساس مجموعه‌ای از نمادهای ثابت مانند ...a,b,s, را برای نشان دادن نقش‌های پروتکل و دو نماد ab و st را به ترتیب برای نشان دادن توانایی و یک گام، جزء مجموعه نمادهای F در نظر می‌گیریم. اگر $\mathbb{N}$ نشان دهندهٔ مجموعهٔ اعداد طبیعی و Roles مجموعه نمادهای در نظر گرفته شده برای نقش‌های پروتکل باشد و فهرستی از عبارتهای متعلق به مجموعهٔ مفروض X را در قالب یک تعریف عام[2] به صورت زیر نشان دهیم:

List(X)=[] U [X|List(X)]

در این صورت هر یک از مجموعه عبارتهای Abilities و Stages با تعاریف زیر نیز متعلق به مجموعهٔ T(F,V) خواهد بود:

$$Abilities = \{ ab(X,Y) \mid X,Y \in List(T_L) \}$$
$$Stages = \{ st(N,R,A) \mid N \in \mathbb{N}, R \in Roles, A \in List(Abilities) \}$$

1. Orthogonal
2. Generic definition

هر توانایی ab دارای دو فهرست از عبارتها در زبان $\mathcal{L}$ به عنوان پیش‌شرط و پس‌شرط است. هر گام st از پروتکل شامل یک شماره گام ، نقشی که آن گام متعلق به آن است و فهرستی از توانایی‌های تشکیل دهندهٔ آن گام است. برای توصیف پروتکل کافی است فهرست گام‌های پروتکل را به کمک داده‌ساز[3]های فوق در قالب مجموعه‌ای از نمادهای st نشان دهیم. به عنوان مثال، عبارت زیر نشان دهندهٔ گام دوم نقش I در یک پروتکل است (که در اجرا مقدار واقعی آن مشخص خواهد شد):

```
st(2,I,[ab([net(R,I,X),s2], [poss(I,X)] ),
    ab([poss( I,[encrypt(X,key(I,R))])],
       [net(I, R,[id(I),encrypt(X,key(I,R))])], s3 )]).
```

توانایی اول این گام بیان می‌کند که به شرط ارسال پیامی مانند X از عاملی مانند R به I (net) و بودن در وضعیت s2 ، نقش I مالک X خواهد شد (poss). در توانایی دوم، به شرط اینکه I بتواند مقدار رمز شدهٔ X را با کلید مشترک بین خود و R تهیه کرده و مالک آن شود، در این صورت می‌تواند آن را همراه با شناسهٔ خود برای R ارسال کرده، و وضعیت را به s3 تغییر دهد.

تهیهٔ فهرست گام‌های پروتکل به صورت فوق، در واقع مرحلهٔ ایده‌آل‌سازی پروتکل برای انجام تحلیل به شمار می‌آید. بعد از این مرحله از روی فهرست گام‌های پروتکل و به کمک یک الگوریتم خاص، دو مجموعه قوانین بازنویسی تولید می‌شود. از این دو مجموعه به‌ترتیب برای تولید دو مسیر اجرای رو به جلو و رو به عقب استفاده می‌شود. در تحلیل پروتکل، از مسیر اجرای رو به جلو برای ایجاد دانش پروتکل و تحلیل ایستا و همچنین ایجاد دانش نفوذی غیرفعال و تحلیل عملکرد وی استفاده می‌شود. از مسیر اجرای رو به عقب نیز در تحلیل نقش نفوذی فعال و مدلسازی قابلیت وی در فریب عوامل پروتکل استفاده می‌شود.

برای ارائهٔ الگوریتم ساخت قوانین بازنویسی از روی

3. Data constructor

مجموعه گام‌های پروتکل، به تعریف چند تابع نیاز داریم. سه تابع append ،first و last ، به‌ترتیب توابعی برای اتصال دو فهرست عمومی و برگرداندن عناصر اول و آخر فهرست هستند: first,last:

$$List(X) \rightarrow X$$

append: $List(X) \times List(X) \rightarrow List(X)$

تعریف بدنهٔ این توابع به کمک روابط بازنویسی ساده است. دو تابع pre,post: Abilities$\rightarrow$ List(T_L) نیز به‌ترتیب فهرست عبارتهای پیش‌فرض و پس‌شرط توانایی را برمی‌گردانند:

$$pre(ab(X,Y)) = X \quad , \quad post(ab(X,Y)) = Y$$

مجموعه قوانین بازنویسی تعریف کنندهٔ این پنج تابع را PRIMITIVES می‌نامیم.

دو تابع match و bmatch هر یک فهرستی از عبارتها را به فهرستی از گام‌ها نگاشت می‌کنند:

match, bmatch: $List(T_L) \rightarrow List(Stages)$

وظیفهٔ این دو تابع، اتصال گام‌هایی از پروتکل به یکدیگر است که بر اساس تعریف ۳ از یکدیگر دسترسی‌پذیر هستند. match این عمل را از گام شروع آغاز کرده و تا گام پایان ادامه می‌دهد. bmatch این کار را به صورت معکوس از گام پایان به سمت گام شروع انجام می‌دهد. به عبارت دیگر match و bmatch مسیر اجرای پروتکل را به‌ترتیب به صورت رو به جلو و رو به عقب می‌سازند. تابع match با دریافت پس‌شرط آخرین توانایی یک گام (غیر از گام پایان)، گام بعدی (گام دسترسی پذیر) آن را برمی‌گرداند. در مقابل، تابع bmatch با دریافت پیش‌شرط اولین توانایی از هر گام (بجز گام شروع)، گام قبلی را (گامی که به این گام دسترسی دارد) بر می‌گرداند. اگر نماد $\downarrow$ نشان دهندهٔ شکل نهایی عبارت قبل از خود تحت قوانین PRIMITIVES باشد و از عملگر . برای اتصال عناصر فهرست استفاده کنیم، ساخت مسیرهای اجرای مستقیم و معکوس به‌ترتیب از طریق مجموعه قوانین بازنویسی FORWARD-PATH و BACKWARD-PATH به صورت زیر امکان‌پذیر است:

FORWARD-PATH:

$protocol \rightarrow S_{INIT} \,.\, S_{START}.match(post(last(S_{START})))$

$match(pre(first(S_i))\downarrow) \rightarrow S_i \,.\, match(post(last(S_i)))$

(for all i in which S_i is an intermediate step)

$match(pre(fisrt(S_{STOP}))\downarrow) \rightarrow S_{STOP}$

BACKWARD-PATH:

$bprotocol \rightarrow bmatch(pre(first(S_{STOP}))) \cdot S_{STOP}$

$bmatch(post(last(S_i))\downarrow) \rightarrow bmatch(pre(fisrt(S_i))) \,.S_i$

(for all i in which S_i is an intermediate step)

$bmatch(post(last(S_{START}))\downarrow) \rightarrow S_{START} \cdot S_{INIT}$

برای ساخت مسیر اجرا به کمک مجموعه قوانین فوق به صورت رو به جلو یا رو به عقب، کافی‌است به‌ترتیب از protocol و bprotocol شروع کنیم. با اعمال گام‌های پی‌درپی محدودسازی، تحت قوانین فوق و قوانین PRIMITIVES نهایتاً این دو به شکل نهایی خود خواهند رسید. شکل نهایی protocol و bprotocol ، به‌ترتیب مسیرهای اجرای مستقیم و معکوس خواهند بود:

$Protocol \rightsquigarrow^{*} (protocol)\downarrow$,

$Bprotocol \rightsquigarrow^{*} (bprotocol)\downarrow$

الگوریتم شکل ٤ با توجه به ایدهٔ فوق، نحوهٔ تولید قوانین بازنویسی FORWARD-PATH و BACKWARD-PATH را از روی مجموعه گام‌های پروتکل نشان می‌دهد. در تولید قوانین از عملگرهای مرسوم فهرست و توابع تعریف شده در فوق استفاده شده است.

٦-٣- توصیف یک پروتکل نمونه

در این بخش پروتکل احراز اصالت یکطرفهٔ Woo-Lam [۱۹] را به عنوان نمونه بر اساس مدل محاسباتی و به کمک تفسیر اجرایی ارائه شده در بخش قبل توصیف می‌کنیم.

پروتکل Woo-Lam: پروتکل احراز اصالت یکطرفه Woo-Lam در شکل ٥ نشان داده شده است. سه نقش آغازگر (A)، مخاطب (B) و میزبان (S) در این پروتکل عمل می‌کنند. نقش آغازگر در گام اول پروتکل با ارسال هویت خود به مخاطب، تقاضای احراز اصالت می‌کند. مخاطب با ارسال نانس N_b ، وی را به چالش[1] فرا می‌خواند. درگام سوم، آغازگر پیام چالش N_b را با کلید

1. Challenge

Algorithm: GEN-TRS
INPUT: A set of protocol steps
OUTPUT: two term rewriting rule sets FORWARD-PATH and BACKWARD-PATH.

Let FORWARD-PATH = ∅,
BAKWARD-PATH = ∅
Let s_{INIT} *be the initialization step of the protocol and*
S be the set of all other steps of the protocol.
For each $s \ni S$
Let $m_1 = post(last(s))$, $m_2 = pre(first(s))$.
if s is the start step *then*
Add "protocol_path → $[s_{INIT} \mid append([s], match(m_1))]$*" to FORWARD-PATH.*
Add "bmatch(m1) → $[s, s_{INIT}]$*" to BACKWARD-PATH.*
else if s is the stop step *then*
Add "match(m2) → s" to FORWARD-PATH.
Add "bprotocol_path → append(bmatch(m2), [s])" to BACKWARD-PATH.
else
Add "match(m2) → [s | match(m1)]" to FORWARD-PATH.
Add "bmatch(m1) → append (bmatch(m2), [s])" to BACKWARD-PATH.

شکل ٤ الگوریتم تولید مجموعه قوانین بازنویسی برای ساخت مسیر اجرا

(1) $A \rightarrow B : A$
(2) $B \rightarrow A : N_b$
(3) $A \rightarrow B : \{N_b\}_{Kas}$
(4) $B \rightarrow S : \{A, \{N_b\}_{Kas}\}_{Kbs}$
(5) $S \rightarrow B : \{N_b\}_{Kbs}$

شکل ٥ پروتکل احراز اصالت Woo-Lam

مشترک بین خود و میزبان رمز کرده و برای مخاطب ارسال می‌کند. مخاطب در این هنگام هیچ چیز از این پیام نمی‌فهمد. به همین دلیل در گام چهارم، کل این پیام را همراه با شناسهٔ عامل مدعی هویت (یعنی آغازگر) با کلید مشترک بین خود و میزبان رمزکرده، برای میزبان ارسال می‌کند. میزبان توانایی رمزگشایی هر دو بستهٔ تودرتو را دارد. لذا داخلی‌ترین پیام یعنی N_b را به دست آورده و این بار با کلید مشترک بین خود و مخاطب آن را رمزکرده و برای وی ارسال می‌کند. در این هنگام مخاطب توانایی بازیابی پیام چالش خود را که در گام دوم برای آغازگر فرستاده بود پیدا می‌کند و بدین ترتیب آغازگر برای مخاطب احراز اصالت می‌شود.

توصیف پروتکل بر اساس مدل محاسباتی: با توجه به مطالب بخش قبل، پروتکل Woo-Lam باید در قالب مجموعه‌ای از گام‌ها که هر یک به شکل

st(N,R,Ability_list) نشان داده می‌شود توصیف شود. N شماره گام، R نقشی است که این گام متعلق به آن است و Ability_list فهرستی از توانایی‌ها به شکل ab(Pre,Post) است که در آن، Pre و Post هر یک فهرستی از دانش‌های اتمیک در زبان $\mathcal{L}$ هستند. مجموعه نقش‌های پروتکل Woo-Lam شامل سه نقش آغازگر، مخاطب و میزبان است که به‌ترتیب آنها را با a و b و s نشان می‌دهیم.

علاوه بر این، نقش ساختگی all را نیز برای رجوع به همهٔ نقش‌ها در نظر می‌گیریم. چهار دانش اتمیک زیر برای توصیف ویژگی‌های پروتکل Woo-Lam در نظر گرفته شده است:

- poss(R,M) : نقش R ، فهرست پیام‌های M را در تملک خود دارد.
- believe(R,M) : نقش R پیام M را باور دارد.
- net(I,R,M) : پیام M برای انتقال از نقش I به نقش R روی شبکه قرار می‌گیرد.
- s1 تا s6 : شش دانش وضعیت هستند.

پیام‌های پروتکل Woo-Lam نیز به یکی از شکلهای زیر است: id(R) (شناسهٔ نقش R)، key(I,R) (کلید مشترک بین نقش‌های I و R)، secret(I,R,M) (پیام M که برای نقش‌های I و R مشترک و مخفی است)، encrypt(M,K) (پیام رمز شده M با کلید K)، decrypt(M,K) (پیام رمزگشایی شده M با کلید K)، nonce(M) (پیام نانس M) و fresh(M) (پیام تازهٔ M). لازم است ذکر شود که فهرستی از پیام‌ها را به صورت متعارف [m,n,...] نشان می‌دهیم.

فرضیات اولیه پروتکل Woo-Lam باید در قالب یک گام مقداردهی اولیه (S_{INIT}) تعیین شود. این فرضیات، دانش و باورهای اولیهٔ همهٔ نقشها را تعیین می‌کند. نقشهای آغازگر و مخاطب پروتکلWoo-Lam دارای کلیدهای اشتراکی با میزبان هستند و مخفی بودن این کلیدها با میزبان را باور دارند. میزبان نیز هر دو کلید مشترک را دارد و مخفی بودن هر یک از آنها را با آغازگر یا مخاطب باور دارد. بعلاوه برای شروع پروتکل لازم است آغازگر شناسهٔ خود را نیز بداند. گام مقداردهی اولیه با شمارهٔ 0 و متعلق به نقش ساختگی all (همهٔ نقش‌ها) در نظر گرفته می‌شود. عبارت زیر گام مقداردهی اولیهٔ پروتکل را نشان می دهد. جملات بعد از علامت % فقط برای توضیح نوشته شده‌اند:

```
st(0,all,[ab([],              % Initial ability of role a
          [poss(a,[id(a)]),poss(a,[key(a,s)]),
           believe(a,secret(a,s,key(a,s)))] ),
         ab([],               % Initial ability of role b
          [poss(b,[key(b,s)]),
           believe(b,secret(b,s,key(b,s)))] ),
         ab([],               % Initial ability of role s
          [poss(s,[key(a,s)]),
           believe(s,secret(a,s,key(a,s))),
           poss(s,[key(b,s)]),
           believe(s,secret(b,s,key(b,s)))] )
  ] ).
```

لازم است ذکر شود که ما در اینجا "نمونه‌ای" از پروتکل Woo-Lam را توصیف می‌کنیم و لذا گام مقداردهی‌اولیه و همچنین گام شروع پروتکل با دانش‌های ground تعریف می‌شود (رجوع به تعریف پروتکل در بخش٤). نقش آغازگر پروتکل مشتمل بر دو گام است که گام اول آن گام شروع پروتکل(S_{START}) است:

```
st(1,a,[ ab([poss(a,[id(a)])],                %Send
          [net(a,b,[id(a)]), s1] )
       ] ).
st(2,I ,[ab([net(R,I,X),s2],                  %Receive
          [poss(I,X)] ),
        ab([poss(I, [encrypt(X,key(I,s))])],  %Send
          [net(I, R,[encrypt(X,key(I,s))])], s3 )
        ] ).
```

گام اول نقش آغازگر فقط توانایی Send را دارد. این گام را براساس دانش پیش‌شرط و پس‌شرط آن اینگونه می‌توان تعبیر کرد که اگر a شناسهٔ خود را داشته باشد، این توانایی قابل انجام است و با انجام آن، این شناسه از طرف وی برای مخاطب b ارسال می‌شود و همچنین وضعیت s1 خواهد بود. گام دوم نقش آغازگر دارای دو توانایی Receive و Send است. به شرط قرار گرفتن در وضعیت s2 و قرار گرفتن پیامی مانند X روی شبکه، از طرف فرستندهٔ R به دریافت کنندهٔ I، I مالک پیام X خواهد شد.

در صورتی که I بتواند X را با کلید مشترک بین خود و میزبان s رمز کند (و آن را مالک شود)، آن را برای انتقال به s روی شبکه خواهد گذاشت و وضعیت نیز S_3 خواهد شد.

نقش مخاطب در پروتکل Woo-Lam شامل سه گام زیر است:

```
st(1,R,[ab([net(I,R,[id(I)]),s1],          %Receive
          [poss(R,[id(I)])] ),
        ab([],                             %Generate nonce
          [poss(R,[nonce(1)]),
           believe(R,fresh(nonce(1)))] ),
         ab([poss(R,[nonce(1)])],          %Send
            [net(R,I,[nonce(1)]),s2] )
    ] ).
st(2,R,[ab([net(I,R,Y),s4],                %Receive
          [poss(R,Y)]),
        ab(                                %Send
          [poss(R,[encrypt(cat([id(I)],Y),key(R,s))])],
        [net(R,s,[encrypt(cat([id(I)],Y),key(R,s))]),s5])
        ]).
st(3,R,[ab([net(s,R,[encrypt(X,key(R,s))]),s6],%Receive
          [poss(R,[encrypt(X,key(R,s))])] )
    ]).
```

مخاطب در گام اول خود سه توانایی دارد، ابتدا شناسهٔ عامل I را دریافت می‌کند، سپس نانس nonce(1) را تولید می‌کند و در پایان آن را برای مخاطب R ارسال می‌کند. تولید نانس توسط مخاطب به این صورت است که وی بدون هیچ پیش‌شرطی مالک nonce(1) می‌شود. علاوه بر این وی معتقد است که این نانس ایجاد شده یک پیام تازه است. دقت کنید که دانش‌های وضعیت s1 و s2 به گونه‌ای هستند که جایگاه گام اول مخاطب پس از گام اول آغازگر و پیش از گام دوم وی قرار گیرد. گام دوم نقش مخاطب مشتمل بر دو توانایی است. ابتدا در وضعیتی که با s4 تطابق پیدا می‌کند پیامی مانند Y را دریافت می‌کند و سپس در صورتی که بتواند رمز شدهٔ ترکیب شناسهٔ عامل I و پیام Y با کلید مشترک بین خود و نقش مشخص میزبان را فراهم کند، آن را برای میزبان می‌فرستد. گام سوم نقش مخاطب، گام پایان پروتکل است. در این گام او پیام رمز شدهٔ X با کلید مشترک خود و میزبان را دریافت و آن را مالک می‌شود.

میزبان پروتکل Woo-Lam فقط دارای یک نقش است. در این نقش، وی در وضعیت s5 پیام رمز شده‌ای را دریافت می‌کند، سپس در طی دو مرحله رمز گشایی به پیام X دست پیدا می‌کند و نهایتاً پیام X را با کلید مشترک بین خود و عامل R رمز نموده و ارسال می‌کند:

```
st(1,S,[ab(                                %Receive
          [net(R,S,[encrypt([id(I),
             encrypt(X,key(I,S))],key(R,S))]),s5],
          [poss(S,[encrypt([id(I),
             encrypt(X,key(I,S))],key(R,S))])] ),
       ab(                                 % Decrypt
          [poss(S,[decrypt([encrypt([id(I),
         encrypt(X,key(I,S))],key(R,S))],key(R,S))])],
          [poss(S,[encrypt(X,key(I,S))])] ),
    ab(                                    %Decrypt
   [poss(S,[decrypt([encrypt(X,key(I,S))],key(I,S))])],
   [poss(S,X)] ),
    ab([poss(S,[encrypt(X,key(R,S))])],    %Send
       [net(S,R,[encrypt(X,key(R,S))]),s6] )
]).
```

با اعمال الگوریتم GEN-TRS روی مجموعه گام‌های فوق، مجموعه‌ای از قوانین بازنویسی ایجاد می‌شود که می‌توان از آنها برای ایجاد مسیرهای اجرای مستقیم و معکوس پروتکل استفاده کرد. بخشی از قوانین تولید شده توسط الگوریتم به صورت زیر است. به منظور صرفه‌جویی در فضا، لیست توانایی‌های هر گام با سه نقطه نشان داده شده است:

```
protocol_path = [st(0,all,[...]) |
                 append([st(1,a,[...])],
                        match([net(a,b,[id(a)]),s1]) ) ].

bmatch([net(a,b,[id(a)]),s1])= st(1,a,[...],st(0,all,[...])].

match([net(R,I,X),s2]) =  [st(2,I,[...]) |
          match([net(I,R,[encrypt(X,key(I,s))]),s3]) ].
bmatch([net(a,R,[encrypt(X,key(a,s))]),s2])=
          append(bmatch([net(R,a,X)]),[st(2,a,[...])] ).
                              ...
```

با اجرای قوانین فوق تحت سیستم اثبات قضیه با امکانات بازنویسی و narrowing ، مسیرهای اجرای پروتکل ایجاد می‌شود. قوانین فوق تحت سیستم Otter [۱۶] اجرا شده و

مسیر اجرای پروتکل Woo-Lam بدست آمده است. این مسیر اجرا به عنوان ورودی بخش تحلیل پروتکل مورد استفاده قرار می‌گیرد.

٦-٤- همگرایی سیستم بازنویسی توصیف پروتکل

سیستم بازنویسی توصیف پروتکل از سه مجموعه قوانین FORWARD-PATH، BACKWARD-PATH و PRIMITIVES تشکیل شده است. چون این سه مجموعه از یکدیگر مجزا هستند، برای نشان دادن همگرایی کل سیستم می‌توانیم همگرا بودن هر سه را نشان دهیم. در این بخش ابتدا در طی دو قضیه نشان می‌دهیم که مجموعه قوانین FORWARD-PATH و BACKWARD-PATH پایان‌پذیر و همگرای محلی هستند. سپس همگرایی کل سیستم را نشان می‌دهیم.

قضیه ۱: مجموعه قوانین FORWARD-PATH و BACKWARD-PATH پایان‌پذیر هستند.

اثبات: برای اثبات پایان پذیری هر یک از مجموعه‌های فوق، یک ترتیب جزئی[1] روی مجموعه نمادهای F ارائه می‌کنیم و نشان می‌دهیم که این ترتیب جزئی یک Recursive path ordering (rpo) [۲۰] را روی مجموعه عبارتهای T(F,V) هر یک ایجاد می‌کند.

همانگونه که در بخش ۵ ضمن تعریف توانایی و گام گفتیم، در هر گام واسط، دانش وضعیت در پس شرط توانایی Send ، فقط باید با دانش وضعیت در پیش شرط توانایی Receive گام دیگری (بجز خود این گام) قابل انطباق باشد. به بیان رسمی‌تر هیچ یک از عبارتهای $pre(first(S_i))\downarrow$ و $post(last(S_i))\downarrow$ (برای تمام i هایی که S_i یک گام میانی است) در تعریف گام‌های پروتکل با یکدیگر منطبق نمی‌شوند و هر $pre(first(S_i))\downarrow$ فقط با یک $post(last(S_j))\downarrow$ منطبق می‌شود. پس‌شرط توانایی Send و پیش‌شرط توانایی Receive شامل دو دانش وضعیت و انتقال پیام است. دانش‌های وضعیت در عمل وظیفهٔ منحصر کردن انطباق پس‌شرط با فقط یک پیش شرط دیگر را به عهده دارند.

با توجه به تعریف مجموعه‌های Stages و Abilities و قوانین PRIMITIVES ، هر یک از عبارتهای $pre(first(S_i))\downarrow$ و $post(last(S_i))\downarrow$ (برای تمام i هایی که S_i یک گام میانی است)، فهرستی از عبارتهای T_L شامل دو زیر عبارت پیام و وضعیت هستند. فرض کنیم پروتکل n گام میانی داشته باشد. همچنین بدون کاستن از کلیت بحث، فرض کنیم عبارتهای دوم هر یک از این فهرست ها عبارتهای وضعیت هستند که در انطباق فهرستها نقش تعیین کننده‌ای دارند؛ یعنی :

for all i in which S_i is an intermediate step:

$$pre(first(S_i))\downarrow = \alpha'_i.\beta'_i \ ,$$

$$post(last(S_i))\downarrow = \alpha''_i.\beta''_i \ , \ \beta'_i \neq \beta''_i \ ,$$

for all other $j \neq i$, there exist only one j such that:

$$\beta'_i = \beta''_j$$

فرض کنیم $post(last(S_{START}))$ و $pre(first(S_{STOP}))$ نیز دارای شکل‌هایی نهایی به صورت زیر باشند:

$$post(last(S_{START}))\downarrow = \alpha_{START}.\beta_{START} \ ,$$

$$pre(first(S_{STOP}))\downarrow = \alpha_{STOP}.\beta_{STOP}$$

بر این اساس، دو مجموعه قوانین مورد بحث را می‌توان به صورت زیر نوشت:

FORWARD-PATH:

$$protocol \rightarrow S_{INIT}.S_{START} \ . \ match(\alpha_{START}.\ \beta_{START})$$

$$match(\alpha'_i.\beta'_i) \rightarrow S_i \ . \ match(\alpha''_i.\beta''_i)$$

(for all i in which S_i is an intermediate step)

$$match(\alpha_{STOP}.\beta_{STOP}) \rightarrow S_{STOP}$$

BACKWARD-PATH:

$$bprotocol \rightarrow bmatch(\alpha_{STOP}.\beta_{STOP}) \ . \ S_{STOP}$$

$$bmatch(\alpha''_i.\beta''_i) \rightarrow bmatch(\alpha'_i.\beta'_i) \ . \ S_i$$

(for all i in which S_i is an intermediate step)

$$bmatch(\alpha_{START} \ . \ \beta_{START}) \rightarrow S_{START} \ . \ S_{INIT}$$

1. Partial ordering

می‌دانیم که عبارتهای β در قوانین فوق دانش‌های وضعیت هستند هر یک از این دانش‌های وضعیت یک مسند قابل تعریف در زبان $\mathcal{L}$ است (که البته در اینجا به عنوان یک تابع در مجموعه نمادهای F شناخته می‌شوند). فرض کنیم γ_i نماد تابع در بالاترین موقعیت β_i باشد؛ یعنی $\gamma_i = \beta_i|\varepsilon$. دو رابطهٔ ترتیب جزیی $<_1$ و $<_2$ را به‌ترتیب برای سیستم‌های FORWARD-TRS و TRS-BAKWARD روی مجموعه نمادهای F به کار رفته تعریف می‌کنیم:

$$\text{protocol} <_1 \text{match} <_1 . <_1 \gamma_{START} = \gamma'_1 <_1 \gamma''_1 = \gamma'_2 <_1 \ldots \gamma'_i <_1 \gamma''_i = \gamma'_{i+1} <_1 \ldots \gamma''_n = \gamma_{STOP} <_1 \text{st}$$

$$\text{bprotocol} <_2 \text{bmatch} <_2 . <_2 \gamma_{STOP} = \gamma''_n <_2 \gamma'_n = \gamma''_{n-1} <_2 \ldots \gamma'_{i+1} = \gamma''_i <_2 \gamma'_i = \ldots \gamma'_1 = \gamma_{START} <_2 \text{st}$$

یادآوری می‌کنیم که هر یک ازگام‌های پروتکل در قوانین بازنویسی فوق نهایتاً به عبارت st بازنویسی می‌شوند.

اگر دو رابطهٔ $<_{rpo1}$ و $<_{rpo2}$ به‌ترتیب روابط rpo [۲۰] متناظر با $<_1$ و $<_2$ روی مجموعه عبارت تشکیل دهندهٔ قوانین بازنویسی فوق باشند، در این صورت بر اساس تعریف rpo می‌توان دید:

$$\text{protocol} >_{rpo1} S_{INIT} . S_{START} . \text{match}(\alpha_{START} . \beta_{START})$$
$$\text{match}(\alpha'_i . \beta'_i) >_{rpo1} S_i . \text{match}(\alpha''_i . \beta''_i)$$
(for all i in which S_i is an intermediate step)
$$\text{match}(\alpha_{STOP}\, \beta_{STOP}) >_{rpo1} S_{STOP}$$

و همچنین:

$$\text{bprotocol} >_{rpo2} \text{bmatch}(\alpha_{STOP}\, \beta_{STOP}) . S_{STOP}$$
$$\text{bmatch}(\alpha''_i . \beta''_i) >_{rpo2} \text{bmatch}(\alpha'_i . \beta'_i) . S_i$$
(for all i in which S_i is an intermediate step)
$$\text{bmatch}(\alpha_{START} . \beta_{START}) >_{rpo2} S_{START} . S_{INIT}$$

و به این ترتیب می‌توان گفت که هر یک از دو مجموعه قوانین FORWARD-TRS و TRS-BAKWARD و در نتیجه ترکیب آنها پایان‌پذیر است.□

با توجه به قضیهٔ ۱، در تعریف گام‌های پروتکل باید دقت شود که پس‌شرط توانایی Send و پیش‌شرط توانایی Receive در ابتدا و انتهای هر گام میانی شامل مسندات وضعیتی باشند که به ترتیب فقط با یک Receive و Send دیگر در گام‌های دیگر منطبق شوند. روش مناسب برای این کار استفاده از دانش وضعیت است که در تعریف مدل محاسباتی نیز پیشنهاد شده است (به تعریف توانایی رجوع کنید).

قضیهٔ ۲: مجموعه قوانین FORWARD-PATH و BACKWARD-PATH ، همگرای محلی هستند.

اثبات: با توجه به تعریف زوج‌های بحرانی در سیستم بازنویسی، می‌توان دید که دو مجموعه قانون مورد بحث دارای هیچ زوج بحرانی نیست و بر این اساس بنا بر لم زوج‌های بحرانی[1] [۲۱]، قضیه ثابت می‌شود.□

قضیهٔ ۳: سیستم بازنویسی ارائه شده برای توصیف پروتکل همگرا است.

اثبات: اگر سیستم مورد بحث را R بنامیم، داریم:

$$R = \text{FORWARD-PATH} \cup \text{BACKWARD-PATH} \cup \text{PRIMITIVES} .$$

مجموعه قوانین PRIMITIVES ، متعامد است، زیرا اولاً تمامی قوانین آن left linear هستند (یعنی هر متغیر در سمت چپ فقط یک بار تکرار شده) و ثانیاً دارای هیچ زوج بحرانی نیستند. بنابراین مجموعه قوانین PRIMITIVES همگرا است [۱۸]. با توجه به اینکه قوانین سیستم‌های فوق از یکدیگر مجزا هستند به کمک دو قضیه قبل، قضیه ثابت می‌شود.□

۷- تحلیل پروتکل بر اساس مدل محاسباتی

توصیف پروتکل به کمک مجموعه‌ای از قواعد بازنویسی نهایتاً یک ترکیب نحوی از گام‌های درهم شده از نقش‌های پروتکل را فراهم می‌کند که آن را مسیر اجرای پروتکل نامیدیم. اجرای پروتکل روی این مسیر اجرا فرایندی معنایی

1. Critical Pair Lemma

است، به این معنا که اجرای پروتکل حاصل، انجام رشته‌ای از استنتاجات روی مسیر اجرای پروتکل برای بررسی اجراپذیر بودن هریک از توانایی‌های تشکیل دهندهٔ آن است. براثر اجرای پروتکل، یک مجموعه دانش دربارهٔ عوامل پروتکل و یک مجموعه دانش دربارهٔ نفوذی به دست می‌آید. مجموعه‌های $\mathcal{PIR}$ و $\mathcal{EIR}$ در گسترش این مجموعه‌های دانش دخالت دارند. وارسی وجود یا عدم وجود ویژگیهای امنیتی در پروتکل از طریق اثباتهایی دربارهٔ این دو مجموعه دانش انجام می‌شود.

به کمک مدل محاسباتی مطرح شده، می‌توان پروتکل را در سه وضعیت مختلف تحلیل کرد: پروتکل بدون وجود نفوذی، پروتکل با وجود نفوذی غیرفعال[1] و پروتکل با وجود نفوذی فعال[2]. در هر یک از این وضعیت‌ها به کمک تعریف اهداف تحلیل مناسب روی مسیر اجرای پروتکل و سعی در اثبات آنها، می‌توان ویژگیهای امنیتی را در پروتکل وارسی کرد یا وجود سناریوهای نقض امنیت را کشف نمود. در مراجع [۲۲] و[۲۳] نسخهٔ اولیه‌ای از روش انجام تحلیل‌های ایستا و پویای پروتکل بر اساس چارچوب تحلیل مورد بحث در این مقاله ارائه شده است. در ادامه به صورت غیر صوری به بیان مختصر چگونگی انجام این سه نوع تحلیل می‌پردازیم.

در تحلیل پروتکل بدون وجود نفوذی، برای پروتکل یک مجموعه دانش تعریف می‌شود. این مجموعه دانش حاوی تمامی‌مسندات صحیح راجع به عوامل پروتکل است و به زبان L بیان می‌شود. مقدار اولیهٔ این مجموعه، فرضیات اولیهٔ پروتکل است ودر طی گام‌های متوالی مسیر اجرا، با به فعلیت رسیدن هر توانایی و به کمک مجموعه قوانین استنتاج عوامل ($\mathcal{PIR}$)، این مجموعه به طور یکنوا[3] گسترش می‌یابد. تحلیل پروتکل در این وضعیت معادل رویکرد روش منطقی است و می‌توان به کمک آن به همان توانایی‌های روش تحلیل منطقی دست یافت.

در تحلیل پروتکل با نفوذی غیر فعال می‌توان ویژگی اختفا[4] در پروتکل را که ممکن است از طریق نفوذی غیرفعال نقض شود، وارسی کرد. نفوذی غیر فعال روی مسیر اجرای پروتکل فقط توانایی شنود دارد. نفوذی غیر فعال نیز مانند خود پروتکل، یک مجموعه دانش دارد. این مجموعه دانش با یک مقدار اولیه شروع می‌شود و با شنود پیام‌های پروتکل در روند اجرا و به کمک مجموعه قوانین استنتاج نفوذی ($\mathcal{EIR}$)، گسترش می‌یابد.

اگر در چارچوب ارائه شده، نفوذی را به صورت فعال در نظر بگیریم،می توان حملاتی که از طریق نفوذی فعال امکان پذیر است[۲٤]، مانند حملات تکرار[5]، انعکاس[6]، در هم بافی[7] و ... را تشخیص داد. این حملات عموماً به نقض ویژگی اصالت[8] موجودیت از طریق فریب عامل وارسی کنندهٔ اصالت منجر می‌شود. نفوذی فعال علاوه بر دارا بودن قابلیت‌های نفوذی فعال یعنی شنود پیام‌های پروتکل و گسترش مجموعه دانش خود به کمک قوانین استنتاجی $\mathcal{EIR}$، می‌تواند به صورت فعال نیز به پروتکل وارد شود و یک یا چند عامل مجاز پروتکل را فریب دهد. عامل وقتی فریب می‌خورد که عامل مجاز تصور کند ویژگی امنیتی رعایت شده، اما در عمل با فعالیت‌هایی که نفوذی انجام داده است، این ویژگی نقض شده باشد.

۸- جمع‌بندی

هدف از تحلیل پروتکلهای رمزنگاری، وارسی وجود یا عدم

1. Passive Intruder
2. Active Intruder
3. Monotonic

4. Secrecy
5. Replay attack
6. Reflection Attack
7. Interleaving Attack
8. Authentication

وجود ویژگیهای امنیتی در آنها است. دو رویکرد کلی در این زمینه، روشهای تحلیل ایستا و پویا است. در روش تحلیل ایستا، با مدلسازی منطق حاکم بر توابع رمزنگاری و روش‌های ساخت پروتکل، اندیشه‌های طراح پروتکل در قالب صوری دقیقی بازسازی می‌شوند و حصول ویژگیهای امنیتی در آنها به طور صوری اثبات می‌شود. در روشهای تحلیل پویا، به جای اثبات وجود ویژگیهای امنیتی در پروتکل، به دنبال یافتن سناریوهایی برای نقض ویژگیهای امنیتی هستیم.

در این مقاله چارچوبی ارائه شد که در آن با بهره‌گیری از ویژگیهای کلی دو رویکرد فوق، وارسی وجود ویژگیهای امنیتی در پروتکل را همراه با وارسی وجود سناریوهای نقض ویژگیهای امنیتی، به طور یکجا می‌توان انجام داد. این چارچوب را به عنوان دسته‌ای کلی از روشهای تحلیل می‌توان در نظر گرفت که به صورت $Analyze(\mathcal{L},\mathcal{M},\mathcal{PIR},\mathcal{EIR})$ نشان داده می‌شود و در آن $\mathcal{L}$ زبانی است که از طریق آن ویژگیهای عوامل مجاز و نفوذی در پروتکل بیان می‌شود، $\mathcal{M}$ مدلی محاسباتی از پروتکل ونحوهٔ اجرای آن است، $\mathcal{PIR}$ و $\mathcal{EIR}$ نیز به‌ترتیب مجموعه اصول برای انجام استنتاج‌های عوامل مجاز و نفوذی است. توصیف پروتکل در مدل محاسباتی $\mathcal{M}$ و اصول لازم در مجموعه‌های $\mathcal{PIR}$ و $\mathcal{EIR}$ به زبان $\mathcal{L}$ بیان می‌شود. زبان $\mathcal{L}$ در اینجا یک زبان مرتبه اول و شامل دو عنصر اصلی پیام و دانش در نظر گرفته شد.

در این مقاله مدلی محاسباتی از پروتکل برای به‌کارگیری در چارچوب مورد بحث ارائه شد. در این مدل محاسباتی، عوامل پروتکل به عنوان نقشهای عام با توانایی‌هایی مشخص در نظر گرفته شد. توانایی‌های بالقوه هر نقش دقیقاً براساس تعریف پروتکل به فعلیت می‌رسد و گامهای متوالی اجرای پروتکل را تشکیل می‌دهند. رشتهٔ خاصی از توانایی‌های نقش‌های مختلف پروتکل که طبق تعریف پروتکل باید به فعلیت برسند، مسیر اجرای پروتکل را تشکیل می‌دهد. روی این مسیر اجرا می‌توان در مورد دانش پروتکل (یعنی مجموعه‌ای از مسندات دربارهٔ باورها، مالکیتها یا سایر ویژگی‌های نقشهای مجاز) ، دانش نفوذی (مجموعه‌ای از مسندات مربوط به یافته‌های نفوذی از پروتکل) و همچنین قابلیت نفوذی در سوء استفاده از این مسیر اجرا برای فریب نقشهای مجاز به استنتاج پرداخت.

تفسیری اجرایی از مدل محاسباتی پروتکل بر اساس سیستم بازنویسی جبری ارائه شد. در این تفسیر اجرایی، پروتکل در قالب مجموعه‌ای از قوانین بازنویسی تعریف می‌شود. با اجرای این قوانین مسیر اجرای پروتکل را به دو صورت روبه‌جلو و روبه‌عقب می‌توان ایجاد کرد. برای توصیف پروتکل کافی است مجموعه گامهای پروتکل را به صورت منفرد بر اساس مدل محاسباتی ارائه شده بیان کرد. سپس بر اساس الگوریتم ارائه شده، فهرست قوانین بازنویسی به طور خودکار ایجاد می‌شود. در قضایایی، پایان پذیری و همگرا بودن سیستم بازنویسی ارائه شده نیز اثبات شده است.

در این مقاله، توصیف صوری پروتکل Woo-Lam به عنوان پروتکل نمونه تحت تفسیر اجرایی مذکور ارائه شد. کار در زمینه تعریف صوری چگونگی انجام تحلیل‌های امنیتی در این چارچوب و تحلیل پروتکل‌های امنیتی مختلف با آن ادامه دارد.

۹- منابع

[1] J., Clark; J., Jacob; "A Survey of Authentication Protocols Literature: Version 1.0"; http://www.cs.york.ac.uk/jac/papers/drareview.ps.gz; November 1997.

[2] P., Syverson; "The Use of Logic in the Analysis of Cryptographic Protocols"; Proc. of The Computer Security Foundations Workshop VII; 1994.

[3] S., Gritzalis; N., Nickitakos; P., Georgiadis; "Formal Methods for the Analysis and Design of Cryptographic Protocols: A State of the Art Review"; Communication and Multimedia Security; Vol. 3; S. Katsicas

(Ed.) IFIP; Chapman & Hall 1997.

[4] C., Meadows; "Formal Verification of Cryptographic Protocols: A Survey"; Technical Report; Center of High Assurance Computer Systems; Naval Research Laboratory; Washington DC; USA, 1995.

[5] R., Kemmerer; C., Meadows; J., Millen; "Three Systems for Cryptographic Protocol Analysis" ; Journal of CRYPTOLOGY; 7; pp. 79-130; 1994.

[6] M., Burrows; M., Abadi; R., Needham; "A Logic of Authentication"; Proc. Royal Society; Series A; Vol. 246; No. 1871; pp. 233-271; 1989.

[7] L., Gong; R., Needham; R., Yahalom; "Reasoning About Belief in Cryptographic Protocols"; Proc. IEEE 1990 symposium on Security and Privacy; Oakland, California; pp. 234-248; May 1990.

[8] P., Syverson; P., Van Oarschot; "On Unifying Some Cryptographic Protocol Logics"; IEEE Computer Society Symposium on research in Security and Privacy; 1994.

[9] D., Dolev; A.C., Yao; "On the Security of Public key Protocols"; IEEE Transactions on Information Theory; 22; 1976.

[10]C., Meadows; "Applying Formal Methods to the Analysis of a Key Management Protocol"; The Journal of Computer Security; Vol. 1; No. 1; Jan. 1992.

[11]R., Kemmerer; "Using Formal Verification Techniques to Analyze Encryption Protocols"; IEEE Computer Society Symposium on Security and Privacy; 1987.

[۱۲]ترک لادانی، بهروز؛ جلیلی، سعید؛ "چارچوبی برای وارسی پروتکل‌های رمزنگاری" ؛ مجموعه مقالات اولین کنفرانس رمز ایران؛ دانشگاه امام حسین(ع)؛ آبان ۱۳۸۰.

[13] C., Meadows; "Invariant Generation Techniques in Cryptographic Protocol Analysis"; Proceedings of the 13th Computer Security Foundations Workshop; IEEE Computer Society Press; July, 2000.

[۱٤]ترک لادانی، بهروز؛ میریان حسین آبادی، سیدحسن؛ "توصیف معنای پروتکل‌های رمزنگاری با استفاده از زبان توصیف Z" ؛ مجموعه مقالات ششمین کنفرانس انجمن کامپیوتر ایران؛ دانشگاه اصفهان؛ اسفند ۷۹.

[15]S., Brackin; "A HOL Extension Of GNY For Automatically Analyzing Cryptographic Protocols"; Proceedings of The 1996 IEEE Computer Security Foundations workshop IX; pp. 62-76; IEEE, 1996.

[16]W., McCune; Otter 3.04 Reference Manual and Guide; Tech. Report ANL-94/6, Argonne National Laboratory, Argonne, IL; 1995; http://www.mcs.anl.gov/AR/otter/.

[17]M., Newman; "On Theories with a Combinatorial Definition of "Equivalence""; Ann. Math; 43, pp. 223-243; 1942.

[18]B.K., Rosen; "Tree-manipulating Systems and Church-Rosser"; Journal of the ACM 20(1); pp. 160-; 187; 1973.

[19] T., Woo; S., Lam; "Authentication for Distributed Systems"; Computer; 25:10-10; March 1992.

[20]N., Dershowitz; "Orderings for Term Rewriting Systems"; Theoretical Computer Science 17, pp. 279-301; 1982.

[21]G., Huet; "Confluent Reductions: Abstract Properties and Applications to Term Rewriting Systems"; Journal of the ACM 27 (4); pp. 797-821; 1980.

[۲۲]ترک لادانی، بهروز؛ جلیلی، سعید؛ "وارسی پروتکل‌های رمزنگاری بدون فرض وجود صریح نفوذی"؛ مجموعه مقالات هفتمین کنفرانس انجمن کامپیوتر ایران؛ مرکز تحقیقات مخابرات؛ اسفند ۱۳۸۰.

[۲۳]ترک لادانی، بهروز؛ جلیلی، سعید؛ "ایجاد خودکار سناریوهای فریب نفوذی در پروتکل‌های رمزنگاری" ؛ گزارش فنی؛ گروه کامپیوتر دانشگاه تربیت مدرس؛ ۱۳۸۱.

[24] P., Ryan; S., Schneider; "Modeling and Analysis of Security Protocols"; Pearson Education, 2001..