دانشگاه صنعتی شریف

دانشکده‌ی علوم ریاضی

پایان‌نامه‌ی کارشناسی ارشد

گرایش علوم کامپیوتر

روش‌های صوری در درستی‌یابی پروتکل‌های رمزنگاری

نگارش: محمد ترابی دشتی

استاد راهنما: دکتر محمد اردشیر

آبان ۱۳۸۲

سپاسگزاری

از دکتر محمد اردشیر به خاطر راهنمایی‌های ارزشمندشان در طول انجام این رساله، و از دکتر امیر دانشگر و دکتر علی موقر که زحمت داوری این پایان‌نامه را بر عهده داشتند صمیمانه سپاسگزارم.

همچنین جا دارد از مرکز تحقیقات مخابرات ایران که از این پایان‌نامه تحت قرارداد شماره ۵۰۰/۵۰۵۷ مورخه‌ی ۱۳۸۲/۵/۱۹ حمایت کرده است قدردانی شود.

چکیده

پروتکل‌های رمزنگاری معمولاً ضامن امنیت در سیستم‌ها و شبکه‌های رایانه‌ای به شمار می‌آیند، حال آنکه حصول اطمینان از امنیت خودِ این پروتکل‌ها به لحاظ طبیعت واکنشیِ آن‌ها، چندان ساده نیست. به نظر می‌رسد روش‌های صوری ابزار مناسب برای درستی‌یابی وجه واکنشی از امنیت پروتکل‌ها هستند. در این پایان‌نامه، ابتدا محدودیت‌های نظری و محاسباتی مسأله‌ی درستی‌یابی صوری پروتکل‌های رمزنگاری را مطالعه کرده و در ادامه مروری بر رویکردهای موجود برای حل این مسأله خواهیم داشت. یکی از کاستی‌های روش‌های موجود، محدود کردن مفهوم امنیت به دو مشخصه‌ی *محرمانه بودن* و *تصدیق هویت* است در حالیکه امروزه حملات عدم سرویس‌دهی جزو رایج‌ترین حملات علیه پروتکل‌هاست. از این رو، ما یک ساختار مبتنی بر *مصرف منابع* برای توصیف و تعریف صوری حملات عدم سرویس‌دهی ارائه می‌کنیم. در انتها نیز حمله‌ی **SYN** علیه پروتکل **TCP/IP** بررسی شده و بر اساس مفهوم مصرف منابع، درستی روش دفاعی **SYN-Cookie** به طور صوری اثبات می‌شود.

کلید واژه‌ها:

روش‌های صوری (**Formal Methods**)، درستی‌یابی (**Verification**)، پروتکل‌های رمزنگاری (**Cryptographic Protocols**)، حمله‌ی **SYN** (**SYN Attack**)، سیستم‌های بلادرنگ (**Real-time Systems**).

فهرست مطالب

فصل ۱: پیش در آمد

رشد چشمگیر اینترنت و سادگی و مزایای تبادلات بر-خط[1]، نیاز به طراحی و پیاده‌سازی پروتکل‌های تجارت الکترونیکی را افزایش داده است. شاید بتوان گفت مهمترین شبهه در به کارگیری این پروتکل‌ها موضوع "امنیت" است. به طور سنتی، "رمزنگاری" ضامن امنیت در چنین ساختارهایی بوده و هست؛ از این رو اثبات امنیت پروتکل‌های رمزنگاری یک زمینه‌ی تحقیقاتی فعال به حساب می‌آید که شامل جنبه-ی محاسباتی امنیت الگوریتم‌های به کاررفته و امنیت خود پروتکل به عنوان یک سیستم واکنشی[2] است. درستی‌یابی پرتکل‌های رمزنگاری به عنوان سیستم‌های واکنشی موضوع این پژوهش می‌باشد که معمولاً توسط روش‌های صوری امکان‌پذیر است. در ادامه، اصول امنیت پروتکل‌های رمزنگاری و مبانی درستی-یابی صوری که بعد از این به کارخواهند آمد، معرفی شده و محدودیت‌های نظری حل مسأله‌ی "درستی‌یابی صوری پروتکل‌های رمزنگاری" بررسی خواهد شد. سپس روش‌های موجود به همراه بسته‌های نرم افزاری مربوط به آنها در قالب دو گروه "روش‌های منطقی" و "روش‌های جبری" ارائه

[1] Online
[2] Reactive System

می‌شوند. همانطور که ساختارهای منطقی و جبری کاملاً مجزا نیستند، این تقسیم‌بندی هم جامع و مانع نیست اما به دریافت ایده‌ها و مسائل ویژه‌ی هر دیدگاه کمک می کند.

روش‌های موجود عمدتاً امنیت را در قالب مشخصه‌های محرمانه‌بودن و تصدیق هویت مطالعه کرده و به راحتی قادر به بررسی امنیت در برابر حملات عدم سرویس‌دهی نیستند. در این پایان‌نامه ما یک ساختار مبتنی بر *مصرف منابع* برای توصیف صوری حملات عدم سرویس‌دهی ارائه می‌کنیم. در این راستا، به عنوان یک مطالعه‌ی موردی، حمله‌ی SYN علیه پروتکل TCP/IP را که از معروفتربن حملات عدم سرویس‌دهی به شمار می‌آید، وارسی کرده و درستی یکی از روش‌های دفاع در برابر آن را (که از ابزار رمزنگاری استفاده می‌کند) به طور صوری اثبات می‌کنیم.

در این پایان نامه تا حد امکان معادل انگلیسی کلمات به صورت پاورقی آمده و امید است که خواننده در ایجاد تناظر بین آنچه که از پیش می‌داند و آنچه که اینجا می‌یابد مشکلی نداشته باشد.

فصل ۲: مقدمات

۲-۱- رمزنگاری و پروتکل‌های رمزنگاری

منظور از یک "پروتکل رمزنگاری"، پروتکلی است که از الگوریتم‌های رمزنگاری استفاده می‌کند. معمولاً حملات به پروتکل‌های رمزنگاری به سه سطح "حمله به پیاده سازی"، "حمله به الگوریتم‌های رمزنگاری به کار رفته" و "حمله به خود پروتکل" تقسیم می‌شوند [۱۱٦]. در حقیقت بین امنیت یک پروتکل رمزنگاری و امنیت الگوریتم‌های رمزنگاری آن تفاوت وجود دارد و هدف اصلی ما بررسی امنیت پروتکل‌ها با فرض پیاده‌سازی صحیح و الگوریتم‌های رمزنگاری امن است. از دیدگاه روش‌های صوری[1] درستی‌یابی (امنیت) الگوریتم‌های رمزنگاری چندان جالب نیست، چراکه غالباً مفاهیم ریاضی و محاسباتی‌ای تولید کننده این الگوریتم‌ها می‌باشند که ازحیطه‌ی عملکرد روش‌های صوری خارج هستند. همچنین جزئیات پیاده سازی قابل بیان و بررسی با روش‌های صوری فعلی نیستند. اما خود پروتکل به عنوان یک سیستم واکنشی کاملاً با ابزار و مفاهیم روش‌های صوری سازگاری دارد و طبیعی می نماید که درستی‌یابی این بخش از "امنیت" توسط روش‌های صوری انجام گیرد. پیچیدگی حملات واقعی که

[1] Formal Methods

معمولاً چند سطح از سطوح ذکر شده را در برمیگیرند (مانند "حمله جعل IP[1]" که علاوه بر ضعف پروتکل TCP/IP، هسته‌ی تولید اعداد تصادفی سیستم‌های عامل را مورد حمله قرار می‌دهد [٤٥]) مهمترین مانع در استفاده‌ی وسیع از روش‌های صوری به شمار می‌رود، با این حال حملاتی وجود دارند که برای اولین بار توسط این روش‌ها پیدا شده و مشوّق تحقیقات جدی در این زمینه هستند [٧٩]. در ادامه چند حمله‌ی مشهور علیه پروتکل‌ها را به طور خلاصه بررسی خواهیم کرد که می‌توانند در روشن شدن تفاوت امنیت یک پروتکل رمزنگاری و الگوریتم‌های به کار رفته در آن کمک کنند.

الف- حمله‌ی بازتابی[2]: [١٣٠]

فرض کنید A و B از قبل یک کلید امن مشترک (مانند K_{AB}) به اشتراک گذاشته‌اند و می‌خواهند با استفاده از آن کلید از هویت یکدیگر مطمئن شوند:

$$A \rightarrow B: A, R_A$$
$$B \rightarrow A: R_B, K_{AB}(R_A)$$
$$A \rightarrow B: K_{AB}(R_B)$$

منظور از $A \rightarrow B: X$ این است که A پیام X را به B می‌فرستد و $K(X)$ نشان‌دهنده‌ی پیام X رمزشده با کلید K است. در قدم اول پروتکل فوق، A نام خودش و یک رشته بیت[3] تصادفی (مانند R_A) را برای B می‌فرستد. B هم این رشته بیت را با کلید مشترکشان (K_{AB}) رمز کرده و همراه یک رشته بیت تصادفی جدید برای A باز می‌فرستد. از آنجا که تنها A و B کلید مشترک (K_{AB}) را در اختیار دارند، در این مرحله A می‌تواند با دریافت رشته بیت رمزشده از هویت B مطمئن شود. به همین ترتیب بعد از قدم سوم، B هم از هویت A مطمئن می‌شود. هرچند که در نگاه اول این پروتکل امن به نظر می‌رسد، خواهیم دید که حتی اگر الگوریتم‌های رمز نگاری به کار رفته در آن امن بوده (مثلاً یک مهاجم[4] نتواند

[1] IP Spoofing
[2] Reflection Attack
[3] Bit
[4] Attacker (or Intruder)

بدون دسترسی به K_{AB} رشته بیت رمز شده را تولید کند) و پیاده سازی نیز صحیح باشد (مثلاً یک مهاجم نتواند اعداد تصادفی به کار رفته را از قبل حدس زده یا اعداد مورد نظر خودش را به پروتکل اعمال کند)، این تصور درست نبوده و حمله‌ی بازتابی به صورت زیر می‌تواند علیه این پروتکل به کار رود:

فرض کنید B عاملی[1] است که به طور همزمان به چند جلسه ارتباطی[2] جداگانه پاسخ می‌دهد (مانند یک عامل بانک) و مهاجم T دو جلسه ارتباطی همزمان را با B آغاز کرده و سعی می‌کند خودش را به جای A جا بزند[3]:

$$T \to B : A, R_T \; [session\,1]$$
$$B \to T : R_B, K_{AB}(R_T) \; [session\,1]$$
$$T \to B : A, R_B \; [session\,2]$$
$$B \to T : R'_B, K_{AB}(R_B) \; [session\,2]$$
$$T \to B : K_{AB}(R_B) \; [session\,1]$$

در این سناریوی حمله، T برای حل مشکل تولید $K_{AB}(R_B)$ از یک جلسه ارتباطی جدید استفاده می‌کند و B همه مراحل پرسش-پاسخ[4] را برای T طی می‌کند و در انتها T به عنوان A شناخته می‌شود. این مثال به وضوح تفاوت بین امنیت خود پروتکل و دیگر اجزای آن را نشان می‌دهد.

ب- حمله‌ی میانگیری[5]: [۱۳۰]

فرض کنید که A و B می‌خواهند با استفاده از پروتکل تبادل کلید Diffie-Hellman یک کلید مشترک تولید کنند:

$$A \to B : n, g, g^x \bmod n$$
$$B \to A : g^y \bmod n$$

[1] Agent
[2] Communication Session
[3] Impersonate
[4] Challenge-Response
[5] Man-In-The-Middle Attack

در این پروتکل n و g اعداد اول بسیار بزرگ با شرایط ویژه هستند (برای جزئیات به [۱۱۶] مراجعه کنید) که برای همه شناخته شده اند. در قدم اول، A یک عدد تصادفی مانند x را تولید کرده و باقیمانده تقسیم g^x بر n را محاسبه کرده و همراه با n و g برای B می‌فرستد. B هم در پاسخ یک عدد تصادفی مانند y را تولید کرده و باقیمانده تقسیم g^y بر n را برای A می‌فرستد. از آنجا که یافتن x با دانستن $g^x \bmod n$ به هیچ وجه ساده نیست و تا کنون هیچ الگوریتم کارآمدی به این منظور پیشنهاد نشده است (مسأله‌ی لگاریتم گسسته[1])، می‌توان فرض کرد که x و y برای عوامل دیگر پروتکل و عامل مهاجم ناشناخته باقی می‌ماند. در این مرحله A و B می‌توانند به ترتیب با محاسبه $(g^y \bmod n)^x = g^{x.y} \bmod n$ و $(g^x \bmod n)^y = g^{x.y} \bmod n$ به یک کلید مشترک دست یابند. اما حمله میانگیری می‌تواند به صورت زیر علیه این پروتکل به کار رود: (عامل مهاجم T، ارتباط بین A و B را قطع کرده و سپس خودش مسیر ارتباطی جدید می‌شود)

$$A \to T(B): n, g, g^x \bmod n$$
$$T(A) \to B: n, g, g^z \bmod n$$
$$T(B) \to A: g^z \bmod n$$
$$B \to T(A): g^y \bmod n$$

منظور از X(Y) این است که گیرنده یا فرستنده‌ی اصلی X است در حالی که طرف مقابل تصور می‌کند با Y تبادل داده می‌کند. همانطوریکه در توصیف فوق دیده می‌شود، T دو کلید متفاوت با A و B ایجاد کرده و بدون اینکه مشکلی در ارتباط آنها به وجود آید، به محتوای پیام‌ها پی می‌برد.

ج- حمله‌ی بازخوانی[2]: [۱۳۰]

یک مرکز توزیع کلید[3] (KDC)، می‌تواند به عنوان شخص ثالث قابل اعتماد[1] در پروتکل‌های رمزنگاری شرکت کند. KDC معمولاً کلیدهای محرمانه ای را با تک تک عوامل پروتکل به اشتراک دارد و برای

[1] Discrete Logarithm Problem
[2] Replay Attack
[3] Key Distribution Center (KDC)

تولید کلید مشترک جدید بین خود عوامل به کار گرفته می‌شود. برای مثال نسخه‌ی ساده شده‌ای از پروتکل "قورباغه–دهان–گشاد[۲]" به صورت زیر از KDC استفاده می‌کند:

$$A \rightarrow KDC : A, K_A(B, K_S)$$
$$KDC \rightarrow B : K_B(A, K_S)$$

در این پروتکل، K_A (K_B) کلید مشترک بین A (B) و KDC است. به این ترتیب هر کدام از عوامل با دیدن پیام رمز شده با این کلیدهای محرمانه به هویت طرف مقابل پی می‌برند. K_S هم کلید جلسه جدیدی است که در انتهای این پروتکل می‌تواند توسط A و B برای آغاز ارتباط به کار گرفته شود. سناریوی زیر می‌تواند نمونه ای از حمله بازخوانی علیه این پروتکل باشد: فرض کنید T کالای خاصی را برای عامل A تهیه کرده و می‌داند که A از عامل بانک B خواهد خواست که هزینه این کالا را به حساب T واریز کند. حال کافی است که T، پیام‌هایی را که بین KDC و B و سپس بین A و B رد و بدل می‌شود ذخیره کرده و در زمان‌های دیگر همان پیام‌ها را برای B بفرستد و هر دفعه همان مبلغ از حساب A به حساب T منتقل می‌شود.

پروتکل‌های زیادی برای تصدیق هویت (همراه با مرکز توزیع کلید) ارائه شده‌اند که مباحث فصول بعدی به برخی از آنها اشاره می‌کنند. خلاصه ای از آنها در اینجا می‌آید: [۱۱۶]

الف– پروتکل Needham-Schroeder:

$$A \rightarrow KDC : R_A, A, B$$
$$KDC \rightarrow A : K_A(R_A, B, K_S, K_B(A, K_S))$$
$$A \rightarrow B : K_B(A, K_S), K_S(R'_A)$$
$$B \rightarrow A : K_S(R'_A - 1), R_B$$
$$B \rightarrow A : K_S(R_B - 1)$$

ب– پروتکل Otway-Rees:

$$A \rightarrow KDC : I, A, B, K_A(R_A, I, A, B)$$

---

[1] Trusted Third Party
[2] Wide-Mouth-Frog Protocol

که I یک اندیس است.

$$B \to KDC : I, A, B, K_A(R_A, I, A, B), K_B(R_B, I, A, B)$$

B می‌تواند از طریق یک کانالِ نه لزوماً امن به I و $K_A(R_A, I, A, B)$ دسترسی داشته باشد.

$$KDC \to B : I, K_A(R_A, K_S), K_B(R_B, K_S)$$
$$B \to A : I, K_A(R_A, K_S)$$

ج- پروتکل کربروس[1]:

کربروس نتیجه رشد تدریجی پروتکل Needham-Schroeder است. در کربروس علاوه بر عوامل معمول A وB دو عامل "سرور تصدیق هویت[2]" (AS) و "سرور صدور بلیت[3]" (TGS) هم شرکت دارند که AS مانند KDC کلیدهای محرمانه ای را با عوامل A وB وTGS به اشتراک دارد. یک نسخه از این پروتکل به این صورت است:

$$A \to AS : A$$
$$AS \to A : K_A(K_S, K_{TGS}(A, K_S))$$
$$A \to TGS : K_{TGS}(A, K_S), B, K_S(t)$$
$$TGS \to A : K_S(B, K_{AB}), K_B(A, K_{AB})$$
$$A \to B : K_B(A, K_{AB}), K_{AB}(t)$$
$$B \to A : K_{AB}(t+1)$$

t یک مهر زمانی است که معمولاً برای جلوگیری از حمله بازخوانی به کار می‌رود.

عامل مهاجم، می‌تواند پیام‌ها را تغییر دهد، تبادل عادی داده را قطع کند، چندین کانال داده همزمان ایجاد کند، پیام‌ها را ذخیره کرده و در زمان‌های دیگر از آنها استفاده کند و .... در بخش‌های بعدی به طور دقیق توانایی‌های مهاجم را بررسی کرده و مدل ریاضی "محیط متخاصم[4]" را خواهیم دید.

[1] Kerberos Protocol
[2] Authentication Server (AS)
[3] Ticket Granting Server (TGS)
[4] Hostile Environment

۲-۲- مسأله‌ی درستی‌یابی صوری

هر کتابی که شامل پادکتابش[1] نباشد، ناتمام به شمار می‌آید.

خورخه لوییس بورخس

تلون، اُقبر، منظومه‌ی سوّم

بسیاری از سیستم های رایانه ای و کنترلی اگر آنطوری که باید، عمل نکنند مسبب فاجعه های انسانی یا زیست محیطی می‌شوند. حتی رقابت اقتصادی در بازار محصولات عادی هم باعث می‌شود که اشتباهات کوچک سازنده‌ها ضررهای مالی و اعتباری بزرگی برای آنها به دنبال داشته باشد (مثلاً اشتباه معروف FDIV در پردازنده‌های پنتیوم[2] حدود ۵۰۰ میلیون دلار به شرکت اینتل[3] خسارت زد [۳۵]). از این نظر حصول اطمینان از چگونگی رفتار سیستم‌ها قبل از به کارگیری آنها بسیار مهم است. روش های درستی-یابی صوری، ابزار لازم جهت مطالعه ارتباط بین آنچه یک سیستم انجام می‌دهد و آنچه باید انجام دهد را فراهم می‌آورند. به عبارت دیگر، با در نظر گرفتن اشیاء ریاضی زیر: [۱۰۸]

S- توصیف: آنچه که سیستم باید انجام دهد.

I- پیاده سازی: آنچه که سیستم انجام می‌دهد.

روش‌های صوری در نهایت می‌توانند صحت و سقم عبارت "S هم ارز I است" را اثبات کند. نکته حایز اهمیت اینجاست که روش‌های صوری در مورد مفاهیم مجرد S وI و ارتباط بین آنها صحبت می-کنند و نمی‌توانند بین آنچه طراح سیستم در ذهن دارد و آنچه در دنیای واقعی پیاده سازی می‌شود قضیه ای را بیان کنند.

یک ساختار درستی‌یابی صوری شامل سه بخش اساسی است: [۸]

[1] Counterbook (Contra book)
[2] Pentium
[3] Intel®

۱- یک زبان مدل سازی برای توصیف سیستم.

۲- یک زبان توصیف صوری برای "مشخصه" ای که باید درستیِ آن بررسی شود.

۳- یک الگوریتم جهت بررسی صحت مشخصه در سیستم توصیف شده.

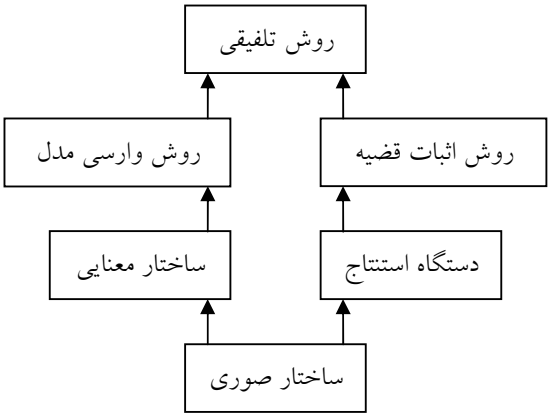
در چنین ساختاری، اینکه آیا سیستم M مشخصه φ را ارضاء می‌کند یا خیر، یک عبارت صوری می-باشد که در قالب‌های ریاضی قابل بررسی است [۸].

وارسی مدل[۱] و اثبات قضیه[۲] دو روش اصلی در درستی‌یابی صوری به کمک رایانه[۳] (یا خودکار[۴]) هستند. می‌توان ریشه این دو روش را در دیدگاه‌های گوناگون اثبات برای یک ساختار صوری[۵] معمول یافت. هر ساختار صوری دارای سه بخش (الف) ساختار نحوی[۶] (ب) ساختار معنایی[۷] و (ج) یک دستگاه استنتاج[۸] است [۱۳۳]. به این ترتیب، می‌توان مشخصات سیستم را در اصول اولیه دستگاه استنتاج گنجانده و از مسیر اثبات مبتنی براستنتاج ($\Gamma \vdash \varphi$) پیش رفت. از طرف دیگر می‌توان با استفاده از ساختار معنایی بررسی کرد که آیا سیستم تحت بررسی، یک مدل برای مشخصه مورد نظر می‌باشد یا خیر($M \models \varphi$) [۶۸]. توسیع این دو دیدگاه مستقیماً به روش‌های وارسی مدل و اثبات قضیه منتهی می‌شود. شکل ۱ مباحث فوق را به صورت خلاصه نمایش می‌دهد.

ارتباط بین دستگاه استنتاج و ساختار معنایی معمولاً طی دو قضیه "درستی[۹]" و "تمامیت[۱۰]" برقرار می-شود. در مورد هر ساختار صوری، قضیه صحت بررسی می‌کند آیا دستگاه استنتاج تنها قضیه‌هایی را اثبات می‌کند که از نظر ساختار معنایی درست هستند یا خیر؛ و قضیه تمامیت بیان می‌کند که آیا همه‌ی

[1] Model Checking
[2] Theorem Proving
[3] Computer Aided Verification
[4] Automatic
[5] Formal Structure
[6] Syntactic
[7] Semantic
[8] Deduction
[9] Soundness
[10] Completeness

چنین قضیه‌هایی (قضایای درست در ساختار معنایی) در دستگاه استنتاج قابل اثبات هستند یا خیر [۱۳۳].



شکل ۱- درستی‌یابی صوری و ساختار صوری

در حقیقت برخی از ساختارهای صوری (مانند حساب پئانو[1]، طبق قضیه عدم تمامیت گودل[2]) با یک ساختار معنایی ویژه، اصولاً نمی‌توانند یک دستگاه استنتاج "درست و تمام[3]" داشته باشند [۵۸].

در نگاه اول، روش "اثبات قضیه" در درستی‌یابی صوری، کاملاً طبیعی و مناسب به نظر می‌رسد چرا که اثبات کردن قضایا مبتنی بر فرضیات مشخص فرآیند متداول و جا افتاده‌ای در علوم تحلیلی[4] است. اما باید توجه داشت که اثبات قضایا به کمک رایانه قصه‌ی دیگری است و نیاز به خودکار بودنِ این عملیات است که پیچیدگی‌های خاصی را به دنبال دارد. حتی ساختارهای صوری نسبتاً ساده‌ای مانند منطق مرتبه اول[5] تصمیم پذیر[6] نیستند، به این معنی که هیچ الگوریتمی وجود ندارد که بتواند در مورد هر عبارت دلخواه در منطق مرتبه اول تصمیم بگیرد آیا آن عبارت یک "همیشه درست[7]" است یا خیر [۵۸]. در

[1] Peano Arithmetics
[2] Gödel's Incompleteness Theorem
[3] Sound and Complete
[4] Analytic Sciences
[5] First Order Logic
[6] Decidable
[7] Tautology

چنین شرایطی، یک سیستم خودکار (رایانه ای) در بهترین حالت می‌تواند "همیشه درست" بودنِ عبارات را در یک زیر ساختار مناسب بررسی کند [۵۸]. معمولاً در روش اثبات قضیه ما با مجموعه‌ای از فرضیات و اصول اولیه مانند Γ روبرو هستیم و باید برای درستی عبارتی مانند φ یک اثبات پیدا کنیم که تنها از قوانین استنتاج از پیش تعیین شده‌ای استفاده کند. در بسیاری از موارد عدم وجود الگوریتمی برای یافتن این اثبات، نقش رایانه را ضعیف کرده و نیاز به دخالت و راهنمایی عامل انسانی را افزایش می‌دهد در حالی که در فرآیند درستی‌یابی به کمک رایانه، خودکار بودن عملیات جزو اهداف اصلی به شمار می‌رود. به علاوه، بیان خواص یک سیستم در قالب فرضیات اولیه‌ای که مناسب برای اعمال قوانین استنتاج باشند، در بسیاری از موارد کار ساده‌ای نبوده و به عنوان یکی دیگر از مشکلات روش اثبات قضیه به شمار می‌آید [۱۸].

همان طور که Pnueli اشاره می‌کند، وارسی مدل اولین گام در مهندسی کردن[1] هنر درستی‌یابی منطقی (صوری) است [۱۰۸]. روش وارسی مدل قابلیت اجرای به صورت تمام خودکار را دارد امّا در عوض در مورد تمام مدل‌ها قابل اجرا نیست. در ابتدا، سیستم تحت مطالعه باید در قالب یک ساختار با حالت متناهی[2] بیان شود [٦٨]. فرضِ مدل با "حالات متناهی" دامنه‌ی کاربرد این روش را محدود می‌کند اما باز هم بسیاری از سیستم‌های واکنشی در آن مجموعه قرار می‌گیرند. در چنین شرایطی (سیستمی که اصولاً قابل بیان در یک مدل با حالات متناهی است)، ترجمه‌ی مشخصات سیستم به یک مدل با حالات متناهی کار ساده‌ای است [۱۸]. برای توصیف یک مشخصه در روش وارسی مدل، دو راه معمول وجود دارد: منطق زمانی[3] (وارسی مدل زمانی[4]) و خودکاره[5] با حالات متناهی (وارسی مدل مبتنی بر خودکاره[6]) [٦٨].

[1] Engineerization
[2] Finite State Structure
[3] Temporal Logic
[4] Temporal Model Checking
[5] Automata
[6] Automata-based Model Checking

در وارسی مدل زمانی، بررسی می‌شود که آیا یک عبارت مشخص در منطق زمانی (یا هر منطق مناسب دیگری که الگوریتم کارآمدی برای اثبات درستی در مدل مورد نظر داشته باشد) در یک (یا چند) حالت از مدل برقرار است یا خیر. در وارسی مدل مبتنی بر خودکاره، مشخصه‌ی مورد بررسی توسط یک خودکاره با حالات متناهی بیان می‌شود و پرسش این است که آیا این خودکاره زبان مدل متناظر با سیستم را می‌پذیرد یا خیر. یکی از مشکلات روش وارسی مدل، پدیده‌ی انفجار حالت[1] است به این معنی که تعداد حالات مدل با رشد متغیرهایی که برای توصیف سیستم به کار می‌روند به صورت نمایی رشد می‌کند [۸]. برای حل پیچیدگیِ محاسباتی ناشی از این پدیده روش‌های گوناگونی (از جمله روش وارسی مدل نمادین[2]) پیشنهاد شده اند و در نرم افزارهای مربوطه به کار گرفته می‌شوند [۳۵] [٦٨].

در مقام مقایسه، بیان مشخصات یک سیستم در قالب مدل با حالات متناهی معمولاً ساده‌تر از بیان آن در قالب فرضیات دستگاه استنتاج است، البته در صورتی که سیستم مورد نظر اصولاً متناهی حالت داشته باشد. اما متناهی بودن حداقل یک الگوریتم وحشیانه[3] برای وارسی مدل را به همراه دارد که ممکن است با روش‌های بهینه‌سازی مختلف به صورت مؤثری اجرا شود. به این ترتیب در روش وارسی مدل (که به طور ضمنی متناهی بودن را در بر دارد) از وجود یک الگوریتم اثبات درستی مطمئن هستیم، چیزی که در روش اثبات قضیه معمولاً وجود ندارد. از این زاویه بهتر مشخص می‌شود که چرا روش اثبات قضیه به صورت بنیادی پیچیده‌تر است. در حقیقت قضیه‌ی تصمیم ناپذیریِ درستی، مستقل از روش آن (وارسی مدل یا اثبات قضیه) برقرار است. در روش وارسی مدل معمولاً یک مدل متناهی از پیش فرض می‌شود در حالیکه روش اثبات قضیه در حالت کلی معادل وارسی مدل برای همه‌ی مدل‌های ممکن است. از این

[1] State Explosion
[2] Symbolic Model Checking
[3] Brute-force

روست که روش اثبات قضیه مستقیماً با نتایج قضیه‌ی تصمیم ناپذیریِ درستی در ساختار صوری مربوطه روبرو می‌شود.

با این حال، ایده‌هایی برای تلفیق این دو روش وجود دارند که علاوه بر حفظ جنبه‌های خودکارِ درستی-یابی، قابلیت کارکردن با سیستم‌های با حالات نامتناهی را داشته یا به صورت کارآتری سیستم‌های با حالات متناهی را درستی‌یابی کنند [۱۸]. یکی از این روش‌ها در اصل برای درستی‌یابی پروتکل‌های رمزنگاری توسعه یافته است که در بخش‌های بعدی مختصراً تحت عنوان "روش تولید نظریه[1]" خواهد آمد [۷۳].

[1] Theory Generation Method

فصل ۳: جنبه‌های محاسباتی

۳-۱- تصمیم (نا)پذیری

امنیت پروتکل‌ها در حالت کلی تصمیم‌پذیر نیست. قضیه‌ی زیر PCP (مسأله‌ی تناظر Post [۱۱۹][1]: مجموعه‌ای از جفت کلمات $\{(u_i,v_i)\,|\,i=1...N\}$ از یک الفبای مشخص داده شده‌اند، آیا دنباله‌ای از اندیس‌های $i_1,i_2,...,i_N$ وجود دارد که طبق آن $u_{i_1}u_{i_2}...u_{i_N}=v_{i_1}v_{i_2}...v_{i_N}$ برقرار باشد؟) را به محرمانه-بودن یک پروتکل به عنوان یک خاصیت امنیتی تبدیل کرده و یک اثبات صوری برای تصمیم ناپذیری امنیت پروتکل ها به شمار می‌آید.

قضیه ۱. [٦٤] مجموعه‌ی پروتکل‌های امن بازگشتی[2] نیست.

هدف یافتن یک کاهش[3] از مسأله‌ی تصمیم ناپذیر PCP به مسأله‌ی امنیت پروتکل‌ها است. پروتکل زیر را در نظر بگیرید:

(۱) شروع کننده $K(\varepsilon,\varepsilon)$ را بفرستد.

[1] Post Correspondence Problem
[2] Recursive
[3] Reduction

(۲) صبر کنید تا $K(X,Y)$ را دریافت کنید.

(۳) اگر $X = Y \neq \varepsilon$ و $|X| = \sum_{i \in I} |u_i|$ آنگاه K را بفرستید، اگر نه $K(X.u_i, Y.v_i)$ را برای یک $i \in I$ی تصادفی تولید کرده و بفرستید.

(٤) به مرحله‌ی ۲ بروید.

در این پروتکل ε رشته‌ی تهی، $|X| = N$ نشان‌دهنده‌ی طول رشته‌ی X و a.b نماینده‌ی رشته‌ی حاصل از چسباندن رشته‌های a و b است. یک الگوریتم رمزنگاری متقارن دلخواه را در نظر بگیرید که از کلید K استفاده می‌کند، اینکه آیا K افشاء خواهد شد یا خیر معادل با حل مسأله‌ی PCP برای مجموعه رشته‌های $U = \{(u_i, v_i) \mid i \in I\}$ است که می‌دانیم تصمیم ناپذیر است.

طرح شهودی زیر به روشن شدن ریشه‌ی تصمیم ناپذیریِ امنیت پروتکل‌ها کمک می‌کند:

(۱) هر ماشین تورینگ[1] یک پروتکل است. برای هر حالت از ماشین تورینگ یک عامل در پروتکل در نظر بگیرید که محتویات نوار را به عامل مناسب (عامل متناظر با حالت بعدی در ماشین تورینگ) می‌فرستد [٤۰].

(۲) امن بودن یک خاصیت بدیهی[2] برای ماشین‌های تورینگ نیست. یعنی همه‌ی ماشین‌های تورینگ امن نیستند یا همه‌ی ماشین‌های تورینگ ناامن نیستند. بنابر قضیه‌ی Rice ([۱۱۹]) خاصیت امن بودن (هر خاصیت غیر بدیهی) برای مجموعه‌ی ماشین‌های تورینگ تصمیم ناپذیر است.

(۳) امنیت برای یک زیر مجموعه از پروتکل‌ها تصمیم ناپذیر است، پس امنیت پروتکل‌ها در حالت کلی تصمیم ناپذیر است.

[1] Turing Machine
[2] Trivial

قضیه ۲. مجموعه‌ی پروتکل‌های امن بازگشتی شمارا[1] نیست.

قضیه‌ی قبل نشان می‌دهد که مجموعه‌ی پروتکل‌های امن بازگشتی نیست. اما می‌توان دید که مجموعه‌ی پروتکل‌های ناامن بازگشتی‌شمارا است. یک ماشین تورینگ در نظر بگیرید که به عنوان ورودی توصیف یک پروتکل را گرفته و به صورت عمق–اول[2] بررسی می‌کند که آیا در رشته‌های اجرای پروتکل، امنیت (طی یک تعریف ویژه) نقض می‌شود یا نه. به این ترتیب اگر پروتکل ناامن باشد در متناهی قدم این موضوع روشن می‌شود اما اگر پروتکل امن باشد ممکن است تا ابد ادامه بدهد. پس "نا امن" بودن قابل تشخیص[3] بوده و مجموعه‌ی پروتکل‌های ناامن بازگشتی شمارا است. حال فرض کنید که مجموعه‌ی پروتکل‌های امن هم بازگشتی شمارا باشد و ماشین تورینگ زیر را در نظر بگیرید:

ماشین تورینگ TM، دو ماشین تورینگ بازشناس[4] برای مجموعه‌ی پروتکل‌های امن و مجموعه‌ی پروتکل‌های نا امن را به صورت موازی شبیه‌سازی می‌کند، در زمان متناهی یکی از آن‌ها به جواب yes خواهند رسید (هر پروتکل یا امن است یا نا امن). اگر بازشناسِ مجموعه‌ی پروتکل‌های امن به جواب yes رسید، TM هم جواب yes می‌دهد و اگر بازشناس مجموعه‌ی پروتکل‌های نا امن به جواب yes رسید، TM جواب No می‌دهد. پس TM مجموعه‌ی پروتکل‌های امن را تصمیم‌گیری می‌کند و مجموعه‌ی پروتکل‌های امن بازگشتی است. از آنجا که این نتیجه در تناقض با قضیه‌ی قبل است، فرض وجودِ بازشناس برای مجموعه‌ی پروتکل‌های امن نادرست بوده و مجموعه‌ی پروتکل‌های امن بازگشتی شمارا نیست.

[1] Recursively Enumerable
[2] Breadth First
[3] Turing Recognizable
[4] Recognizer Turing Machine

۳-۲- مدل صوری مهاجم

بسیاری از نتایج و الگوریتم‌ها در مبحث درستی‌یابی صوری پروتکل‌های رمزنگاری بر اساس یک مدل صوری بنا شده‌اند که نشان دهنده‌ی توانایی‌های عامل مهاجم است. شاید بتوان گفت که بخش بزرگی از پیچیدگی‌های محاسباتی که در درستی‌یابی چنین پروتکل‌هایی وجود دارد، قدرت عمل بسیار زیاد عامل مهاجم است که البته در بسیاری از موارد هنوز ضعیف‌تر از یک مهاجم واقعی است. سه دیدگاه اصلی برای مدل سازی صوری مهاجم وجود دارند:

(۱) مدل Dolev-Yao (Dolev و Yao اولین افرادی بودند که ایده‌ی استفاده از روش‌های صوری در درستی‌یابی پروتکل‌های رمزنگاری را مطرح کردند [۹۲].)

(۲) مدل حساب[1] Spi

(۳) مدل احتمالاتی

مدل حساب Spi را در فصل‌های بعد خواهیم دید. مدل احتمالاتی هم ارتباط نزدیکی با مدل حساب Spi داشته و به علاوه جنبه‌های محاسباتی مسأله را هم در نظر می‌گیرد [۹۶]. این مدل کاربرد زیادی نداشته و هنوز یک مدل تحقیقاتی به شمار می‌آید. تمرکز این بخش بر مدل Dolev-Yao است که مهمترین و پر کاربرد ترینِ این مدل‌هاست. هرچند که یک رویکرد مبتنی بر منطق دانش[2] برای مدل کردن مهاجم بسیار قدرتمند بوده و مدل احتمالاتی و Dolev-Yao را در بر می‌گیرد [۶۳].

مدل Dolev-Yao فرض می‌کند: [۶۵]

بازیگران عادی پروتکل قوانین را به درستی اجرا می‌کنند در حالی که عامل مهاجم ممکن است تلاش کند که از پروتکل پیروی نکند. این گونه اندیشیده می‌شود که شبکه تحت کنترل کامل عامل مهاجم بوده

[1] Spi-calculus
[2] Logic of Knowledge

و او می‌تواند تمام پیام‌ها را ذخیره کرده، نابود کرده، بازخوانی کرده، بازمسیردهی[1] کرده و بازترتیب دهی[2] کند. همچنین او می‌تواند پیام‌های جدید نیز بیافریند. این طور تصور نمی‌شود که عامل مهاجم یک هدف خاص را دنبال می‌کند، بلکه یک منبع تصادفی ایجاد مزاحمت[3] به حساب می‌آید. مهاجم یک پروسه‌ی نامعین[4] است که در هر زمان ممکن است هر پیامی را که بتواند تولید کند و برای بازیگران دیگر پروتکل بفرستد. هدف اصلی در چنین مدلی، اثبات یک خاصیت امنیتی (نظیر محرمانه بودن یا تصدیق هویت) برای بازیگران عادی پروتکل با وجود یک عامل مهاجم است. توانایی‌های مهاجم در آفرینش پیام‌های جدید به صورت زیر محدود می‌شود:

- مهاجم همه‌ی مقادیر قابل پیش بینی و عمومی مانند نام‌ها، کلیدهای همگانی[5] و مُهرهای زمانی را می‌داند.
- او می‌تواند مقادیر تازه نظیر کلیدهای نشست[6]، نام‌ها و مقادیر نَمان[7] (می‌گوییم مقادیرِ نمان مانند رازِ مگو) را بیافریند.
- مهاجم دارای نام است و بازیگران عادی پروتکل می‌توانند با او ارتباط برقرار کنند و در هر زمانی که بخواهد می‌تواند نام‌های جدیدی برای خودش تولید کند.
- مهاجم مجموعه‌ای از کلیدهای خصوصی[8] یا عمومی را که به عنوان یک بازیگر عادی به آنها دسترسی دارد، می‌داند.
- مهاجم می‌تواند مقادیری را که از آنها اطلاع دارد با کلیدهایی که می‌داند، رمز کند.
- مهاجم می‌تواند پیام‌ها را رمز گشایی کند تنها در صورتی که کلید آنها را داشته باشد.

[1] Reroute
[2] Reorder
[3] Noise
[4] Nondeterministic
[5] Public Keys
[6] Session Keys
[7] Nonce Values
[8] Private Keys

- مهاجم می‌تواند دو پیام را به هم چسبانده یا یک پیام مرکب را به اجزای آن قسمت کند.

در این مدل فرض می‌شود که هر مقدار[1] تنها یک نمایش[2] دارد (خاصیت جبر آزاد[3]) [٦٥]. برخی پژوهشگران تلاش کرده‌اند با حذف این فرض مدل را قدرتمندتر کنند [٦١]. ایده این است که رابطه‌ای بین مدل صوری رمزنگاری و مدل محاسباتی آن پیدا کنند، دیدگاهی که بعد از مقاله‌ی معروفی از Abadi و Rogaway مرسوم شد [٦٥][١]. از طرف دیگر با استفاده از ایده‌های بازنویسیِ چند مجموعه‌ای[4] در مدل کردن پروتکل‌های رمزنگاری، قضیه[5]ی مهمی اثبات شده است. بازنویسی چند مجموعه‌ای به همراه سورهای وجودی[6] برای بیان مقادیر نمان، قادر است پروتکل‌های رمزنگاری را مدل کند [٣٢]. Cervesato از این ابزار استفاده کرده و نشان داد که ***مدل Dolev-Yao برای مهاجم تواناترین مدل مهاجم است*** [٣٣]. در حقیقت او اثبات کرد که هر مهاجمی که در سیستم بازنویسی چند مجموعه‌ای قابل تصور است توسط مهاجم Dolev-Yao شبیه سازی می‌شود (پیوست ١، جزئیات بیشتری از این قضیه را می‌آورد) [٣٣]. همچنین ارتباط بین مدل Dolev-Yao با برخی ساختارهای صوری دیگر مانند حساب سیستم‌های مخابراتی[7] و منطق جریان[8] در [٨٣] و [٢١] مطالعه شده است.

٣-٣- محدودیت‌های محاسباتی

می‌توان برای واقعی‌تر کردن مدل، پروتکل‌های با تعداد کراندار نشست‌ها و حجم محدود دانش عامل مهاجم را در نظر گرفت. مهمترین نتایج این دیدگاه توسط Durgin به دست آمده‌اند. وی درپایان نامه‌ی

[1] Value
[2] Representation
[3] Free Algebra Property
[4] Multi Set Rewriting
[5] Meta Theorem
[6] Existential Quantifiers
[7] Calculus of Communication Systems (CCS)
[8] Flow Logic

دکترایش بازنویسی چند مجموعه‌ای را توسعه داده و قضیه‌های بسیاری در مورد کران‌های محاسباتی درستی‌یابی پروتکل‌های امنیتی اثبات نمود [۵۲]. مهمترینِ این نتایج عبارتند از: [۵۲]

- اگر اندازه‌ی پیام‌ها و عمق رمزنگاری پیام‌ها محدود باشند، برای حالتی که تعداد نامحدودی بازیگر در پروتکل شرکت دارند و تعداد ناکرانداری از نمان‌های جدید می‌توانند تولید شوند مسأله‌ی محرمانه بودن تصمیم ناپذیر است.
- اگر تعداد نمان‌ها را هم محدود کنیم، مسأله‌ی محرمانه بودن عضو مجموعه‌ی DEXPTIME-تمام[1] می‌شود.
- اگر علاوه بر تعداد نمان‌ها، تعداد بازیگران هم محدود شوند مسأله‌ی محرمانه بودن عضو مجموعه‌ی NP-تمام[2] می‌شود.

از طرف دیگر، وارسی مدل یک سیستم با حالات نامتناهی (که نتیجه فوری مدل مهاجم Dolev-Yao است) محدودیت‌های محاسباتی‌ای به همراه دارد که عمدتاً توسط محققان فرانسوی بررسی شده است [۴۰]. مهمترین نتیجه‌ی این دیدگاه بیان می‌کند که مسأله‌ی تحلیل دانش نمادین[3] مدل Dolev-Yao با تعداد کراندار نشست‌ها عضو مجموعه‌ی NP-تمام است (پیوست ۲ طرح اثبات این قضیه است) [۱۱۳]. به علاوه [۴۱] نشان می‌دهد که اگر حمله‌ای با وجود n بازیگر وجود داشته باشد، همواره حمله‌ای (نه لزوماً مشابه) با حضور تنها تعداد کرانداری b (معمولاً ۲) بازیگر هم وجود دارد. بنابراین اگر فرض شود که تنها b عامل درگیر پروتکل هستند و اثبات شود که هیچ حمله‌ای محتمل نیست، امنیت پروتکل در حالت کلی اثبات شده است.

[1] DEXPTIME-Complete
[2] NP-Complete
[3] Symbolic Knowledge Analysis

٤- روش‌های منطقی

٤-١- منطق BAN و توسعه‌های آن

شاید بتوان گفت که مطالعه‌ی جدی درستی‌یابی صوری پروتکل‌های رمزنگاری با مقاله‌ای توسط Abadi، Burrows و Needham آغاز شد. این مقاله یک *منطق* برای توصیف و درستی‌یابی پروتکل‌های رمزنگاری ارائه داد که به منطق BAN (ابتدای نام نویسنده‌های مقاله) مشهور شد [۲۹]. در حقیقت این منطق دارای یک ساختار نحوی و دستگاه استنتاجِ همراه آن بوده که با اندیشه‌ی معناییِ مناسب برای پروتکل‌های رمزنگاری طراحی شده است. به هر حال مقاله‌ی اصلی هیچ ساختار صوری معنایی‌ای ارائه نکرده است [۲۹]. بنابراین اثباتی در مورد تمامیت یا صحت دستگاه استنتاج آن هم وجود ندارد. BAN برای کارکردن با پروتکل‌های رمزنگاری نمادهای جدید و قوانین استنتاج آنها را پیشنهاد کرده که خلاصه‌ای از آن در ادامه می‌آید:

(۱) $P \mid\equiv X$: عامل P پیام X را باور دارد (BAN یک منطق باور است).

(۲) $P \lhd X$: عامل P پیام X را می‌بیند؛ به عنوان مثال روی شبکه X را دریافت می‌کند.

(۳) $P \mid\sim X$: عامل P یک بار (مشخص نیست در چه زمانی) پیام X را فرستاده است.

(٤) $P \Rightarrow X$: عامل P صاحب پیام X است؛ برای مثال X کلید عمومی P است.

(٥) $\#(X)$: پیام X تازه است؛ مثلاً X یک مقدار نمان است.

(٦) $P \xleftrightarrow{K} Q$: عامل‌های P و Q کلید K را به اشتراک دارند.

(۷) $\xrightarrow{K} P$: عامل P کلید عمومی K را در اختیار دیگران گذاشته است.

(۸) $P \overset{X}{\Longleftrightarrow} Q$: عامل‌های P و Q پیام محرمانه‌ی X را به اشتراک دارند.

(۹) $\{X\}_K$: پیام X توسط کلید K رمز شده است.

(۱۰) $(X)_Y$: پیام X با پیام Y ادغام شده است؛ مثلاً توسط پیام Y امضاء شده است.

همانطور که در نحوِ منطق BAN دیده می‌شود زمان مدل نمی‌شود، تنها "بعد" و "قبل" معنی دارند. از این رو اصولاً BAN نمی‌تواند مشکلات همزمانی[1] یا جابجایی[2] قدم‌های پروتکل را توصیف کرده و تشخیص دهد. اما تمهیدات لازم برای کارکردن با رمزنگاری نامتقارن اندیشیده شده است و مقاله‌ی اصلی مثالی از آنها را هم شامل می‌شود [۲۹]. در زیر چند نمونه از قانون‌های استنتاج BAN را می‌بینیم: [۲۹]

$$\frac{P \mid\equiv Q \overset{Y}{\Longleftrightarrow} P, P < (X)_Y}{P \mid\equiv Q \mid\sim X} \quad (۱)$$

این قانون بیان ریاضی این مطلب است که اگر عامل P باور داشته باشد که پیام محرمانه‌ی Y را با عامل Q به اشتراک دارد و پیام X را در حالی که با Y امضاء شده دریافت کند، به این باور خواهد رسید که عامل Q زمانی پیام X را فرستاده است.

[1] Concurrency
[2] Permutation

$$\frac{P \mid\equiv \#(X), P \mid\equiv Q \mid\sim X}{P \mid\equiv Q \mid\equiv X} \quad (۲)$$

اگر عامل P باور داشته باشد که پیام X تازه است و بداند که زمانی عامل Q آنرا فرستاده، به این باور می‌رسد که عامل Q هم پیام X را در همین زمان باور دارد.

هدف این است که در نهایت قضیه‌ای در مورد سطح باور طرفین پروتکل به یک پیام یا کلید مشترک (برای کاربردهای تصدیق هویت) اثبات شود. معمولاً قضایای زیر مورد توجه قرار می‌گیرند:

$\begin{cases} A \mid\equiv A \overset{K}{\Leftrightarrow} B \\ B \mid\equiv A \overset{K}{\Leftrightarrow} B \end{cases}$ یعنی عامل‌های A و B به این باور برسند که کلید یا پیام K را به اشتراک دارند.

و یا قضیه‌ی زیر:

$\begin{cases} A \mid\equiv B \mid\equiv A \overset{K}{\Leftrightarrow} B \\ B \mid\equiv A \mid\equiv A \overset{K}{\Leftrightarrow} B \end{cases}$ یعنی طرفین در مورد سطح باور طرف مقابل هم به باور برسند.

این قانون‌ها از یک پیش زمینه‌ی فکری در مورد پروتکل‌های تصدیق هویت نشأت گرفته اند. اما نداشتن ساختار صوری معنایی منجر به این می‌شود که کاربرانی که با این قانون‌ها مواجه می‌شوند آنها را مطابق با ادراک خودشان به کار برند. به این ترتیب Nessett مثالی ارائه کرد که نشان دهد که چطور می‌توان با استفاده از BAN ثابت کرد که یک پروتکل نا امن، امن است [۱۰۲]. پس BAN یک ساختار درست[1] نیست (اینکه یک پروتکل نا امن است در لایه‌ی فرازبانیِ[2] BAN بدست می‌آید چرا که اصولاً BAN دارای ساختار معنایی برای تعریف امنیت نیست). سپس Snekkenes نشان داد که مثال Nessett حالت ویژه‌ای است که تفاوت درک کاربر از قوانین استنتاج BAN را آشکار می‌کند [۱۲۰]. به همین ترتیب Monniaux اثبات کرد که BAN تصمیم پذیر است (اینجا هم چون BAN ساختار معنایی ندارد، اثبات تصمیم پذیری بی معنا می‌نماید، آنچه Monniaux نشان داده این است که همواره یک

[1] Sound
[2] Metalanguage

الگوریتم عقب‌گرد[1] برای یافتن اثبات یک نتیجه در دستگاه استنتاج BAN وجود دارد) [۹۹]. ابزار نرم افزاری گوناگونی برای تولید خودکار اثبات برای قضایای BAN آماده شده و یا اختصاصاً نوشته شده اند که از جمله آنها می‌توان به EVES ، Jape و SPEAR اشاره کرد [٤٣] [۱۱] [۱۱٤].

مشکلات منطق BAN و جذاب بودن ایده‌ی آن باعث شد که کارهای پژوهشی بسیاری در پیروی از آن انجام شوند. یکی از مشهورترین این تلاش‌ها، منطقی بود که Gong، Needham و Yahalom ارائه کردند؛ این منطق قانون‌های استنتاج BAN را توسعه می‌داد و به منطق GNY معروف شد [٥٦]. ابزار نرم افزاری گوناگونی مانند [۸۹، ۲۷، ۷۸ و۱۱٥] برای خودکارسازیِ GNY معرفی شدند؛ همچنین [۹۰] نشان داد که این منطق هم درست نیست. Abadi و Tuttle به منظور رفع مشکلات BAN یک ساختار معنایی به آن افزودند که به منطق AT مشهور شد [۲]. Van Oorschot هم BAN را برای کارکردن با پروتکل‌های موافقت کلید[2] توسعه داد که به منطق VO مشهور شد [۱۳٤]. پراکندگی این کارها باعث شد که Syverson و VanOorschot منطق‌های BAN، GNY، AT و VO را تحت نام منطق SVO جمع کرده و یک ساختار معنایی برای آن ارائه دادند و نشان دادند که ساختار استنتاج SVO درست است [۱۲٤]. یک ابزار نرم افزاری هم بر اساس SVO توسعه یافته است [٥۰].

یکی از منطق‌هایی که بر اساس BAN بنا شده و در درستی‌یابی پروتکل‌های تجارت الکترونیک (SET و Payword) به کار رفت AUTLOG است [۷۱][۷۲]. این منطق قوانین استنتاج جدیدی به BAN افزوده که (طی یک ساختار معنایی در لایه‌ی فرازبانی) همخوانی این قوانین با BAN اثبات شده است [۷۱].

همچنین Mao و Boyd منطق BAN به همراه ایده آل سازی صوری پروتکل را توسعه داده و بخشی از مشکلات BAN ناشی از نداشتی ساختار معنایی (حمله‌ی Nessett) را حل کردند [۸٤][۸٥].

[1] Backward
[2] Key Agreement Protocol

تلاش‌هایی هم برای افزودن مفهوم زمان به BAN انجام گرفت. از جمله Benerecetti و بقیه منطق BAN را با یک منطق زمانی تلفیق کرده و الگوریتم وارسی مدل مربوط به آن را هم ارائه کرده‌اند [۱۶]. همچنین Syverson عملگرهای زمانی را به BAN افزوده و ساختار معنایی آن را هم طرح کرد [۱۲۵]. یک منطق دیگر که بر اساس BAN توسعه یافت RV است [۷۳]. RV از این نظر اهمیت دارد که با یک دستگاه اثبات خودکار می‌آید. ایده این است که یک نمایش متناهی از یک نظریه (نه لزوماً متناهی) در اختیار داشته باشیم که بتواند در تصمیم گیری عضویت یک گزاره در آن به ما کمک کند و خودش هم به صورت کارآیی قابل محاسبه باشد. این نمایش متناهی "نظریه‌ی متناهی کارآ[1]" نامیده می‌شود [۷۳]. "اشباع نظریه[2]" یکی از روش‌های تولید یک نظریه‌ی متناهی کارآ است که ابتدا برای منطق مرتبه اول پیشنهاد شد [۵۵]. همانطور که از تصمیم ناپذیری منطق مرتبه اول انتظار داریم، روش اشباع نظریه هیچ اطمینانی در مورد پایان پذیری[3] الگوریتم تولید نظریه‌ی متناهی کارآ برای منطق مرتبه اول ارائه نمی-دهد [۵۵]. الگوریتمی که Kindred در رساله‌ی دکترایش ارائه می‌دهد تولید نظریه[4] نام دارد و در مورد گروه خاصی از نظریه‌ها به کار می‌رود [۷۳]. تحت شرایط خاصی برای نظریه‌ی اصلی، این امکان وجود دارد که نظریه‌ی متناهی کارآیی متناظر با آن تولید شود؛ به طور بسیار مختصر باید یک زیر مجموعه‌ی ویژه از قوانین استنتاج انتخاب شده و تنها گزاره هایی در نظریه‌ی متناهی کارآ قرار داده شوند که یک درخت استنتاج[5] منتهی به یکی از قانون‌های آن زیر مجموعه داشته باشد [۷۳]. معمولاً قانون‌هایی در این مجموعه‌ی ویژه قرار می‌گیرند که اندازه-کاه[6] باشند [۷۳]. ادعای مهم Kindred این است که BAN و

[1] Finite Efficient Theory
[2] Theory Saturation
[3] Termination
[4] Theory Generation
[5] Derivation Tree
[6] Size Shrinking

یک نسخه‌ی بهبود یافته‌ی آن از نظر معنایی (یعنی RV) شرایط اعمال روش تولید نظریه را دارند و یک بسته‌ی نرم افزاری برای پیاده سازی ایده‌ی فوق ارائه می‌کند [۷۳].

همانطور که در گفته‌های فوق مشهود است، بسیاری از توسعه‌های BAN ساختار معنایی به آن افزودند. Accorsi و بقیه ساختار معنایی ویژه‌ای برای BAN طرح کرده اند که سعی می‌کند مسأله‌ی "همه چیز دانی[1]" را در مورد این منطق حل کند [۷]. موضوع از این قرار است که در بررسی‌های نظری هنگامی که می‌گوییم یک عامل مجموعه‌ای از گزاره‌ها را می‌داند، به طور ضمنی فرض می‌شود که نظریه‌ی حاصل از آن مجموعه (نظریه‌ی حاصل از یک مجموعه از گزاره‌ها، مجموعه‌ای است از گزاره هاست که با اعمال نامتناهی قوانین استنتاج بر روی اعضای مجموعه‌ی اصلی بدست می‌آید [۱۳۳]) را هم می‌داند. در حالی که به علت محدودیت منابع و زمان در سیستم‌های رایانه‌ای (و حتی انسانی) این فرض درست نیست، به این معنی که یک رایانه همه‌ی آنچه را که می‌تواند بداند، نمی‌داند. از آنجا که این مسأله به طور جدی‌تری خودش را در مباحث هوش مصنوعی نشان می‌دهد محققین این شاخه اولین راه حل‌ها را برایش پیشنهاد کرده‌اند که از جمله‌ی آنها دیدگاه "هشیاری[2]" است [۵۳]. در دیدگاه هشیاری، دانش یک عامل به دو دسته‌ی دانش ضمنی[3] و دانش صریح[4] تقسیم می‌شود و حدس زدن اینکه هر کدام از آنها به چه بخشی از دانایی اشاره دارند ساده است. در همین راستا، Accorsi و بقیه یک ساختار معنایی مبتنی بر هشیاری برای BAN ارائه دادند [۷].

[1] The Omniscience Problem
[2] Awareness
[3] Implicit Knowledge
[4] Explicit Knowledge

٤-٢- منطق‌های دانش در برابر منطق های باور

من هرگز برای باورهایم نخواهم مُرد، چراکه ممکن است در اشتباه باشم.

برتراند راسل

این جمله‌ی راسل هرچند ممکن است کمی تلخ به نظر رسد، اما به نکته‌ای اشاره دارد که اساس تفاوت بین منطق‌های دانش و منطق‌های باور[1] است. منطق دانایی[2] و منطق باور[3] هر دو جزو منطق‌های وجهی[4] به شمار می‌آیند، با این تفاوت که در منطق دانایی دانش یک عامل از درستی یک گزاره به معنی درستی آن است، اما در منطق‌های باور ممکن است یک عامل به درستی یک گزاره باور داشته باشد اما آن گزاره درست نباشد [١٢٦]. به طور خلاصه اگر عملگر O نماینده دانش (یا باور) یک عامل مانند i باشد، اصل[5] زیر در منطق دانایی وجود داشته اما در منطق باور وجوبی ندارد:

$$O_i P \overset{?}{\Longrightarrow} P$$

منطق BAN و توسعه‌هایش منطق‌های باور هستند و این امکان وجود دارد که منطق‌های دانایی هم برای مدل سازی و درستی‌یابی پروتکل‌های رمزنگاری به کار روند. به نظر می‌رسد مهمترین کاری که در این زمینه انجام شده مقاله‌ای از Syverson باشد که در آن اصولاً نوع نگاه منطق‌های متفاوت به مسأله "امنیت" بررسی شده است [١٢٦]. فرض و بحث شهودی مقاله بر این است که موضوع "اعتماد[6]" (اساس تصدیق هویت) ریشه در باور دارد در حالیکه "محرمانه بودن" ریشه در دانش دارد. هر چند که به طور مستدل نمی‌توان این ادعا را ثابت کرد اما چندان دور از ذهن هم نیست. با این فرض، Syverson یک ترجمه‌ی نحوی (مسلما بدون توجه به معنا، اگر نه دو منطق یکسان می‌شدند) برای

[1] Logics of Belief
[2] Epistemic Logic
[3] Doxiastic Logic
[4] Modal Logics
[5] Axiom
[6] Trust

تبدیل هر کدام از این منطق‌ها به دیگری ارائه می‌دهد. از آنجا که اثبات قضایا در منطق دانش اصولاً ساده تر از منطق باور است (تلاشی برای اثبات درستی یک گزاره بعد از اثبات دانش یک عامل نسبت به درستی آن لازم نیست)، پیشنهاد می‌شود که حتی منطق‌های باور به منطق‌های دانش ترجمه شده وفقط ابزار کار در منطق دانایی توسعه یابد [۱۲۶]. تنها نقطه تاریک استدلال فوق این است که ممکن است در یک منطق باور عبارت $O_j P \& \sim P$ (نقیضِ اصلِ فوق) درست باشد، و ما نمی‌دانیم چطور این عبارت را در منطق دانایی (که چنین عبارتی همواره نادرست است) گنجانده و با آن کار کنیم. ادعای Syverson بر این است تنها مواقعی این اتفاق رخ می‌دهد که یک مشکل امنیتی از نوع محرمانه بودن هم وجود دارد (بر اساس مطالعات موردی که در مقاله ذکر شده است) و باز باید توسط منطق دانایی بررسی شود [۱۲۶]. او نام این موقعیت را باور نابجا[1] گذاشته و استدلال می‌کند با چنین باوری حتما محرمانه بودن اطلاعات هم برقرار نخواهد بود. به این ترتیب استفاده از منطق‌های دانش در درستی‌یابی صوری پروتکل‌های رمزنگاری دارای توجیه مناسبی شد. اولینِ آنها منطق CTK5 است که توسط Bieber ارائه شد [۱۹]. سپس Carlsen یک کاربرد جذاب از این منطق را نشان داد؛ به این صورت که یک حفره امنیتی در لایه‌ی پیاده سازی یک پروتکل را تشخیص داد [۳۰]. هر چند که این حفره تا حدی ساختگی بوده و به راحتی به حالت کلی قابل تعمیم نیست (مشکل مورد بحث ناشی از عدم تفکیک انواع داده‌ها[2] در پیام‌هاست) اما باز هم تنها نمونه‌ای است که یک مدل منطقی در لایه‌ی پیاده سازی پیش می‌رود. Carlsen همچنین ابزار تبدیل خودکار توصیف پروتکل به ساختار نحوی منطق CTK5 را هم ارائه داد [۳۱]. منطق KPL یکی دیگر از منطق‌های دانایی است که همزمان با CTK5 توسط Syverson ارائه شده است [۱۲۷]. هرچند که KPL همراه ساختار صوری معنایی مربوط به آن آمد اما اثباتی در مورد

[1] Misplaced Belief
[2] Data Types

صحت و تمامیت آن وجود ندارد [۱۲۷]. یک رویکرد تلفیق منطق دانش با احتمالات هم توسط Syverson و Gray مطالعه شده است [۱۲۸].

گذشته از بحث‌های Syverson در مورد استفاده از منطق دانایی بجای منطق باور، به نظر می‌رسد که یک دیدگاه محافظکارانه استفاده از هر دو منطق به صورت تلفیقی است مشروط به این که حجم محاسبات مورد نیاز دچار انفجار[1] نشود. جالب است که اولین منطق تلفیقی پیش از همه‌ی این داستان‌ها در سال ۱۹۸۹ توسط Moser استفاده شد [۱۰۰]. منطق Moser یک منطق نایکنواخت[2] نیز به شمار می‌آید که با معرفی عملگر "مگر[3]" ارزش نایکنواخت گزاره‌ها را مدل می‌کند [۱۰۰]. در این منطق با وجود $O_i p \; unless \; O_i q$ عامل i گزاره‌ی p را باور دارد مگر اینکه به گزاره‌ی q باور پیدا کند، پس باور نسبت به p ممکن است در زمان عوض شود [۱۰۰]. منطق تلفیقی Coffey و Saidha هم در سال ۱۹۹۶ برای درستی‌یابی پروتکل‌های رمزنگاری ارائه شد [۳۹]. هر دو منطق فوق عملگرهای مستقلی برای باور و دانش دارند. جدول ۱ مطالب فوق را خلاصه می‌کند.

جدول ۱- منطق‌های دانایی و باور

تلفیقی	منطق دانایی	منطق باور
Moser 89	Bieber 90 (CTK5)	R.V.Rangan 88 [۱۱۰]
Coffey, Saidha 96	Syverson 90 (KPL)	BAN 89
	Syverson 95 دیدگاه احتمالاتی	توسعه‌های BAN

[1] Explosion
[2] Non-monotonic Logic
[3] Unless

٤-٣- منطق‌های دیگر

یکی از انتقاداتی که ممکن است بر استفاده از منطق‌های خاص منظوره وارد باشد، نداشتن سابقه‌ی نظری کافی و به تبع آن نداشتن ابزار نرم افزاری کارآ ست. کمتر منطق‌دان حرفه‌ای وجود دارد که منطق BAN را مطالعه کرده باشد و شاید مشکلات معنایی BAN از اینجا باشد. هرچند که شرایط در حال تغییر است اما استفاده از منطق‌های متعارف[1] تدبیر نادرستی به نظر نمی‌رسد. البته نباید از نظر دور کرد که بیان و توصیف مسأله در یک منطق چند منظوره ممکن است به سادگی BAN نباشد. منطق زمانی در زمینه درستی‌یابی پروتکل‌ها سابقه‌ی طولانی دارد و تلاش برای استفاده از آن در مورد پروتکل‌های رمزنگاری ابتدا توسط Gray و McLean مطرح شد [٦٠]. به این ترتیب، مستقیماً پس‌زمینه‌ی نظری وحمایت نرم افزاری منطق زمانی شامل حال پروتکل‌های امنیتی می‌شود. همچنین Huima و Aura منطق S4n را که یک منطق چندوجهی[2] است وارد این تحقیقات کردند [٦٦]. در سال ۱۹۹۹ ایده‌ی استفاده از قطعه‌ای از منطق مرتبه اول برای این منظور مطرح شد [۱۳۵]. و در سال ۲۰۰۳ نشان داده شد که آن قطعه از منطق مرتبه اول تصمیم پذیر است [٤٢]. به همین ترتیب Delzanno از منطق شهودگرای hhf[3] و نرم افزار lambda-prolog برای توصیف و درستی‌یابی پروتکل‌های امنیتی استفاده کرد [٤٨].

منطق بازنویسی[4] یک ساختار منطقی درست و کامل است که ایده‌ی تغییر را در خود دارد. از نظر منطق بازنویسی هر قانون استنتاج معادل با بازنویسی یک عبارت با عبارت معادلش طبق قانون است [۸۸]. از این رو بسیاری از منطق‌های دیگر به نوعی یک منطق بازنویسی به شمار می‌آیند. حتی حساب لاندا[5] قابل بیان در قالب یک منطق بازنویسی است [۸۸]. بسته‌های نرم افزاری بسیاری برای پیاده سازی منطق بازنویسی وجود دارند که از میان آنها daTac توسط Jacquemard و بقیه (به همراه یک ابزار تولید

[1] Standard
[2] Multi Modal Logic
[3] hhf Intuitionistic Logic
[4] Rewriting Logic
[5] Lambda Calculus (λ-calculus)

خودکار توصیف صوری) و Maude توسط Denker و بقیه برای درستی‌یابی پروتکل‌های رمزنگاری به کار رفته‌اند [۵۱][٦٩].

در کنار این روش‌ها در سال ۱۹۹۳ یک ساختار صوری (مانند مدل Dolev-Yao) برای مدل سازی پروتکل‌های رمزنگاری توسط Woo و Lam ارائه شد که در برخی تحقیقات بعدی به کار گرفته شد [۱۳٦]. آنها برای بررسی امنیت از ایده‌های "تناظر" و "محرمانه بودن" استفاده کردند. تناظر به این معنی به کار رفت که در یک پروتکل تصدیق هویت همه‌ی دست اندر کاران باید در پروتکل قفل شده و تا انتهای آن حضور داشته باشند [۱۳٦]. برای عنوان مثال اگر تصدیق هویت کننده به انتهای وظایف خودش طبق پروتکل می‌رسد، طرف تصدیق هویت شونده باید حضور داشته و به مرحله‌ی آخر وظایف خودش رسیده باشد. با توجه به این ایده، آنها یک دستگاه استنتاج برای اثبات امنیت پروتکل‌های رمزنگاری ارائه کردند [۱۳٦].

فصل ۵: روش‌های جبری

۵-۱- حساب Spi

مقصود از روش‌های جبری، روش‌هایی هستند که اختصاصاً بر یک جبر فرآیند[1] تکیه دارند. از آنجا که هم جبر فرآیند و هم منطق گونه‌هایی از ساختارهای صوری هستند طبقه بندی جامع و مانعی برای آنها و به تبع آن بین استفاده آنها در درستی‌یابی پروتکل‌های رمزنگاری نمی‌توان قائل شد. در ساختارهایی که به لحاظ تاریخی منطق نامیده شدند، تکیه بیشتر بر جنبه‌های نظری بوده و معمولاً تنها ابزار اثبات درستی، روش اثبات قضیه است؛ در حالیکه جبر فرآیند در شاخه‌های کاربردی طراحی شده و در بسیاری از موارد دارای مشخصات مناسب جهت اثبات درستی به روش وارسی مدل هستند. در قالب جبر فرآیند حساب‌های گوناگونی برای مدل کردن سیستم‌های همگام[2]، نا همگام[3]، توزیع شده[4] و ... طراحی شده و به همراه ابزار مناسب برای درستی‌یابی خودکار به کار می‌روند. از آن میان می‌توان به CCS، CSP و

[1] Process Algebra
[2] Synchronous
[3] Asynchronous
[4] Distributed

ACP اشاره کرد [٥٤]. Milner در سال ۱۹۸۹ جبر فرآیند CCS را توسعه داده و حساب π[1] را ارائه کرد [۱۰۹]. در حقیقت حساب π در محاسبه‌ی همزمان[2] متناظر با حساب لاندا در محاسبه‌ی ترتیبی[3] است. حساب π می‌تواند برای مدل کردن پروتکل‌های مخابراتی به کار رود اما برای همخوانی بیشتر در سال ۱۹۹۸، Abadi و Gordon ساختارهای ابتدایی رمزنگاری را به آن افزوده و حساب Spi را معرفی کردند [۳]. حساب Spi علاوه بر ساختار نحوی حساب π دارای بندهای زیر هم هست: [۳]

در نحو عبارت[4] ها:

$\{M\}_N$: دارای معنای "رمزشده‌ی عبارت M با عبارت N" است. همانطور که دیده می‌شود تمهیدات لازم برای توصیف رمزنگاری با کلید عمومی وجود ندارد، هرچند که در [۳] ایده‌هایی ارائه شده اما مدل سازی کامل این مبحث به مطالعات بعدی واگذار شده است [۳].

در نحو پروسه‌ها:

$case\ \mathrm{L}\ \mathrm{of}\ \{\mathrm{x}\}_N\ \mathrm{in}\ \mathrm{P}$: به این معناست که پروسه سعی می‌کند عبارت L را با کلید N رمزگشایی کند، اگر L عبارتی به صورت $\{M\}_N$ باشد پروسه با $P[M/x]$ ادامه داده اگر نه متوقف می‌شود.

این نحو جدید با ساختار معنای مربوط به آن، فرض‌های زیر را به طور ضمنی به همراه دارد که در کاربردهای رمزنگاری اهمیت دارند: [۳]

- برای رمزگشایی یک پیام رمزشده تنها باید کلید آن را داشت و راه دیگری وجود ندارد.
- کلید رمزنگاری به همراه پیام رمزشده انتقال نیافته و در نتیجه فاش نمی‌شود.
- در پیام‌ها به اندازه‌ی کافی اضافات[1] وجود دارند، که یک پروسه می‌تواند تشخیص دهد که آیا متن رمزگشایی شده واقعاً یک پیام است یا فقط حاصل اعمال کلید اشتباه می‌باشد.

[1] Pi Calculus (π-calculus)
[2] Concurrent Computation
[3] Sequential Computation
[4] Term Syntax

به عنوان مثال توصیف نسخه‌ی ساده شده‌ی پروتکل "قورباغه‌ی دهان گشاد" که در بخش ۲-۱ بررسی شده را به حساب Spi می‌بینیم: [۳]

$$A(M) \stackrel{\Delta}{=} (\nu K_{AB})(\overline{c_{AS}}\langle\{K_{AB}\}_{K_{AS}}\rangle.\overline{c_{AB}}\langle\{M\}_{K_{AB}}\rangle)$$

$$S \stackrel{\Delta}{=} c_{AS}(x).case\ \mathrm{x\ of}\ \{\mathrm{y}\}_{\mathrm{K_{AS}}}\ \mathrm{in}\ \overline{\mathrm{c_{SB}}}\langle\{y\}_{K_{SB}}\rangle$$

$$B \stackrel{\Delta}{=} \left| \begin{array}{l} c_{SB}(x).case\ \mathrm{x\ of}\ \{\mathrm{y}\}_{\mathrm{K_{SB}}}\ \mathrm{in} \\ \mathrm{c_{AB}}(z).case\ \mathrm{z\ of}\ \{\mathrm{w}\}_{\mathrm{y}}\ \mathrm{in\ F(w)} \end{array} \right.$$

$$Inst(M) \stackrel{\Delta}{=} (\nu K_{AS})(\nu K_{SB})(A(M) \mid S \mid B)$$

همانطور که در بخش ۲-۱ دیدیم در این پروتکل سه عامل A، B و KDC (که اینجا S نامیده می‌شود) شرکت دارند. پروسه‌ی A می‌خواهد پیام M را برای B بفرستد. به این منظور یک کلید جلسه‌ی نو مانند K_{AB} تولید کرده (νK_{AB}) و سپس آنرا با کلید مشترکش با سرور S رمز کرده و از طریق کانالی که با S دارد برایش می‌فرستد ($\overline{c_{AS}}\langle\{K_{AB}\}_{K_{AS}}\rangle$). سپس پیام M را با کلید جلسه‌ی K_{AB} رمز کرده و از طریق کانال مشترکش با B آنرا برای B می‌فرستد. سرور S روی کانال AS منتظر پیام رمزشده‌ای است که از A می‌آید، اگر چنین پیامی وارد شود (رمزشده با کلید K_{AS} باشد) محتوای آنرا با کلید مشترکش با B (K_{SB}) رمز کرده و برای B می‌فرستد. با این تفاسیر مشخص است که پروسه‌ی B چه می‌کند. تنها نکته این است که B پیام رسیده از A را بعد از رمزگشایی به پروسه‌ی F می‌سپارد. Inst هم نماینده‌ی کل پروتکل است که با فرض وجود کلیدهای K_{AS} و K_{SB} پروسه‌های A، B و S به طور موازی در آن زندگی می‌کنند. ایده‌ی مهم حساب Spi نوع نگاه ویژه‌ی آن به "امنیت" است. فرض کنید می‌خواهیم بررسی کنیم که پروتکل فوق هیچگاه مقدار M را فاش نمی‌کند، کافی است مطمئن شویم که هر پروسه-ی Spiی دیگر که موازی با آن اجرا می‌شود نمی‌تواند بین اینکه آیا این پروتکل با مقدار M اجرا می-شود یا با مقدار M` تشخیص قائل شود. مفهوم تشخیص ناپذیری[۲] پروسه‌ها نقش اصلی را در این ایده

[1] Redundancy
[2] Indistinguishability

بازی می‌کند. البته باید توجه داشت که تشخیص ناپذیری در مورد پروسه‌های رمزنگاری به نوعی یک خاصیت ضمخت[1] است. فرض کنید پروسه‌ی A مقدار M را با یک کلید محرمانه که فقط خودش اطلاع دارد رمز کرده و روی شبکه می‌فرستد، اگر A مقدار `M را رمز کند و بفرستد پروسه‌ی مهاجم قادر است بین این حالات تفاوت شود (چرا که پیام‌های رمز شده به هرحال متفاوت هستند) اما باز هم ما پروسه‌ی A را ناقض محرمانه بودن نمی‌دانیم. در پروتکل "قورباغه‌ی دهان گشاد" فرض می‌کنیم که پروسه‌ی F پیام M را فاش نمی‌کند، به این ترتیب خاصیت محرمانه بودن به صورت زیر بیان می‌شود: [۳]

$$Inst(M) \approx Inst(M') \text{ if } F(M) \approx F(M'), \forall M, M'$$

در این بیان، ≈ نشان دهنده‌ی مفهوم تشخیص ناپذیری است.

در مورد خاصیت مهم دیگر امنیت یعنی "تصدیق هویت"، ایده این است که پروسه‌ی B در حالت ایده‌آل می‌داند که پیام رسیده واقعا از طرف A است یا نه و آن را به F می‌سپارد (B باید A را تصدیق هویت کند). در صورت برقراری تصدیق هویت درست در این پروتکل انتظار داریم که از دید پروسه‌ی خارجی رفتار پروتکل در دو حالت (ایده آل و واقعی) یکسان باشد. پس داریم: [۳]

$$Inst(M) \approx Inst_{spec}(M), \forall M$$

که در آن $Inst_{spec}(M)$ نمونه پروتکلی است که در آن پروسه‌ی B با پروسه‌ی ایده آل فوق جایگزین شده و از هر جهت دیگر (از جمله بخش F) مشخصات یک B عادی را دارد. این نوع نگاه به خاصیت‌های امنیتی نتیجه‌ی بسیار جالبی را در مورد مدل صوری مهاجم به دنبال دارد. تقریباً واضح است که در این چهارچوب هیچ شرطی بر روی محیط متخاصم گذاشته نمی‌شود، هر پروسه‌ی Spi می‌تواند

[1] Coarse-Grained

نماینده‌ی یک مهاجم باشد که با توجه به جامع[1] بودن حساب π در مدل کردن پروسه‌ها تقریباً هیچ محدودیتی برای عامل مهاجم وجود ندارد. در نتیجه قوی ترین مدل مهاجم، مهاجم حساب Spi است [۳].

برای اثبات تشخیص ناپذیری پروسه‌ها، مقاله‌ی اصلی Abadi و Gordon یک روش مبتنی بر همسازی قوی[2] حساب π را برای کاربردهای رمزنگاری اصلاح کرده و روش همسازی خاردار[3] را ارائه می‌دهد [۳]. این روش در [٤] بهبود یافته و صحت آن را در قالب حساب Spi اثبات شد. روش دیگر اثبات توسط Boreale ارائه شد که همراه با یک بسته‌ی نرم افزاری می‌آید و بر اساس مطالعه‌ی نمادین گذار پروسه[4] ها و رد پا[5] بنا شده است [۲۳]. در ادامه صحت روش مطالعه‌ی رد پا در قالب Spi اثبات شده و یک گونه‌ی *تمام*[6] آن هم معرفی شد [۲٤].

هرچند که حساب Spi به طور مستقیم به کار گرفته نشد و ابزار اثبات مناسبی ندارد، اما بسیاری از روش‌های که در بخش‌های بعد می‌بینیم ایده‌های آن را به کار برده‌اند. البته این موضوع چندان هم غیرطبیعی نمی نماید، معمولاً یک ساختار جبر فرآیند تنها حامی نظری ابزار به شمار می‌آید.

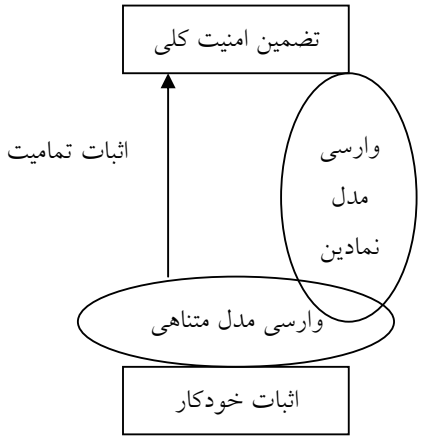
۵-۲- وارسی مدل

این بخش و بخش بعدی بر اساس روش اثبات برای جبر فرآیندها تقسیم بندی شده اند. به این معنی که در بخش "وارسی مدل" جبر فرآیندهایی را خواهیم دید که از روش وارسی مدل سعی می‌کنند درستی را اثبات کنند. همانطور که در بخش ۲-۲ آمد، روش‌های وارسی مدل معمولاً یک مدل متناهی (یا متناهی

[1] Universal
[2] Strong Bisimulation
[3] Barbed Bisimulation
[4] Process Transitions
[5] Trace
[6] Complete

شده) را در نظر می‌گیرند و از این نظر اثبات امنیت پروتکل در حالت کلی ممکن نیست. در عوض، متناهی بودن مدل امکان خودکار اثبات را به همراه دارد که مورد نیاز ابزار اثبات به کمک رایانه است. حال اگر بتوان مدل متناهی یافت که امن بودنِ آن به معنی امنیت مدل اصلی (نه لزوماً متناهی) باشد، همه چیز را با هم خواهیم داشت: تضمین امنیت کلی و اثبات خودکار به کمک رایانه.

از طرف دیگر ایده‌هایی برای وارسی مدل نمادین پروتکل‌های رمزنگاری وجود دارند که علاوه بر حفظ خودکار بودن فرآیند اثبات، مدل‌های نامتناهی را هم ارزیابی کنند. این دو رویکرد محتوای این بخش را تشکیل می‌دهند که در ادامه بررسی خواهند شد. شکل ۲ طرح وارسی مدل با هدف رسیدن به تضمین امنیت کلی را نشان می‌دهد.



شکل ۲- طرح وارسی مدل

۵-۲-۱- وارسی مدل: متناهی حالت

جالب ترین نتیجه در این بخش اثبات تمامیت وارسی مدل برای خانواده‌ی خاصی از پروتکل‌ها توسط Lowe است [۸۰]. Lowe شرایطی (در ادامه خواهد آمد) را برای آن خانواده فرض کرده و اثبات می‌کند هر عضو آن خانواده اگر "کوچک سیستم امن[1]" باشد کاملاً امن است [۸۰]. به عبارت دیگر وارسی مدل چنین پروتکلی با یک مدل کوچک (یک بازیگر درستکار برای هر نقش پروتکل و یک عامل

[1] Small System Secure

مهاجم)، برای اثبات امنیت تمامیت دارد. شرایط Lowe برای این خانواده به سادگی قابل بررسی هستند: [۸۰]

اصول وضوح[1]:

۱- اگر هویت یک بازیگر در معنی یک پیام تاثیر داشته باشد، باید نامش به طور روشن در پیام ذکر شود.

۲- اگر ساختارهای متفاوت پیام معانی متفاوتی را می‌رسانند، باید نوع ساختار یک پیام به وضوح قابل تشخیص بوده و امکان اشتباه بین آنها وجود نداشته باشد.

فرضیات:

۱- (فرض مقادیر اتمی) فرض می‌شود انواع گوناگون داده (مانند هویت عامل‌ها، کلید ها و ...) قویاً نوعدار[2] بوده و هیچ امکان اشتباه بین آنها نیست. همچنین مقادیر بلند مدت[3] که وابسته به عوامل درگیر در پروتکل هستند (مانند کلید عمومی عامل‌ها که در اجراهای مختلف پروتکل ثابت هستند) با هم متفاوت و قابل تفکیک هستند.

۲- همه‌ی متغیرهای محرمانه‌ی گذرا نو[4] هستند.

۳- هیچ مقدار محرمانه‌ی بلند مدتی (مانند رمز کارت در SET) نباید در پیام‌ها ظاهر شود.

٤- همه‌ی زیر بخش‌های رمزشده‌ی یک الگوی پیام باید نام همه‌ی بازیگران درگیر را به همراه داشته باشند.

۵- قالب پیام‌های رمز شده باید طوری باشد که بتوان تشخیص داد آن قطعه مربوط به کدام پیام و کدام قطعه از آن است.

[1] Explicitness Principles
[2] Strictly Typed
[3] Long Term
[4] Fresh

٦- این امکان وجود ندارد که یک مقدار محرمانه به صورت موضعی استفاده شده و سپس برای همه افشاء گردد (هیچ پروتکل واقعی‌ای این شرط را نقض نمی‌کند).

همانطور که در شرایط فوق دیده می‌شود بررسی آنها کار ساده‌ای است، اما بسیاری از پروتکل‌ها (از جمله کربروس) آنها را نقض می‌کنند. به علاوه قضیه‌ی تمامیت Lowe فقط برای مشخصه‌ی محرمانه بودن است و شامل تصدیق هویت نمی‌شود. در سال ۲۰۰۱، Stoller نشان داد که این شرایط قابل تعدیل بوده و مجموعه‌ی پروتکل‌های فوق قابل توسعه است اما بررسی شرایط جدید وی پیچیده (هرچند تصمیم پذیر) است [۱۲۳].

ابزار وارسی مدلِ گوناگونی برای درستی‌یابی پروتکل‌های رمزنگاری اختصاصاً توسعه یافته و یا به کار رفته اند. در ادامه یک مرور تاریخی از تجربه‌های مشهور را می‌بینیم.

۱- تحلیل کننده‌ی پروتکل NRL:

این ابزار اصولاً برای ارزیابی پروتکل‌های رمزنگاری توسط Meadows طراحی شده و از استقراء[1] برای کاهش مدل استفاده می‌کند [۹۳].

۲- CSP/FDR:

در سال ۱۹۹۵ برای اولین CSP و ابزار وارسی مدل FDR در درستی‌یابی یک پروتکل تعویض کلید به کار رفت [۱۱۲]. اما کار Lowe در پیدا کردن یک حمله‌ی ناشناخته علیه پروتکل معروف Needham-Schroeder بوسیله‌ی CSP/FDR در سال ۱۹۹٦ باعث توجه جدی به کاربرد CSP و وارسی مدل در این زمینه شد [۷۹]. این روش برای درستی‌یابی پروتکل TMN هم به کار رفت [۸۱]. همچنین Lowe یک مترجم خودکار برای تولید توصیف CSP از یک نمایش مجرد از پروتکل‌ها ارائه کرد که Casper نام دارد [۸۲].

[1] Induction

۳- مورفی[1]:

در سال ۱۹۹۷، Mitchell و بقیه ایده استفاده از بستر وارسی مدل مورفی را در زمینه‌ی پروتکل‌های رمزنگاری با بررسی پروتکل‌های مهمی نظیر کربروس و TMN به نمایش گذاشتند [۹۷]. در حقیقت مورفی یک سیستم وارسی مدل بسیار بهینه می‌باشد که دارای روش‌های کاهش حالت[2] ویژه‌ی پروتکل-های رمزنگاری هم هست [۱۱۸]. تحلیل پروتکل SSL توسط مورفی یکی از معدود نمونه‌های کاربرد روش‌های صوری در پروتکل‌های رمزنگاری در حد تجاری به شمار می‌رود [۹۸].

٤- بروتوس[3]:

بروتوس یک ابزار وارسی مدل تک منظوره برای پروتکل‌های رمزنگاری است [۳٦]. ریشه‌های توسعه‌ی بروتوس را می‌توان در [۳۷، ۸٦ و ۸۷] یافت. همچنین فن کاهش درجه‌ی جزئی[4] هم در بروتوس منظور شده است [۳۸].

۵- LOTOS/CADP:

Leduc و Germeau زبان توصیف همه منظوره‌ی LOTOS به همراه یک ابزار درستی‌یابی (CADP) را برای درستی‌یابی پروتکل OKAPI به کار بردند [۷۵] [۷٦]. از نظر استفاده از مترجم‌های میانی و ابزار درستی‌یابی معمول، این تجربه به مثال استفاده از *μCRL* در درستی‌یابی پروتکل Needham-Schroeder شبیه است [۱۰٤].

۵-۲-۲- وارسی مدل: نامتناهی حالت

تحلیل مدل‌های با نامتناهی حالت، تنها با روش‌های استنتاج و استقرای نمادین امکان پذیر است [٤۰]. هر کدام از الگوریتم‌هایی که در این بخش ذکر می‌شوند یک جبر فرآیند (چند منظوره یا ویژه) را به کار

[1] Murφ
[2] State Reduction Techniques
[3] BRUTUS
[4] Partial Order Reduction

گرفته و سعی می‌کنند با ایده‌های ابتکاری از پس مشکلات کار کردن با حالات نامتناهی برآیند. همه‌ی آنها فرض می‌کنند که تعداد نشست‌های پروتکل کراندار است، بنابراین توانایی‌های مدل مهاجم (Dolev-Yao) است که حالات نامتناهی را تولید می‌کند. الگوریتم‌های معروف این زمینه فهرست‌وار می‌آیند.

- Huima : یک جبر فرآیند ویژه به کار می‌رود [٦٧].
- Basin : بر اساس نوع داده کُند[1] است [١٢].
- Amadio و بقیه : بر اساس یک نسخه از حساب Spi است [٩، ١٠].
- Goubalt-Larrecq : یک خودکاره‌ی ویژه (خودکاره‌ی درخت پارامتری[2]) برای پذیرش دانش یک مهاجم Dolev-Yao معرفی شده است [٥٩].
- Millen و Shmatikov : یک رویکرد مبتنی بر فضای رشته[3] به کار می‌رود [٩٥].
- Bozzano : بر اساس بازنویسی چند مجموعه‌ای است [٢٦].
- Boreale : بر اساس حساب Spi است [٢٣، ٢٥].

تنها الگوریتمی که در تصمیم گیری شرط کراندار بودن جلسه‌ها را بر می‌دارد، یک روش مبتنی بر عبارات هورن[4] است که نادرست[5] است و تضمینی بر پایان پذیری آن وجود ندارد [٢٠]. در حقیقت بدون داشتن این شرط مسأله تصمیم پذیر نبوده و نتایج فوق دور از ذهن نیست [٥٢].

[1] Lazy Data Type
[2] Parameterized Tree Automata
[3] Strand Space
[4] Horn Clauses
[5] Unsound

۵-۳- اثبات قضیه

روش اثبات قضیه به لحاظ تولید نتایج کلی در مورد امنیت بسیار جذاب می‌نماید، اما مشکلاتی که در بخش ۲-۲ ذکر شد گریبانگیر این روش است. شاید بتوان گفت روش استقرایی Paulson حاوی مهمترین ایده در این زمینه است که در روشن شدن طبیعت پروتکل‌های رمزنگاری و جنبه های درستی-یابی آن هم موثر بوده است [۱۰۵]. فرض کنید که هر اجرای پروتکل یک *ردپا* را تشکیل می‌دهد، ایده این است که در هر قدم این ردپا توسط *رویداد*[1]های ویژه‌ای که از پیش تعیین شده هستند (مثلاً قوانین پروتکل یا توانایی‌های مهاجم) پیش می‌رود. حال اگر برای یک ردپای تهی یک خاصیت امنیتی برقرار بوده و بتوانیم اثبات کنیم که هر کدام از رویدادها، آن خاصیت را در مورد یک ردپای امن به هم نمی‌زنند طبق استقراء برای حالت کلی امنیت پروتکل را اثبات کرده ایم. به زبان ریاضی: [۱۰۵]

$$P([])$$
$$\forall ev : P(evs) \Rightarrow P(ev\#evs)$$

که P نشان دهنده‌ی یک خاصیت امنیتی، [] نشان دهنده‌ی ردپای تهی، evs یک ردپا از رویدادها و ev هر کدام از رویدادهای محتمل است [۱۰۵]. توصیف زیر بخشی از تشکیل ردپای پروتکل Yahalom را نشان می‌دهد: [۱۰۶]

$$[evs \in Yahalom; Says\ A\ B\ X \in set\ evs]$$
$$\Rightarrow Gets\ B\ X\ \#\ evs \in Yahalom$$

که معادل رویداد زیر است:

$$A \rightarrow B : X$$

مهاجم توسط عملگرهای جبری زیر توصیف می‌شود: [۱۰۵]

[1] Event

فرض کنید H مجموعه‌ی شامل دانش اولیه‌ی مهاجم و تمام پیام‌هایی باشد که تا کنون روی شبکه رد و بدل گشته، $\text{analz } H$ مجموعه‌ای است که به صورت نامتناهی با افزودن اجزاء پیام‌های مرکب و رمزگشایی پیام‌هایی که کلیدشان در $\text{analz } H$ است به دست می‌آید:

$$\textit{Crypt}\ K\ X \in \text{analz } H, K^{-1} \in \text{analz } H \Rightarrow X \in \text{analz } H$$
$$\text{analz}\,(\text{analz } H) = \text{analz } H$$

منظور از $\text{Crypt } K\ X$ رمزشده‌ی پیام X با کلید K است.

$\text{synth } H$ مجموعه‌ای است که به صورت نا متناهی با تشکیل پیام‌های مرکب از اجزاء، افزودن نام‌ها و رمزکردن پیام‌ها با کلیدهای موجود در H تشکیل می‌شود:

$$X \in \text{synth } H, K \in H \Rightarrow \textit{Crypt}\ K\ X \in \text{synth } H$$
$$K \in \text{synth } H \Rightarrow K \in H$$
$$\text{synth}\,(\text{synth } H) = \text{synth } H$$

به این ترتیب مهاجم تمام آنچه را که روی شبکه می‌گذرد در H ذخیره کرده و می‌تواند پیام‌هایی را از مجموعه‌ی $\text{synth}(\text{analz } H)$ برای دیگر بازیگران بفرستد [۱۰۵].

Paulson یک دستگاه استنتاج برای این ساختار طراحی و در ابزار Isabelle (کمک اثبات[1] برای منطق مرتبه‌ی بالاتر[2]) پیاده سازی کرده و به کمک آن یک حمله‌ی ناشناخته علیه‌ی نسخه‌ای از پروتکل Otway-Rees را گزارش کرد [۱۰۵]. البته شاید تجربه‌ی Paulson در کار با HOL و Isabelle بود (وی یکی از توسعه دهندگان کلیدی Isabelle است) که توانست یک حمله علیه پروتکل فوق پیدا کند چرا که اگر یک ابزار اثبات قضیه در اثبات موفق نشود، در مورد رد یابی این شکست کمک زیادی به کاربر نمی‌دهد. به هر حال Isabelle/HOL برای اثبات قضایایی در مورد امنیت پروتکل‌های مهمی نظیر کربروس، بخشی از SET و TLS به کار رفت [۱٤، ۱۵، ۱۰۷].

در ادامه یک فهرست از آزمایش‌ها در زمینه اثبات قضایای امنیتی را می‌بینیم.

[1] Proof Assistant
[2] Higher Order Logic (HOL)

- Kemmerer : مبتنی بر ابزار Ina Jo [۷۰].
- Paulson : مبتنی بر ابزار Isabelle/HOL [۱۰۵].
- Schneider و بقیه : جبر فرآیند CSP مبتنی بر ابزار PVS [۲۸، ۱۱۷].
- Bolognano : مبتنی بر ابزار Coq [۲۲].
- Nylen : مبتنی بر ایده‌های کنترل سیستم های نامتناهی حالت [۱۰۳].
- De Decker و Piessens : ابزار کمک اثبات خاص منظوره‌ی CryptoLog [٤٧].
- Krishnan و Renaud : یک جبر فرآیند خاص منظوره و ابزار PVS [٧٤].
- Delzanno و Etalle : یک حساب رشته‌ها[1] به همراه ابزار مبتنی بر Prolog [٤٩].

٥-٤- رویکردهای دیگر

فضای رشته[2]

یک رشته، دنباله‌ای از پیام‌هایی است که یک بازیگر معمولی یا مهاجم در یک اجرای پروتکل فرستاده یا دریافت می‌کند. هنگامی پروسه‌ی مربوط به یک رشته پیامی را فرستاده و پروسه‌ی رشته‌ی دیگری همان پیام را دریافت می‌کند، آن دو رشته توسط گره[3] به هم متصل می‌شوند. در حالت کلی یک گره رشته‌های همه‌ی بازیگران یک پروتکل را به هم متصل می‌کند. در سال ۱۹۹۸ ، Thayer و بقیه یک توصیف صوری از ایده‌ی فوق را به همراه یک فرمول بندی سببی[4] و یک دستگاه استنتاج ارائه کردند و آن را فضای رشته نامیدند [۱۳۱]. درستی یک پروتکل رمزنگاری در این دیدگاه شبیه مدل Woo و Lam است که در بخش ٤-۳ معرفی شد. در فضای رشته اثبات قضیه‌ها کوتاه هستند و خودکار نبودن دستگاه

[1] Sequent Calculus
[2] Strand Space
[3] Bundle
[4] Causal

استنتاج مشکل محسوسی به شمار نمی‌آید [۱۳۱]. این ایده در مسیرهای مختلفی پیشرفت کرده و مفاهیم عمیق تری بر اساس آن توسعه یافتند [٦٢، ١٣٢]. در سال ۱۹۹۹ یک ساختار معنایی مبتنی بر فضای رشته برای منطق BAN پیشنهاد شد و در سال ۲۰۰۰ هم یک طرح وحدت[1] بین فضای رشته‌ها و بازنویسی چند مجموعه‌ای ارائه شد که عملاً مدل غالبِ دیدگاه جبری به شمار می‌رود [٣٤، ١٢٩]. از این نظر فضای رشته به عنوان یک بستر نظری که تقریباً همه‌ی دیدگاه‌های درستی‌یابی پروتکل‌های رمزنگاری را در خود جای داده، اهمیت دارد. حاصل این همبستگی نظری ابزار "آتنا[2]" است. آتنا یک ساختار بسیار بهینه برای درستی‌یابی خودکار پروتکل‌های رمزنگاری به شمار می‌رود (آتنا در مقایسه با مورفی یا بروتوس تقریباً بدون مکث جواب می‌دهد [۱۲۱]) که دارای یک منطق ساده برای بیان مشخصات امنیتی مورد نظر، یک ارزیاب مدل همراه با ایده‌های کاهش حالت برگرفته از فضای رشته و دستگاه اثبات قضیه برای تحدید فضای جستجو (در مسیرهایی که دست نایافتنی[3] هستند) است [۱۲۱]. آتنا فرض می‌کند که تعداد جلسه‌های ارتباطی و طول پیام‌ها کراندار است، در غیر این صورت، تضمینی برای پایان پذیریِ الگوریتم آتنا وجود ندارد؛ هر چند این امکان پیش بینی شده که کاربر شرایط فوق را حذف کرده و (مانند کمک اثبات‌ها) فرآیند جستجو را خودش هدایت کند [۱۲۱].

ارزیابی نوع[4]

ارزیابی نوع یک الگوریتم سبُک از نظر محاسباتی بوده و با وجود برخی ضعف‌هایش در بسیاری از پروژه‌های تجاری که محدودیت زمانی دارند شاید بهترین راه حل به شمار آید. ایده‌ی استفاده از ارزیابی نوع در درستی‌یابی پروتکل‌های رمزنگاری در سال ۱۹۹۸ توسط Abadi ارائه شد [۵]. به طور ساده، به انواع داده و کانال‌های مخابراتی برچسب "امن" و "نا امن" داده شده و بر روی اجراهای مختلف

[1] Unification
[2] Athena
[3] Unreachable
[4] Type Checking

پروتکل ارزیابی نوع انجام می‌شود و هرگاه یک داده‌ی از نوع امن روی یک کانال نا امن ظاهر شد، یک خطای محرمانه بودن به حساب می‌آید [۵].

در سال ۲۰۰۱ Gordon و Jeffrey یک بسته‌ی نرم افزاری به نام CRYPTYC برای این منظور طراحی کردند [۵۷]. در سال ۲۰۰۲ هم امکان کار با رمزنگاری نامتقارن به فرمول بندی اولیه‌ی آن افزوده شد [٦].

شبکه‌های پتری[1]

Crazzolara و Winskel یک جبر فرآیند همراه با ساختار معنایی شبکه‌های پتری برای درستی‌یابی پروتکل‌های رمزنگاری ارائه کردند [٤٤]. شبکه‌های پتری ساختارهای جبری رویدادگرایی[2] هستند که سابقه‌ی طولانی در بررسی سیستم‌های موازی و همزمان دارند. به لحاظ "شبکه" بودن، الگوریتم‌های گراف در تعیین وضعیت حالات آن نظیر "دسترس پذیری[3]" و ... به کار می‌روند [۱۰۱]. از این رو استفاده از این مفاهیم و ابزار در درستی‌یابی پروتکل‌های رمزنگاری طبیعی می‌نماید. از نظر رویدادگرایی این روش با ایده‌ی استقراء Paulson و از نظر بررسی یک رشته از رویدادها با فضای رشته ارتباط نزدیکی دارد [٤٤]. روش خام تری هم برای استفاده از شبکه‌های پتری در مبحث پروتکل های امنیتی در [۱۳] ارائه شده است.

[1] Petri Nets
[2] Event Driven
[3] Reachablity

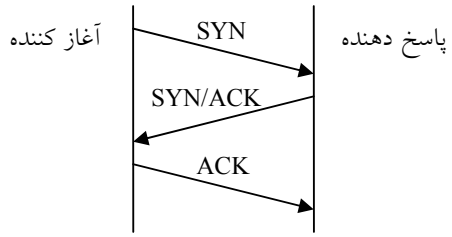
٦- وارسی صوری پروتکل TCP/IP در برابر حمله‌ی SYN

٦-١- پیش زمینه

در بخش‌های قبل امنیت تنها در قالب مشخصات محرمانه بودن و تصدیق هویت بررسی شد. به طور طبیعی در آن مدل‌ها عامل مهاجم تلاش می‌کند به سرویس‌های مورد نظرش دسترسی غیر مجاز پیدا کند، اما در گروه دیگری از حملات، مهاجم سعی دارد از دسترسی عامل‌های مجاز به یک سرویس جلوگیری کند. این حملات به عدم سرویس دهی[1] معروف هستند. شاید معروف ترینِ این حملات، حمله‌ی SYN (به مناسبت یک پرچم[2] در سربارهٔ بسته‌ی[3] حمله) علیه پروتکل TCP/IP باشد که با توجه به استفاده‌ی وسیع TCP/IP به راحتی می‌توان خطر گسترده‌ی این حمله را حدس‌زد [٤٦]. مرحله‌ی اول پروتکل TCP/IP، توافقِ سه جانبه[4] است [١٣٠]. شکل ٣ این مرحله را نشان می‌دهد.

نکته اینجاست که هنگامی که پاسخ دهنده یک بسته‌ی با سربارهٔ SYN دریافت می‌کند، مقداری حافظه برای نشستی که تصور می‌کند آغاز خواهد شد اختصاص می‌دهد. مهاجم می‌تواند با تعداد زیادی

[1] Denial of Service (DoS)
[2] Flag
[3] Packet Header
[4] Three Way Handshake



شکل ۳- توافق سه جانبه

بسته‌ی جعلی حافظه‌ی پاسخ دهنده را اشباع کند و از برقراری ارتباط عامل‌های مجاز جلوگیری کند (تعداد بسته‌های مورد نیاز جعلی حاوی SYN در عمل چندان زیاد نیست) [٤٦].

از دیدگاه رمزنگاری، این ضعف ناشی از تصدیق هویت نشدنِ آغاز کننده توسط پاسخ دهنده است. اما هر پروسه‌ی تصدیق هویت خودش (به لحاظ بار محاسباتی پاسخ دهنده) می‌تواند یک حفره‌ی عدم سرویس دهی به شمار آید. از این رو، حمله‌ی SYN به عنوان یک مدل نظری از حمله علیه پروتکل-های تصدیق هویت هم مطرح می‌باشد [٩٤].

به نظر می‌رسد که در هر صورت، چه وجود و چه عدم وجودِ لایه‌ی تصدیق هویت منجر به حفره‌ی عدم سرویس دهی می‌شود. این پیچیدگی ایده‌ی تصدیق هویت مرحله‌ای را به دنبال داشت [٩٤]. ابتدا یک الگوریتم تصدیق هویت سبُک اجرا می‌شود و در صورت گذشتن از این مرحله، تصدیق هویت کامل انجام خواهد شد [٩٤]. به این ترتیب، حجم محاسبه‌ی زیادی بر پاسخ دهنده تحمیل نمی‌شود و حمله‌ی عدم سرویس دهی تنها برای مهاجم های بسیار قوی ممکن خواهد بود که از نظر عملی وجود ندارند [٩٤]. در اینجا نکته‌ی اصلی "مصرف منابع" توسط پاسخ دهنده است. از این رو مدل‌هایی که در بخش-های قبل دیدیم برای بررسی حملات عدم سرویس دهی چندان مناسب نیستند و یک مدل مبتنی بر مصرف منابع سیستم طبیعی‌تر می‌نماید. از طرف دیگر، مدل Dolev-Yao به مهاجم اجازه‌ی قطع ارتباط بازیگران پروتکل را می‌دهد که نتیجه‌ی مستقیم آن می‌تواند عدم سرویس دهی باشد.

نتیجه‌ی تفکر تصدیق هویت چند مرحله‌ای برای مقابله با حمله‌ی SYN روش SYN Cookie است [۱۱۱]. پاسخ دهنده در حالت عادی بعد از دریافت یک SYN، مقداری حافظه برای دنبال کردن جلسه-ی ارتباطی فرضی تخصیص می‌دهد، اما در صورت استفاده از SYN Cookie، تنها یک مقدار درهم شده[1] از شناسه‌ی خودش و فرستنده و ... را تولید کرده و برای آغازکننده می‌فرستد (در حقیقت مقادیری که می‌بایست در سمت پاسخ دهنده ذخیره شوند حالا به سمت عامل آغازگر فرستاده می‌شوند) [۱۱۱]. اگر آغازگر جعلی باشد هرگز این مقادیر بر نخواهند گشت، در غیر این صورت پاسخ دهنده با دریافت آنها در بسته‌ی ACK جلسه ارتباطی را برقرار و دنبال می‌کند. این روش امروزه در هسته‌ی[2] سیستم عامل لینوکس[3] به کار می‌رود [۷۷]. با وجود این مرحله (تصدیق هویت اولیه)، تنها در صورتی که مهاجم قادر به انجام محاسبات و فرستادن بسته‌ها در حد 2^{23} بسته در ٤ ثانیه باشد می‌تواند در حمله‌ی SYN موفق باشد [۷۷].

٦-۲- توصیف و درستی‌یابی

در این بخش یک ساختار نوین مبتنی بر ***مصرف منابع*** برای توصیف و درستی‌یابی حملات عدم سرویس‌دهی ارائه خواهد شد. به عنوان یک مطالعه‌ی موردی، قطعه‌ی توافق سه جانبه از پروتکل TCP/IP مدل شده و وجود حمله‌ی SYN علیه آن بررسی می‌شود. سپس امکان SYN Cookie به پروتکل اضافه شده و اثبات می‌شود که (در قالب پیش فرض‌های مشخص) پروتکل امن است.

از آنجا که بسیاری از جزئیات حمله‌ی SYN به لایه‌ی پیاده سازی بر می‌گردد نکاتی در مدل TCP/IP وجود دارند که از دانش نسبت به پیاده سازی نشأت می‌گیرند و توضیح آنها در ادامه خواهد آمد.

[1] Hashed Value
[2] Kernel
[3] Linux

در این مطالعه از SMV نسخه‌ی ۲/۵۰ برای سیستم عامل Microsoft Windows NT® استفاده شده است. SMV یک ابزار وارسی مدل متناهی حالت با زبان توصیف مشخصه‌ی مبتنی بر منطق زمانی CTL[1] است [۹۱]. SMV اولین ابزاری است که از مفاهیم درخت تصمیم دودویی[2] استفاده کرده و جزو پیشگامان فن آوری وارسی مدل به حساب می‌آید [۶۸]. هرچند ساختار نحوی و معنایی SMV بسیار به زبان طبیعی نزدیک است ولی برای جزئیات بیشتر راهنمای کاربر آن توصیه می‌شود [۹۱].

درستی‌یابی در مدل مصرف منابع: قالب صوری

فرض می‌کنیم سطح مصرف منابع توسط هر پروسه توسط اعضای یک الفبا[3] مانند Z بیان می‌شود. به این ترتیب تابع مصرف لحظه‌ای به صورت زیر تعریف می‌شود:

$$\forall A \in \text{Names}, \widetilde{C}(A, j) : \pi(A, j) \to Z$$

مطابق مدل پیوست ۲، π یک اجرای درست از پروتکل و A معرف یک بازیگر پروتکل است. تابع *مصرف* را برای یک عامل A به صورت زیر تعریف می‌کنیم:

$$C(A,i) = \max_{j<=i}\{\widetilde{C}(\pi(A, j))\}$$

تابع R به صورت زیر تعریف می‌شود که $R(x \in Z) = 1$ نشان دهنده‌ی زیر مجموعه‌ی خاصی از سطح مصرف توان است که *خطرناک* به‌شمار می‌آیند:

$$R : Z \to \{0,1\}$$

به این ترتیب یک حمله‌ی عدم سرویس‌دهی عبارت است از یک اجرای درست مانند π که:

$$\forall A \in \text{Names} - \{\text{ATTACKER}\}, \exists j \in |\, I \,| : R(C(A, j)) > R(C(\text{ATTACKER}, j))$$

[1] Computation Tree Logic (CTL)
[2] Binary Decision Diagram (BDD)
[3] Alphabet

که در آن ATTACKER عامل مهاجم و مجموعه‌ی I مطابق تعریف پیوست ۲ است. توابع R و $\widetilde{C}$ توسط کاربر و در حقیقت خارج از مدل ما فراهم می‌آیند، از این روست که اطلاع از برخی جزئیات پیاده‌سازی و حجم محاسبات یا حافظه‌ی مصرفیِ قدم‌های پروتکل ضروری به نظر می‌رسد.

مدل توافق سه جانبه

در این مدل دو بازیگر (نه لزوماً درستکار) و یک کانال ارتباطی وجود دارد. توصیف آنها در پیوست ۳ آمده است. در این توصیف network نماینده‌ی کانال ارتباطی و player نماینده‌ی یک پروسه‌ی بازیگر پروتکل است. فرض کرده‌ایم که روی کانال بسته‌های syn، syn/ack و ack ردوبدل شده و یا کانال خالی (nul) است. این کانال بین پروسه‌ی ۱ (بازیگر عادی) و پروسه‌ی ۲ (مهاجم) مشترک است. جدول ۲ حالاتی را که هر کدام از پروسه‌ها می‌توانند در آن قرار گیرند خلاصه می‌کند.

جدول ۲- خلاصه‌ی حالات پروسه‌ها در توافق سه جانبه

حالت	توضیح
idle	بیکار، منتظر
Synsnt	بسته‌ی syn را فرستاده است.
synrcvd	بسته‌ی syn را دریافت کرده است.
synackd	بسته‌ی syn/ack را فرستاده است.
synackrcvd	بسته‌ی syn/ack را دریافت کرده است.
synackackd	بسته‌ی ack را فرستاده است.
ackrcvd	بسته‌ی ack را دریافت کرده است.

همانطور که مدل پیوست ۳ نشان می‌دهد، پروسه‌ی بازیگر در حالت "بیکار" آغاز می‌شود و به طور نامعین[1] ممکن است یک بسته‌ی syn بفرستد. فرض می‌شود که پروسه‌ای که یک جلسه را آغاز می‌کند بر فرستادن بسته‌های syn اصرار می‌کند (در عمل هم تقریباً همین اتفاق میافتد [۱۲۲]). بسته‌ای که پروسه روی کانال می‌فرستد بر اساس حالت آن تعیین می‌شود که در توصیف به صورت واضح آمده-است. به این ترتیب امکان مدل سازی تأخیر تولید بسته‌ها هم وجود دارد.

متغییر مصرف (consume) نشان دهنده‌ی گذار پروسه به حالات مصرف کننده‌ی منابع (در این مورد، حافظه) است. در حقیقت این متغییر مدل کننده‌ی تابع "مصرف منابع" است. تنها حالات idle، synsnt و synrcvd ساختار حافظه (کنترل نشده) را تشکیل نمی‌دهند (تابع R فقط برای این حالات صفر است). ایده این است که اگر پروسه‌ی آغازگر (مهاجم) به حالات مصرف کننده‌ی منابع نمی‌رود نباید بتواند پاسخ‌دهنده را به حالات مصرف کننده‌ی منابع بفرستد (تعریفِ یک حمله). در حقیقت این امکان وجود دارد که فرض کنیم مهاجم هم به حالات مصرف کننده‌ی حافظه می‌رود (توافق سه جانبه به پایان می‌رسد) و به هر حال منابع پاسخ‌دهنده را تلف می‌کند. اما این دیدگاه علاوه بر ایجاد مشکلات امنیتی برای مهاجم (مانند ردیابی)، از نوع حملات عدم سرویس دهی به نوع حملات عدم سرویس دهی توزیع شده[2] منتقل می‌شود که از دامنه‌ی این مطالعه خارج است. و از اینجاست که مدل مهاجم به شکل زیر توصیف می‌شود.

مدل مهاجم

مهاجم بدون حافظه تصور می‌شود، در نتیجه هیچگاه نمی‌تواند وارد حالات مصرف کننده‌ی منابع شده و بسته‌های SYN/ACK و ACK را تولید کند. تنها حالات ممکن برای مهاجم idle، synsnt، synrcvd و ackrcvd هستند که حالات synrcvd و ackrcvd هم روی idle تصویر می‌شوند

[1] Nondeterministic
[2] Distributed Denial of Service (DDoS)

(چراکه مهاجم تنها می‌تواند بسته‌ی SYN تولید کند و توانایی دنبال کردن دیگر بسته‌ها را ندارد). متغییر *مصرف* هم برای مدل کردن استفاده از حافظه آمده که البته طبق مدل همواره صفر است. مهاجم بـه طـور نامعیین می‌تواند بین حالات بیکار و تولید SYN تغییر وضعیت دهد.

حمله‌ی SYN

همان طور که در پیوست ۳ آمده، یک پروسه‌ی مهاجم و یک پروسه‌ی بازیگر به صـورت همزمـان اجـرا شده و از طریق کانال با هم ارتباط دارند. ابتدا بررسی می‌کنیم که آیا امکان دارد بازیگر بعـد از رفـتن بـه حالت synrcvd قفل شده و هرگز به حالت idle بازنگردد یا خیر. مشخصه‌ی زیـر بیـانگر ایـن مفهـوم است:

```
AG (proc1.state=synrcvd -> AF (proc1.state=idle))
```

نتیجه‌ی این وارسی مدل در پیوست ۳ آمده است. واضحاً این خاصیت برقرار نیست و حمله‌ای که SMV پیشنهاد می‌کند مفهوم حمله‌ی SYN را در خود دارد: پروسه‌ی مهاجم یک بسته‌ی SYN فرستاده و به حالت idle می‌رود در حالی که پروسه‌ی پاسخ دهنده به حالت synrcvd رفته و در حالت synackd قفل می‌شود. آنچه که در نهایت برای ما اهمیت دارد برقراری مشخصه‌ی زیر است:

```
AG (proc1.state=synrcvd -> A[!(proc1.consume &
!proc2.consume) U proc1.state=idle])
```

به این معنی که هرگاه پروسه‌ی بازیگر به حالت synrcvd رفت، برای تمام مسیرهای ممکن این پروسه به حالت idle بازگشته و قبل از آن وارد حالات مصرف کننده‌ی منابع هم نشود. اما از آنجا که طبق پیوست ۳، بازگشت به حالت idle تضمینی ندارد مشخصه‌ی فوق با مشخصه‌ی زیر هم ارز است:

```
AG (proc1.state=synrcvd -> AG !(proc1.consume &
!proc2.consume))
```

این موضوع با مقایسه‌ی پیوست‌های ٤ و ٥ روشن می‌شود. خروجیِ SMV در پیوست ٥ هم حمله‌ی SYN را نشان می‌دهد.

استفاده از SYN Cookie

در حقیقت استفاده از SYN Cookie فقط متغییر consume را بازتعریف می‌کند. با وجود SYN-Cookie پروسه‌ی بازیگر در حالت synackd هم حافظه‌ای را اختصاص نمی‌دهد؛ در نتیجه، در این حالت هم Consume=0 است. پیوست ٦ مدل جدید و نتیجه‌ی ارزیابی آن را نشان می‌دهد. در این شرایط مشخصه‌ی

```
AG (proc1.state=synrcvd -> AG !(proc1.consume &
!proc2.consume))
```

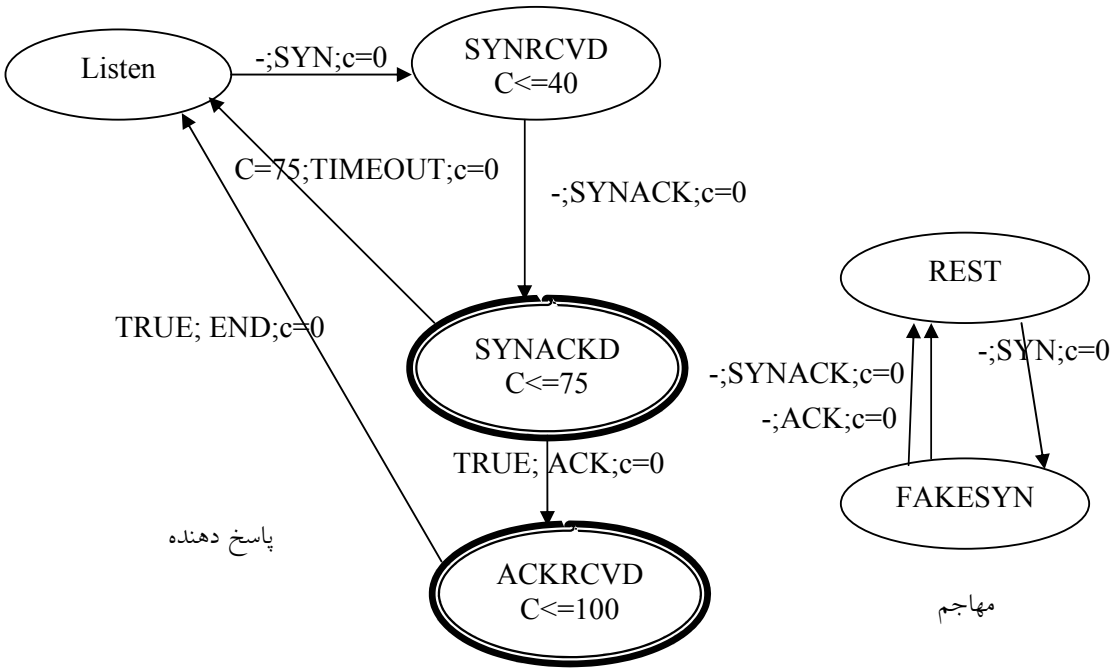
برقرار بوده و پروتکل در برابر حمله‌ی SYN آسیب پذیر نیست.

فرضی که به طور ضمنی در تحلیل این نتایج وجود دارد، زمان‌دار بودنِ حالات در پیاده‌سازی واقعی پروتکل TCP/IP است (زیربخش بعد و پیوست ۷ را ببینید). در حقیقت، ماشین حالتی که در سیستم عامل برای پیاده‌سازی پشته‌ی[1] TCP به کار می‌رود تا ابد در حالت synackd نمی‌ماند و بعد از زمان مشخصی (۷۵ ثانیه در لینوکس [٤٦]) به حالت idle بر می‌گردد (در صورت استفاده از SYN Cookie اصولاً چنین حالتی وجود نداشته و پروتکل تنها در برابر دریافت بسته‌های ACK حاوی Cookie است که تشکیل ساختار حافظه –حالت– داده و بسته‌های دریافتی را دنبال می‌کند). از این نظر تفاوت مدل‌های پیوست ۵ و ٦ فقط در سطح مصرف منابع است (در حقیقت سیستم عامل‌ها تعداد محدودی – حدود ۱۰ [٤٦]– نشست نیمه‌باز[2] را قبول می‌کنند، از این روست که هدر رفتن این تعداد توسط تقاضاهای جعلی مشکل‌ساز می‌شود). به این ترتیب مدل کردن منابع مصرفی و تولید قضیه بر اساس آن اهمیت پیدا کرده و روشنگر نقطه‌ی اصلی آسیب پذیریِ پروتکل است.

[1] TCP Stack
[2] Half-Open Sessions

توصیف و وارسی مدل بلادرنگ[1]

همانطور که در زیربخش‌های قبل دیدیم، در مدل‌سازی پروتکل TCP/IP مفهوم زمان را حذف کرده بودیم چرا که ابزار وارسی مدل SMV توانایی وارسی زمانی را نداشت. در این زیر بخش سعی می‌کنیم با استفاده از KRONOS یک بررسی دقیق از مرحله‌ی توافق سه جانبه در TCP/IP را ارائه کنیم. KRONOS جزو معدود ابزار وارسی مدل است که توانایی وارسی مشخصه‌های در قالب TCTL (یا CTL زماندار[2]) با خودکاره‌ی زمان‌دار[3] برای تجرید مدل را دارند [۱۷]. در این مطالعه از KRONOS نسخه‌ی ۲,۵۲ برای سیستم عامل MS Windows NT® استفاده شده است که جزئیات بیشتر در مورد ساختار نحوی و معنایی آن در [۱۳۷] آمده است. شکل ٤ مدل پاسخ دهنده و مهاجم را نشان می‌دهد. پیوست ۷ هم شامل توصیف کامل این مدل و دستورات مربوطه برای وارسی است.



شکل ٤- مدل بلادرنگ توافق ۳ جانبه

[1] Real-time Specification and Model Checking
[2] Timed CTL
[3] Timed Automata

در مدل زماندار توافق سه جانبه (شکل ٤)، هر گذار دارای ۳ مشخصه است: شرط زمانی، مجموعه‌ی رویدادها و زمان‌هایی که مقداردهی می‌شوند. برای مثال گذار c<=40;SYN;c=0 بیان می‌کند که اگر زمان‌سنج c کمتر از ٤٠ باشد آنگاه رویداد SYN اتفاق میافتد، زمان‌سنج c صفر شده و سیستم به حالت بعدی منتقل خواهد شد. وجود شرط زمانی در داخل حالت‌ها مشخصه‌ی ناوردا[1] نامیده می‌شود و تنها در صورت برقراری آن شرط است که سیستم می‌تواند در آن حالت باقی بماند [۱۳۷].

خاصیت‌های زیر از مدل فوق وارسی شدند که نشان دهنده‌ی وجود حمله‌ی SYN است و امن بودن پروتکل با استفاده از SYN Cookie است (ساختار نحوی KRONOS مختصراً در پیوست ۷ آمده است)

۱- آیا هر حالت SYNRCVD یک حالت LISTEN را به دنبال دارد؟ بله.

```
init imp lab (SYNRCVD impl (ad LISTEN))
```

با وارسی این مشخصه، دیگر نیازی به فرض قفل شدن پروتکل (مانند مدل بدون زمان) نداریم.

۲- آیا حالت SYNACKD دسترس پذیر است؟ بله.

```
init impl (ed SYNACKD)
```

این مشخصه بیانگر امکان رسیدن به حالت SYNACKD با وجود مهاجم نامعین است. در صورتی که SYN Cookie استفاده نشده باشد رسیدن به این حالت یک حمله به شمار می‌رود.

۳- آیا حالت ACKRCVD دسترس پذیر است؟ خیر.

همانطور که انتظار داریم این حالت دست نایافتنی است. در حقیقت پروتکل بدون ردوبدل شدنِ ACK ادامه داده و هیچگاه این بسته تولید نمی‌شود.

[1] Invariant Property

با توجه به حالات دوخط (نماینده‌ی حالاتی که R برای آن‌ها ۱ است) در شکل ٤، وجود حمله‌ی SYN توسط مشخصه‌ی زیر بیان می‌شود:

```
init impl ((ed SYNACKD) or (ed ACKRCVD))
```

KRONOS درستی این عبارت را نشان می‌دهد که اثبات کننده‌ی وجود حمله‌ی SYN است.

در صورت استفاده از SYN Cookie حالات دوخط تنها شامل ACKRCVD می‌شود و وجود حمله توسط عبارت زیر بیان می‌شود:

```
init impl (ed ACKRCVD)
```

که برقرار نبوده و امن بودن پروتکل بعد از استفاده از SYN Cookie را نتیجه می‌دهد.

متأسفانه ابزار وارسی مدل زمانی هنوز در مراحل تحقیقاتی بوده و در بسیاری از موارد مستندسازی[1] نشده‌اند؛ از این رو پیوست ۷، دستورات لازم جهت وارسی مدل را هم ذکر کرده‌است.

در این مطالعه چه آموختیم؟

در این مطالعه نمونه‌ای از درستی‌یابی امنیت یک پروتکل در برابر حملات عدم سرویس‌دهی در ساختار مصرف منابع را دیدیم. امکان استفاده از ابزار وارسی مدل برای پیاده‌سازی این مفاهیم، نقطه‌ی مثبتی است که زمینه‌ی کاربرد فراگیر آنها را فراهم می‌آورد.

نکات زیر نیز به لحاظ دیدگاه ویژه‌ای که برای حالت کلی مسأله به ما می‌دهند، دارای اهمیت می‌باشند:

۱- مدل‌های معمول در ارزیابی محرمانه بودن و تصدیق هویت در مورد سرویس‌دهی پایدار چندان مناسب نیستند. هر چند [۹٤] تغییرات جزئی در ابزار کنونی را برای این منظور کافی می‌داند اما مدل کردن و تحلیل منابع مصرفی نیازمند توجه بیشتر است.

۲- از آنجا که موضوع منابع مصرفی در لایه‌ی پیاده سازی تعیین می‌شود مدل سازی پروتکل، توصیف مشخصه‌ی مورد بررسی و تحلیل نتایج نیازمند دانش بیشتر در مورد جزئیات پیاده‌سازی پروتکل است.

[1] Documentation

در نتیجه یک پیش مرحله‌ی مجردسازی ضروری به نظر می‌رسد. به ویژه که در بسیاری از موارد راه‌حل-های حملات عدم سرویس دهی هم در لایه‌ی پیاده‌سازی توصیه می‌شوند. از طرفی مدل Dolev-Yao عملگرهای رمزنگاری را به صورت جعبه‌ی سیاه[1] در نظر می‌گیرد، از این رو گنجاندن مفاهیم مصرف منابع در این مدل چندان ساده به نظر نمی‌رسد.

[1] Black Box

۷- نتیجه گیری

در این پایان‌نامه حملات عدم سرویس‌دهی به عنوان نوع متفاوتی از حملات امنیتی، مورد مطالعه قرارگرفتند. در این راستا، مشاهده شد که مدل‌های متداول برای پروتکل‌های رمزنگاری که معمولاً برای مشخصات محرمانه‌بودن و تصدیق هویت طراحی شده‌اند، چندان مناسب نیستند. از این رو، یک مدل مبتنی بر مصرف منابع ارائه شد که برای وارسی مدلِ مرحله‌ی توافق سه‌جانبه‌ی پروتکل TCP/IP در برابر حمله‌ی SYN به کار رفت. از آنجا که بسیاری از حملات عدم سرویس‌دهی به نحوه‌ی پیاده‌سازی پروتکل هم بستگی دارند، مدل منابع مصرفی به عنوان یک ساختار صوری برای تجرید پیاده‌سازی مناسب به نظر می‌رسد. به هر حال، باز هم نقش عامل انسانی در مدل کردنِ نحوه‌ی پیاده‌سازی یک پروتکل حیاتی است. شاید اگر نرم افزار مربوطه با قالب‌های رسمی توصیف شده‌باشد بتوان معیاری برای حجم محاسبه یا حافظه‌ی آن یافته و بخش بزرگتری از وارسی پروتکل را به کمک رایانه انجام داد. در مجموع، از فصل‌های پیشین بر می‌آید که جامعه‌ی تحقیقاتی هنوز در مرحله‌ی یادگیری در مورد درستی‌یابی صوری پروتکل‌های رمزنگاری است و هنوز بررسی پروتکل‌های بزرگ تجارت الکترونیک غالباً بسیار زمان بر و دور از دسترس به نظر می‌رسد. اما پیشرفت‌های نظری و مهندسی این زمینه هم

چشمگیر بوده‌اند، تقریباً هرچه در دنیای درستی‌یابی صوری وجود داشت در مورد پروتکل‌های امنیتی آزمایش شد، جرقه‌های جدیدی زده شده که برای دیگر زمینه‌های درستی‌یابی صوری هم تازه بود و انواع ابزار برای این منظور توسعه یا بهبود یافته اند. هنگامی که هدف، چیزی نظیر *تضمین امنیت* است، باید هم راه سختی را برای آن انتظار داشت.

خلاصه ماجرا از دیدگاه نظری به شکل زیر است:

- تعدادی ساختار صوری با دستگاه‌های معنایی متفاوت داریم که برای درستی‌یابی یک هدف ویژه طراحی شده اند.
- این ساختارهای صوری در حالت کلی تصمیم پذیر نیستند.
- یک راهِ گذشتن از سد تصمیم ناپذیری، روش‌های تقریبی خودکار با تضمین همگرایی است.
- راه دیگر، استفاده از شیوه‌های است که تمام بوده اما خودکار نیستند.
- در این میان ساختارهای معنایی کمک می‌کنند تا ارتباطِ بینِ ساختاریِ بهتری برقرار شود و از طرف دیگر ایده‌های تسریع اثبات و روش‌های تقریبی از آنها نشأت می‌گیرند.

آنچه که در پیش روست، بهبود ابزار، توسیع مفاهیم به خواص امنیتی دیگر علاوه بر محرمانه بودن و تصدیق هویت نظیر عامل‌ناشناس[1]، پایداری در مقابل حملات عدم سرویس‌دهی[2] و تکذیب ناپذیری[3] و همچنین گسترش کاربرد و تجربه‌ی درستی‌یابی در پروتکل‌های بزرگ تجارت الکترونیک است. [۹۲]

[1] Anonymity
[2] DoS Survivability
[3] Irrefutability

پیوست ۱: مهاجم Dolev-Yao تواناترین مهاجم است.

شاید ادعای "تواناترین مهاجم" قدری غیر واقعی به‌نظر برسد، در این پیوست طرح اثبات "مهاجم Dolev-Yao تواناترین مهاجم است" را که در [۳۳] آمده می‌بینیم. مراحل اثبات به این صورت است:

- قدم‌های پروتکل در قالب بازنویسی چند مجموعه‌ای مدل می‌شوند.
- تواناترین مهاجم به این معنی است: مهاجمی که در قالب بازنویسی چند مجموعه‌ای بیان شود و درضمن قوانین "کنترل دسترسی" را (که تعریف می‌شوند) نقض نکند؛ بجز این قوانین محدودیت دیگری بر تواناترین مهاجم وجود ندارد.
- مهاجم Dolev-Yao در قالب بازنویسی چند مجموعه‌ای بیان می‌شود.
- اثبات می‌شود که هر اجرایِ پروتکل که در برگیرنده‌ی تواناترین مهاجم باشد، به اجرایی معادل با شرکت مهاجم Dolev-Yao قابل ترجمه است.

بررسی دقیق این مراحل از حوصله‌ی این پیوست خارج بوده و حتی مرجع اصلی این قضیه ([۳۳]) بسیاری از لِم‌های مورد نیاز را به گزارش‌های دیگر ارجاع داده است؛ از این رو در ادامه *طرح* اثبات این قضیه را می‌بینیم.

۱- نمایش پروتکل در قالب بازنویسی چند مجموعه‌ای

هر قدم پروتکل به صورت یک قانون بازنویسی بیان می‌شود:

$$F_1,...,F_k \to \exists x_1...\exists x_j.G_1,...,G_n$$

به این معنی که اگر بازیگر در حالتی باشد که شامل $F_1,...,F_k$ است، یک حالت بعدی متصور برای آن بازیگر حالتی است که $F_1,...,F_k$ از آن حذف شده، $G_1,...,G_n$ اضافه شده و برای متغییرهای $x_1,...,x_j$ مقادیر جدیدی اختیار می‌شوند. معمولاً عامل شبکه (N) هم در پروتکل فرض می‌شود که نشان دهنده‌ی کانال ارتباطی بین بازیگرهاست. به عنوان مثال برای نمایش یک مرحله از پروتکل که عامل A مقدار تصادفی N را تولید کرده و روی شبکه می‌فرستد، قانون زیر به کار می‌رود:

$$A_0() \to \exists x.A_1(x), N_1(x)$$

آنچه که در پرانتزها می‌آید بیانگر دانش عامل‌ها در حالت مربوطه هستند. زیرنویس‌ها هم حالت عامل را نشان می‌دهند (A_0 نشان دهنده‌ی عامل A در حالت صفر است).

۲- تواناترین مهاجم

قوانین کنترل دسترسی‌ای وجود دارند که تنها محدودیت برای یک مهاجم در قالب بازنویسی چند مجموعه‌ای به‌شمار می‌آیند. برای مثال:

$$\frac{\Sigma\,|\alpha\;\; P,(\Sigma,A:\mathrm{Principal})\,|\alpha\;_A\,\rho}{\Sigma\,|\alpha\;\; P,\rho^{\forall A}}$$

بیان می‌کند که اگر یک پروتکل مانند P قوانین کنترل دسترسی را نقض نکند ($\Sigma\,|\alpha\;\; P$) و قانون ρ هم قوانین کنترل دسترسی با وجود عامل A را نقض نکند ($\Sigma\,|\alpha\;_A\,\rho$)، می‌توان نتیجه گرفت که توسعه‌ی

پروتکل P با قانون ρ به ازای هر عامل جدید قوانین کنترل دسترسی را نقض نخواهد کرد. فهرست کامل این قوانین در [۳۳] آمده است.

۳- مهاجم Dolev-Yao در قالب بازنویسی چند مجموعه‌ای

به سادگی می‌توان توانایی‌های مهاجم Dolev-Yao را به صورت قانون‌های بازنویسی بیان کرد. برای مثال قانون زیر بیانگر فرض "مهاجم تمام پیام‌های رد و بدل شده را دریافت کرده و ذخیره می‌کند" است.

$$\forall t : \text{msg}.\ N(t) \rightarrow I(t)$$

که msg نوع پیام، N عامل شبکه و I عامل مهاجم است. فهرست کامل این قوانین در [۳۳] آمده است.

٤- مهاجم Dolev-Yao تواناترین مهاجم است.

فرض کنید استنتاج زیر هیچکدام از قوانین کنترل دسترسی را نقض نمی‌کند.

$$P > [S]_{\Sigma}^{R} \rightarrow^{*} [S']_{\Sigma'}^{R'}$$

در این طرز نمایش R (R') نشان دهنده‌ی مجموعه‌ی بازیگرها، S (S') مجموعه‌ی حالات بازیگرها و P پروتکل مربوطه است. Σ,Σ' هم بیانگر درستی عبارات از نظر نحوی هستند. ادعا این است که همواره یک ترجمه وجود دارد که طی آن: (۱) R، R' و P به مجموعه‌هایی با بازیگرانی با حداکثر توان عامل مهاجم Dolev-Yao تبدیل می‌شوند (۲) S و S' به مجموعه حالاتی تبدیل می‌شوند که اعضای آنها با قوانین تولید حالت Dolev-Yao هم‌خوانی دارند (۳) استنتاج $S \rightarrow^{*} S'$ بعد از ترجمه هم درست است. به این ترتیب هر مهاجمی در قالب بازنویسی چند مجموعه‌ای به مهاجم Dolev-Yao ترجمه می‌شود.

پیوست ۲: ناامن بودنِ پروتکل با تعداد کراندار نشست، NP-تمام است.

در این پیوست طرح اثبات NP-تمام بودنِ مسأله‌ی ناامنی پروتکل با تعداد کراندار نشست در مدل Dolev-Yao را می‌بینیم که برگرفته از [۱۱۳] است.

۱- نمایش پروتکل در قالب بازنویسی چند مجموعه‌ای

همانند پیوست ۱، داریم:

$$P = \{(l, R_l \Rightarrow S_l) \mid l \in I\}$$
$$I = \{(A,i) \mid A \in Names, i \in W_A\}$$

P نشان دهنده‌ی یک پروتکل و $R_l \Rightarrow S_l$ قوانین بازنویسی هستند. هر l یک بازیگر (مانند A) و یک شماره (مانند i) از ترتیب حالات آن بازیگر (W_A) است. یک اجرای درست از پروتکل به صورت زیر تعریف می‌شود:

$$\pi : I \to \{1,...,|I|\}$$
$$\forall A \in \text{Names}, i <_{W_A} j : \pi(A,i) < \pi(A,j)$$

۲- تصمیم گیری در مورد وجود حمله علیه یک پروتکل، عضو NP است.

اگر پروتکل P به صورت $P = \{R'_l \Rightarrow S'_l \mid l \in I\}$ داده شده و یک پیام محرمانـه (Secret) تعریـف شده باشد، با فرض S_0 به عنوان دانش اولیه‌ی مهاجم، "حمله" اجرای درستی مانند π است که:

$$\pi : I \to 1,...,k$$
$$\forall i = 1...k :$$
$$R_i\sigma \in synth(analz(S_0, S_1\sigma,..., S_{i-1}\sigma))$$
$$\text{Secret} \in synth(analz(S_0, S_1\sigma,..., S_k\sigma))$$
$$R_i = R'_{\pi^{-1}(i)}, S_i = S'_{\pi^{-1}(i)}$$

در این نمایش σ یک جایگزینی بـرای متغییرهـای پروتکـل اسـت ([۱۱۳] از تـابع Forge بجـای synth(analz) استفاده کرده، از آنجا که در بخش‌هـای قبـل توابـع synth و analz را دیـده‌ایـم، نیازی به تعریف Forge نیست). حال اثبات این موضوع که وجود حمله عضـو NP اسـت چنـدان پیچیده نمی‌نماید: کافی است که یک اجرای درست حدس زده و بررسـی کنـیم آیـا Secret فـاش خواهد شد یا نه (اثبات چند جمله‌ای بودنِ زمان[1] اجرای چنین الگوریتمی در [۱۱۳] آمده است).

۳- تصمیم گیری در مورد وجود حمله علیه یک پروتکل، عضو NP-سخت است.

یک کاهش از مسأله‌ی ۳-SAT (یک مسأله‌ی NP-تمام [۱۱۹]) به مسأله‌ی تصمیم گیری در مـورد وجود حمله علیه یک پروتکل وجود دارد که نشان می‌دهد این مسأله NP-سخت است.

مسأله‌ی ۳-SAT زیر را در نظر بگیرید:

- گزاره‌های $x_1,...,x_n$.
- مسأله‌ی $f(\vec{x}) = \wedge_I (x_{i,1}^{\varepsilon_{i,1}} \vee x_{i,2}^{\varepsilon_{i,2}} \vee x_{i,3}^{\varepsilon_{i,3}})$

که در آن $\varepsilon_{i,j} \in \{0,1\}$ و $x^0 = x, x^1 = \sim x$. تابع g به صورت زیر تعریف می‌شود:

$$g(\varepsilon_{i,j}, x_{i,j}) = \begin{cases} x_{i,j} \text{ if } \varepsilon_{i,j} = 0 \\ \{x_{i,j}\}_K \text{ if } \varepsilon_{i,j} = 1 \end{cases}$$

[1] Polynomial Time

که در آن K یک کلید رمزنگاری متقارن است که از پیش فرض شده‌است. توابع کمکی زیر را در نظر بگیرید:

$$\forall i \in I : f_i(\overset{\cup}{x}) =< g(\varepsilon_{i,1}, x_{i,1}), < g(\varepsilon_{i,2}, x_{i,2}), g(\varepsilon_{i,3}, x_{i,3}) >>$$

نماد <> به معنای به هم چسباندن رشته‌ها است. حال پروتکل زیر را در نظر بگیرید:

عامل A:

$$(A,1) :< x_{1,1},...,x_{n,3} >\Rightarrow \{< f_1(\overset{\cup}{x}), < f_2(\overset{\cup}{x}), < ..., < f_n(\overset{\cup}{x}), end >>>>\}_P$$

عامل B:

$$\forall i : 1 \le i \le | f(\overset{\cup}{x}) |, (B,i) : \{<< T, < x, y >>, z >\}_P \Rightarrow \{z\}_P$$

عامل 'B:

$$\forall i : 1 \le i \le | f(\overset{\cup}{x}) |, (B',i) : \{<< \{\neg T\}_K, < x, y >>, z >\}_P \Rightarrow \{z\}_P$$

عامل‌های C،'C و D،'D: مانند B و 'B برای عبارات

$$< x, < T, y >>, < x, < \{\neg T\}_K, y >>, < x, < y, T >>, < x, < y, \{\neg T\}_K >>$$

عامل E:

$$(E,1) : \{\text{end}\}_P \Rightarrow Secret$$

فرض می‌شود $S_0 = \{K^{-1}, T, \neg T\}$ دانش اولیه‌ی مهاجم است. تنها در صورتی که مسأله‌ی ۳-SAT ذکرشده جواب داشته باشد، مهاجم می‌تواند یک بردار ارضاء کننده‌ی آن را برای عامل A فرستاده و در نهایت توسط عامل E از Secret مطلع شود. از این رو وجود جواب برای مسأله‌ی ۳-SAT قابل کاهش به تصمیم گیری در مورد وجود حمله علیه‌ی پروتکل فوق است.

٤- تصمیم گیری در مورد وجود حمله علیه یک پروتکل، عضو NP-تمام است.

با توجه به نتایج فوق، تصمیم گیری در مورد وجود حمله علیه یک پروتکل، عضو NP-تمام است.

پیوست ۳: توصیف و وارسی مدل بدون SYN Cookie (حالت ۱)

```
MODULE main
      VAR
            network : {syn, ack, synack, nul};
            proc1 : process player(network);
            proc2 : process attacker(network);
      ASSIGN
            init(network) := nul;
      SPEC
            AG (proc1.state=synrcvd -> AF(proc1.state=idle))
MODULE attacker(network)
      VAR
            state:   {idle,   synsnt,   synrcvd,   synackd,   synackrcvd,
            synackackd, ackrcvd};
            consume : boolean;
      ASSIGN
            init(state) := idle;
            next(state) := {idle, synsnt};
            next(network) :=
                  case
                        state=synsnt : syn;
                        1 : nul;
                  esac;
            consume :=
                  case
                        state=idle : 0;
```

```
                        state=synsnt : 0;
                        state=synrcvd : 0;
                        1 : 1;
                  esac;

MODULE player(network)
      VAR
            state:   {idle,   synsnt,   synrcvd,   synackd,   synackrcvd,
            synackackd, ackrcvd};
            consume: boolean;
      ASSIGN
            init(state) := idle;
            next(state) :=
                  case
                        state=idle & network=nul : {idle, synsnt};
                        state=idle & network=syn : synrcvd;
                        state=synsnt & network= synack : {synackrcvd};
                        state=synrcvd : synackd;
                        state=synackd & network=ack : ackrcvd;
                        state=synackd : {idle, synackd};
                        state=synackrcvd : synackackd;
                        state=synackackd : {idle, synackackd};
                        state=ackrcvd: {idle, ackrcvd};
                        1 : state;
                  esac;
            next(network) :=
                  case
                        state=idle : nul;
                        state=synsnt : syn;
                        state=synackd : synack;
                        state=synackackd : ack;
                        1 : nul;
                  esac;
            consume :=
                  case
                        state=idle : 0;
                        state=synsnt : 0;
                        state=synrcvd : 0;
                        1 : 1;
                  esac;
      FAIRNESS
```

```
            running

-- specification AG (proc1.state = synrcvd -> AF proc1.st... is false
-- as demonstrated by the following execution sequence
state 1.1:
network = nul
proc1.state = idle
proc1.consume = 0
proc2.state = idle
proc2.consume = 0

state 1.2:
[executing process proc2]

state 1.3:
[executing process proc2]
proc2.state = synsnt

state 1.4:
[executing process proc1]
network = syn
proc2.state = idle

state 1.5:
network = nul
proc1.state = synrcvd

state 1.6:
[executing process proc1]

-- loop starts here --
state 1.7:
proc1.state = synackd
proc1.consume = 1

state 1.8:
[executing process proc1]

state 1.9:
[executing process proc2]
network = synack
```

```
state 1.10:
network = nul


resources used:
processor time: 0.03 s,
BDD nodes allocated: 2847
Bytes allocated: 1045152
BDD nodes representing transition relation: 146 + 14
```

پیوست ٤: توصیف و وارسی مدل بدون SYN Cookie (حالت ۲)

```
MODULE main
      VAR
            network : {syn, ack, synack, nul};
            proc1 : process player(network);
            proc2 : process attacker(network);
      ASSIGN
            init(network) := nul;
      SPEC
            AG (proc1.state=synrcvd -> A[!(proc1.consume &
            !proc2.consume) U proc1.state=idle])
MODULE attacker(network)
      VAR
            state:   {idle,   synsnt,   synrcvd,   synackd,   synackrcvd,
            synackackd, ackrcvd};
            consume : boolean;
      ASSIGN
            init(state) := idle;
            next(state) := {idle, synsnt};
            next(network) :=
                  case
                        state=synsnt : syn;
                        1 : nul;
                  esac;
            consume :=
                  case
```

```
                        state=idle : 0;
                        state=synsnt : 0;
                        state=synrcvd : 0;
                        1 : 1;
                  esac;

MODULE player(network)
      VAR
            state:   {idle,   synsnt,   synrcvd,   synackd,   synackrcvd,
            synackackd, ackrcvd};
            consume: boolean;
      ASSIGN
            init(state) := idle;
            next(state) :=
                  case
                        state=idle & network=nul : {idle, synsnt};
                        state=idle & network=syn : synrcvd;
                        state=synsnt & network= synack : {synackrcvd};
                        state=synrcvd : synackd;
                        state=synackd & network=ack : ackrcvd;
                        state=synackd : {idle, synackd};
                        state=synackrcvd : synackackd;
                        state=synackackd : {idle, synackackd};
                        state=ackrcvd: {idle, ackrcvd};
                        1 : state;
                  esac;
            next(network) :=
                  case
                        state=idle : nul;
                        state=synsnt : syn;
                        state=synackd : synack;
                        state=synackackd : ack;
                        1 : nul;
                  esac;
            consume :=
                  case
                        state=idle : 0;
                        state=synsnt : 0;
                        state=synrcvd : 0;
                        1 : 1;
                  esac;
```

```
      FAIRNESS
            running


-- specification AG (proc1.state = synrcvd -> A((!(proc1.... is false
-- as demonstrated by the following execution sequence
state 1.1:
network = nul
proc1.state = idle
proc1.consume = 0
proc2.state = idle
proc2.consume = 0

state 1.2:
[executing process proc2]

state 1.3:
[executing process proc2]
proc2.state = synsnt

state 1.4:
[executing process proc1]
network = syn
proc2.state = idle

state 1.5:
network = nul
proc1.state = synrcvd

state 1.6:
[executing process proc1]

state 1.7:
proc1.state = synackd
proc1.consume = 1


resources used:
processor time: 0.04 s,
BDD nodes allocated: 2599
Bytes allocated: 1045152
BDD nodes representing transition relation: 146 + 14
```

پیوست ۵: توصیف و وارسی مدل بدون SYN Cookie (حالت ۳)

```
MODULE main
      VAR
            network : {syn, ack, synack, nul};
            proc1 : process player(network);
            proc2 : process attacker(network);
      ASSIGN
            init(network) := nul;
      SPEC
            AG (proc1.state=synrcvd -> AG !(proc1.consume &
            !proc2.consume))
MODULE attacker(network)
      VAR
            state:   {idle,   synsnt,   synrcvd,   synackd,   synackrcvd,
            synackackd, ackrcvd};
            consume : boolean;
      ASSIGN
            init(state) := idle;
            next(state) := {idle, synsnt};
            next(network) :=
                  case
                        state=synsnt : syn;
                        1 : nul;
                  esac;
            consume :=
                  case
```

```
                        state=idle : 0;
                        state=synsnt : 0;
                        state=synrcvd : 0;
                        1 : 1;
                  esac;

MODULE player(network)
      VAR
            state:   {idle,   synsnt,   synrcvd,   synackd,   synackrcvd,
            synackackd, ackrcvd};
            consume: boolean;
      ASSIGN
            init(state) := idle;
            next(state) :=
                  case
                        state=idle & network=nul : {idle, synsnt};
                        state=idle & network=syn : synrcvd;
                        state=synsnt & network= synack : {synackrcvd};
                        state=synrcvd : synackd;
                        state=synackd & network=ack : ackrcvd;
                        state=synackd : {idle, synackd};
                        state=synackrcvd : synackackd;
                        state=synackackd : {idle, synackackd};
                        state=ackrcvd: {idle, ackrcvd};
                        1 : state;
                  esac;
            next(network) :=
                  case
                        state=idle : nul;
                        state=synsnt : syn;
                        state=synackd : synack;
                        state=synackackd : ack;
                        1 : nul;
                  esac;
            consume :=
                  case
                        state=idle : 0;
                        state=synsnt : 0;
                        state=synrcvd : 0;
                        1 : 1;
                  esac;
```

```
      FAIRNESS
            running

-- specification AG (proc1.state = synrcvd -> AG (!(proc1... is false
-- as demonstrated by the following execution sequence
state 1.1:
network = nul
proc1.state = idle
proc1.consume = 0
proc2.state = idle
proc2.consume = 0

state 1.2:
[executing process proc2]

state 1.3:
[executing process proc2]
proc2.state = synsnt

state 1.4:
[executing process proc1]
network = syn
proc2.state = idle

state 1.5:
network = nul
proc1.state = synrcvd

state 1.6:
[executing process proc1]

state 1.7:
proc1.state = synackd
proc1.consume = 1



resources used:
processor time: 0.016 s,
BDD nodes allocated: 2743
Bytes allocated: 1045152
BDD nodes representing transition relation: 146 + 14
```

پیوست ٦: توصیف و وارسی مدل با SYN Cookie

```
MODULE main
      VAR
            network : {syn, ack, synack, nul};
            proc1 : process player(network);
            proc2 : process attacker(network);
      ASSIGN
            init(network) := nul;
      SPEC
            AG (proc1.state=synrcvd -> AG !(proc1.consume &
            !proc2.consume))
MODULE attacker(network)
      VAR
            state:   {idle,   synsnt,   synrcvd,   synackd,   synackrcvd,
            synackackd, ackrcvd};
            consume : boolean;
      ASSIGN
            init(state) := idle;
            next(state) := {idle, synsnt};
            next(network) :=
                  case
                        state=synsnt : syn;
                        1 : nul;
                  esac;
            consume :=
                  case
```

```
                        state=idle : 0;
                        state=synsnt : 0;
                        state=synrcvd : 0;
                        1 : 1;
                  esac;

MODULE player(network)
      VAR
            state:   {idle,   synsnt,   synrcvd,   synackd,   synackrcvd,
            synackackd, ackrcvd};
            consume: boolean;
      ASSIGN
            init(state) := idle;
            next(state) :=
                  case
                        state=idle & network=nul : {idle, synsnt};
                        state=idle & network=syn : synrcvd;
                        state=synsnt & network= synack : {synackrcvd};
                        state=synrcvd : synackd;
                        state=synackd & network=ack : ackrcvd;
                        state=synackd : {idle, synackd};
                        state=synackrcvd : synackackd;
                        state=synackackd : {idle, synackackd};
                        state=ackrcvd: {idle, ackrcvd};
                        1 : state;
                  esac;
            next(network) :=
                  case
                        state=idle : nul;
                        state=synsnt : syn;
                        state=synackd : synack;
                        state=synackackd : ack;
                        1 : nul;
                  esac;
            consume :=
                  case
                        state=idle : 0;
                        state=synsnt : 0;
                        state=synrcvd : 0;
                        state=synackd : 0;
                        1 : 1;
```

```
                esac;
      FAIRNESS
            running

-- specification AG (proc1.state = synrcvd -> AG (!(proc1... is true

resources used:
processor time: 0 s,
BDD nodes allocated: 945
Bytes allocated: 1045152
BDD nodes representing transition relation: 146 + 14
```

پیوست ۷: توصیف و وارسی مدل بلادرنگ

دو خودکاره‌ی زمانی برای پاسخ دهنده و مهاجم ساخته می‌شوند:

```
-----8<-----server.tg
#locs 4
#trans 5
#clocks c1
#sync SYN ACK SYNACK
loc: 0
prop: LISTEN
invar: TRUE
trans:
TRUE => SYN; c1:=0; goto 1
loc: 1
prop: SYNRCVD
invar: c1<=40
trans:
TRUE => SYNACK; c1:=0; goto 2
loc: 2
prop: SYNAcKD
invar: c1<=75
trans:
c1<=75 => ACK; c1:=0; goto 3
c1=75 => TIMEOUT1; c1:=0; goto 0
```

```
loc: 3
prop: ACKRCVD
invar: c1<=100
trans:
c1=100 => END1; c1:=0; goto 0
-----8<-----attacker.tg
locs 2
#trans 3
#clocks c3
#sync SYN ACK SYNACK
loc: 0
prop: REST
invar: TRUE
trans:
TRUE => SYN; c3:=0; goto 1
loc: 1
prop: FAKESNT
invar: TRUE
trans:
TRUE => SYNACK; c3:=0; goto 0
TRUE => ACK; c3:=0; goto 0
```

حاصل ضرب این خودکاره ها مدل نهایی است که به صورت زیر به دست می‌آید:

```
Kronos –out fake.tg server.tg attacker.tg
```

برای بیان هر مشخصه باید آن را در قالب یک فایل با پسوند tctl ذخیره کرد؛ یکی از مشخصاتی که در مورد هر سیستمی باید بررسی شود، نا زینو[1] بودن (امکان جریانِ نامنقطعِ زمان) است که توسط مشخصه‌ی زیر بیان می‌شود: [۱۳۷]

$$init \Rightarrow \forall\Box\exists\Diamond_{=1}\texttt{true}$$

در ساختار نحویِ KRONOS، ∀ با a، ∃ با e، □ با b و ◊ با d نمایش داده می‌شوند. و زیرنویس (=۱) در کنار ◊ نشان دهنده‌ی یک قدم زمانی می‌باشد (این علائم در منطق‌های وجهی تعریف می‌شوند که در [۵۸] آمده است). نتیجه یک فایل nz.tctl است که فقط شامل مشخصه‌ی زیر است:

```
init impl ab (ed{=1} true)
```

[1] Non-Zeno (Zeno: نام یک فیلسوف یونانی که در مورد تغییرات و زمان متناقض‌نماهای معروفی بیان کرده‌است)

وارسی مدل توسط دستور زیر امکان پذیر است:

```
kronos -v -forw fake.tg nz.tctl
```

که نام nz.tctl با نام فایل حاویِ مشخصه عوض می‌شود. گزینه‌ی forw هم ممکن است با back جایگزین شود که به ترتیب به معنی وارسیِ پیشرو[1] و عقب‌گرد هستند. [۱۳۷]

[1] Forward

مراجع

[1] M. Abadi, P. Rogaway, *Reconciling Two Views of Cryptography*, Journal of Cryptography, Vol.15, No.2, 2002.

[2] M. Abadi, M. R. Tuttle, *A Semantics for a Logic of Authentication*, Proc. 10th ACM Symposium on Principles of Distributed Computing, Montreal, Canada, August 1991.

[3] M. Abadi, A. D. Gordon, *A Calculus for Cryptographic Protocols: The Spi Calculus*, SRC Research Report 149, Digital Equipments Corporation, USA, January 1998.

[4] M. Abadi, A. D. Gordon, *A Bisimulation Method for Cryptographic Protocols*, Nordic Journal of Computing, Vol.5, No.4, winter 1998.

[5] M. Abadi, *Secrecy by Typing in Security Protocols*, Journal of ACM, Vol.46, No.5, September 1999.

[6] M. Abadi, B. Blanchet, *Secrecy Types for Asymmetric Communication*, Theoretical Computer Science, Volume 298, Issue 3, April 2003.

[7] R. Accorsi, D. Basin, L.Vigano, *Towards an Awareness-based Semantics for Security Protocol Analysis*, Proc. Logical Aspects of Cryptographic Protocol Verification, Paris, France, July 2001.

[8] R. Alur, T. A. Henzinger, **Computer Aided Verification**, Draft Book Available at http://www-cad.eecs.berkeley.edu/~tah/cavbook/.

[9] R. M. Amadio, D. Lugiez, *On the Reachability Problem in Cryptographic protocols*, Research Report 3915, INRIA, France, March 2000.

[10] R. M. Amadio, D. Lugiez, V. Vanackere, *On the Symbolic Reduction of Processes with Cryptographic Functions*, Research Report 4147, INRIA, France, March 2001.
[11] D. Aspinall, *Lectures on Computer Security: Formal Approaches*, School of Informatics, University of Edinburgh, Scotland, February 2003.
[12] D. Basin, *Lazy Infinite State Analysis of Security Protocols*, In Lecture Notes in Computer Science Volume 1740, Springer Verlag, 1999.
[13] A. M. Basyouni, S. E. Tavares, *New Approach to Cryptographic Protocol Analysis Using Colored Petri Nets*, Proc. Canadian Conference on Electrical and Computer Engineering, Newfoundland, Canada, May 1997.
[14] G. Bella, L. C. Paulson, *Using Isabelle to Prove Properties of the Kerberos Authentication System*, Proc. DIMACS Workshop on Design and Formal Verification of Security protocols, NJ, USA, September 1997.
[15] G. Bella, F. Massacci, L. C. Paulson, P. Tramontano, *Formal Verification of Cardholder Registration in SET*, In Lecture Notes in Computer Science Volume 1895, Springer Verlag, 2000.
[16] M. Benerecetti, F. Giunchiglia, M. Panti, N. Spalazzi, *A Logic of Belief and a Model Checking Algorithm for Security Protocols*, Technical Report 0001-05, ITC, Italy, January 2000.
[17] B. Berard, M. Bidoit, A. Finkel, F. Laroussinie, A. Petit, L. Petrucci, P. Schnoebelen, P. McKenzie, **Systems and Software Verification: Model-Checking Techniques and Tools**, Springer Verlag, 2001.
[18] S. Berezin, *Model Checking and Theorem Proving: A Unified Framework*, PhD Thesis, School of Computer Science, Carnegie Mellon University, USA, January 2002.
[19] P. Bieber, *A Logic of Communication in Hostile Environment: Preliminary Version*, Proc. The European Workshop on Logic in Artificial Intelligence, Roscoff, 1988.
[20] B. Blanchet, *From Secrecy to Authenticity in Security Protocols*, In Proc. 9th International Symposium on Static Analysis, Springer Verlag, 2002.
[21] C. Bodei, P. Degano, F. Nielson, H. Riis Nielson, *Flow Logic for Dolev-Yao Secrecy in Cryptographic Processes*, Future Generation Computer Systems, Vol.18, No.6, 2002.
[22] D. Bolignano, *An Approach to the Formal Verification of Cryptographic Protocols*, Proc. 3rd ACM Conference on Computer and Communications Security, New Delhi, India, March 1996.

[23] M. Boreale, *Symbolic Trace Analysis of Cryptographic Protocols*, Proc. 28th International Colloquium on Automata, Languages and Programming (ICALP), Crete, Greece, July 2001.
[24] M. Boreale, R. De Nicola, R. Pugliese, *Proof Techniques for Cryptographic Processes*, Proc. 14th Symposium on Logic in Computer Science, Trento, Italy, July 1999.
[25] M. Boreale, M. G. Buscemi, *Experimenting with STA, A Tool for Automatic Analysis of Security Protocols*, Proc. ACM symposium on Applied Computing, Madrid, Spain, 2002.
[26] M. Bozzano, *Ensuring Security through Model Checking in a Logical Environment*, Proc. Workshop on Specification, Analysis and Validation for Emerging Technologies, Paphos, Cyprus, December 2001.
[27] S. H. Brackin, *A HOL Extension of GNY for Automatically Analyzing Cryptographic Protocols*, Proc. 9th IEEE Computer Security Foundations Workshop, County Kerry, Ireland, March 1996.
[28] J. Bryans, S. Schneider, *CSP, PVS and a Recursive Authentication Protocol*, Proc. DIMACS Workshop on Formal Verification of Security Protocols, NJ, USA, September 1997.
[29] M. Burrows, M. Abadi, R. Needham, *A Logic of Authentication*, SRC Research Report 39, Digital Equipment Corporation, USA, 1989.
[30] U. Carlsen, *Using Logics to Detect Implementation-Dependent Flaws*, Proc. 9th Annual Computer Security Applications Conference, Orlando, USA, December 1993.
[31] U. Carlsen, *Generating Formal Cryptographic Protocol Specifications*, Proc. IEEE Symposium on Security and Privacy, CA, USA, May 1994.
[32] I. Cervesato, N. A. Durgin, P. D. Lincoln, J. C. Mitchell, A. Scedrov, *A Meta-notation for Protocol Analysis*, Proc. 12th IEEE Computer Security Foundations Workshop, Mordano, Italy, June 1999.
[33] I. Cervesato, *MSR, Access Control, and the Most Powerful Attacker*, Proc. 6th IEEE Symposium on Logic in Computer Science, MA, USA, June 2001.
[34] I. Cervesato, N. Durgin, J. Mitchell, P. Lincoln, A. Scedrov, *Relating Strands and Multi Set Rewriting for Security Protocol Analysis*, Proc. 13th IEEE Computer Security Foundations Workshop, Cambridge, UK, July 2000.
[35] E. M. Clarke, R. P. Kurshan, *Computer-aided Verification*, IEEE Spectrum, Vol.33, No.6, June 1996.

[36] E. M. Clarke, S. Jha, W. Marrero, *Verifying Security Protocols with Brutus*, ACM Transactions on Software Engineering and Methodology, Vol.9, No.4, October 2000.
[37] E. M. Clarke, S. Jha, W. Marrero, *Using State Space Exploration and a Natural Deduction Style Message Derivation Engine to Verify Security Protocols*, Proc. IFIP Working Conference on Programming Concepts and Methods, NY, USA, June 1998.
[38] E. M. Clarke, S. Jha, W. Marrero, *Partial Order Reductions for Security Protocol Verification*, In Lecture Notes in Computer Science Volume 1785, Springer Verlag, March/April 2000.
[39] T. Coffey, P. Saidha, *Logic for Verifying Public-Key Cryptographic Protocols*, IEE J. Proceedings Computers and Digital Techniques, Vol. 144, Issue 1, January 1997.
[40] H. Comon-Lundh, V. Shmatikov, *Is It Possible to Decide Whether a Cryptographic Protocol Is Secure or Not?*, Journal of Telecommunication and Information Technology, special Issue on Cryptographic Protocol Verification (ed. J. Goubault-Larrecq), Vol. 4, pages 5-15, 2002.
[41] H. Comon-Lundh, V. Cortier, *Security Properties: Two Agents Are Sufficient*, Proc. 12th European Symposium on Programming, Warsaw, Poland, April 2003.
[42] H. Comon-Lundh, V. Cortier, *New Decidability Results for Fragments of First-Order Logic and Applications to Cryptographic Protocols*, Research Report LSV-03-2, CNSR, France, January 2003.
[43] D. Craigen, M. Saaltink, *Using EVES to Analyze Authentication Protocols*, TR-96-5508-05, ORA, Canada, March 1996.
[44] F. Crazzolara, G. Winskel, *Language, Semantics, and Methods for Cryptographic Protocols*, Report RS-00-18, BRICS, Denmark, August 2000.
[45] Daemon9, *IP Spoofing Demystified*, Phrack Magazine, Issue 48, September 1996. (http://www.phrack.org/)
[46] Daemon9, *Project Neptune*, Phrack Magazine, Issue 48, September 1996. (http://www.phrack.org/)
[47] B. De Decker, F. Piessens, *CryptoLog: A Theorem Prover for Cryptographic Protocols*, Proc. DIMACS Workshop on Formal Verification of Security Protocols, NJ, USA, September 1997.
[48] G. Delzanno, *Specifying and Debugging Security Protocols in the hhf Fragment of Intuitionistic Logic – A Case Study*, Proc. 5th International Symposium on Functional and Logic Programming, Tokyo, Japan, March 2001.

[49] G. Delzanno, S. Etalle, *Proof Theory, Transformations, and Logic Programming for Debugging Security Protocols*, In Lecture Notes in Computer Science Volume 2372, Springer Verlag, 2002.
[50] A. H. Dekker, *C3PO: a Tool for Automatic Sound Cryptographic Protocol Analysis*, Proc. 13th IEEE Computer Security Foundations Workshop, Cambridge, UK, July 2000.
[51] G. Denker, J. Meseguer, C. Talcott, *Protocol Specification and Analysis in Maude*, Proc. Workshop on Formal Methods and Security Protocols, Indiana, USA, June 1998.
[52] N. A. Durgin, *Logical Analysis and Complexity of Security Protocols*, PhD Thesis, Department of Computer Science, Stanford University, USA, 2003.
[53] R. Fagin, J. Y. Halpern, *Belief, Awareness, and Limited Reasoning*, Artificial Intelligence, Vol.34, 1988.
[54] W. Fokkink, J. Friso Groote, M. Reniers, **Modeling Distributed Systems**, Draft Book Available at http://www.cwi.nl/~wan/.
[55] H. Ganzinger, R. Nieuwenhuis, P. Nivela, *The Saturate System*, Available at http://www.mpi-sb.mpg.de/SATURATE/Saturate.html.
[56] L. Gong, R. Needham, R. Yahalom, *Reasoning about Belief in Cryptographic protocols*, Proc. IEEE Symposium on Security and Privacy, CA, USA, May 1990.
[57] A. D. Gordon, A. Jeffrey, *Authentication by Typing for Security Protocols*, Technical Report MSR-TR-2001-49, Microsoft Corporation, USA, May 2001.
[58] J. Goubault-Larrecq, I. Mackie, **Proof Theory and Automated Deduction**, Kluwer Academic Publishing, 1997.
[59] J. Goubault-Larrecq, *A Method for Automatic Cryptographic Protocol Verification*, In Lecture Notes in Computer Science Volume 1800, Springer Verlag, 2000.
[60] J. W. Gray, J. McLean, *Using Temporal Logic to Specify and Verify Cryptographic Protocols*, Proc. 8th IEEE Computer Security Foundations Workshop, County Kerry, Ireland, March 1995.
[61] J. Guttman, F. J. Thayer, L. D. Zuck, *The Faithfulness of Abstract Protocol Analysis: Message Authentication*, Proc. 8th ACM Conf. Computer and Communications Security, Pennsylvania, USA, November 2001.
[62] J. D. Guttman, F. J. Thayer, *Authentication Tests and the Structure of Bundles*, Theoretical Computer Science, Vol.238, Issue 2, June 2002.
[63] J. Y. Halpern, R. Pucella, *Modeling Adversaries in a Logic for Security Protocol Analysis*, Proc. Formal Aspects of Security (FASec '02), Royal Holloway University of London, UK, December 2002.

[64] N. Heintze, J. D. Tygar, *A Model for Secure Protocols and Their Compositions*, IEEE Trans. Software Engineering, Vol.22, No.1, January 1996.
[65] J. Herzog, *Computational Soundness of Formal Adversaries*, MS Thesis, Department of Electrical Engineering and Computer Science, MIT, USA, 2002.
[66] A. Huima, T. Aura, *Using a Multimodal Logic to Express Conflicting Interests in Security Protocols*, Proc. DIMACS Workshop on Design and Formal Verification of Security Protocols, NJ, USA, September 1997.
[67] A. Huima, *Efficient Infinite State Analysis of Security Protocols*, Proc. Workshop on Formal Methods and Security Protocols , Trento, Italy, July 1999.
[68] M. Huth, M. Ryan, **Logic in Computer Science: Modeling and Reasoning about Systems**, Cambridge University Press, 2001.
[69] F. Jacquemard, M. Rusinowitch, L. Vigneron, *Compiling and Verifying Security Protocols*, Research Report 3938, INRIA, France, May 2000.
[70] R. A. Kemmerer, *Analyzing Encryption Protocols Using Formal Verification Techniques*, IEEE Journal on Selected Areas in Communications, Vol.7, No.4, May 1989.
[71] V. Kessler, G. Wedel, *AUTLOG – An Advanced logic of Authentication*, Proc. 7th IEEE Computer Security Foundations Workshop, New Hampshire, USA, June 1994.
[72] V. Kessler, H. Neumann, *A Sound Logic for Analyzing Electronic Commerce Protocols*, Proc. 5th European Symposium on Research in Computer Security, Louvain-la-Neuve, Belgium, September 1998.
[73] D. Kindred, *Theory Generation for Security Protocols*, PhD Thesis, School of Computer Science, Carnegie Mellon University, USA, 1999.
[74] P. Krishnan, A. Renaud, *Protocol Specification and Verification of Security Properties: A General Approach*, Proc. 6th New Zealand Formal Program Development Colloquium, New Zealand, August 2000.
[75] G. Leduc, F. Germeau, *Verification of Security Protocols Using LOTOS-method and Application*, Computer Communications, Special Issue on Formal Description Techniques in Practice ,Vol. 23, No. 12, July 2000.
[76] G. Leduc, O. Bonaventure, L. Leonard, E. Koerner, C. Pecheur, *Model-based Verification of a security Protocol for Conditional Access to Services*, Formal Methods in System Design, Vol.14, No. 2, March 1999.
[77] J. Lemon, *Resisting SYN Flood DoS Attacks with a SYN Cache*, Proc. BSDConf on File and Storage Technology, CA, USA, February 2002.

[78] R. W. Lichota, G. L. Hammonds, S. H. Brackin, *Verifying the Correctness of Cryptographic Protocols Using 'Convince'*, Proc. 12th Annual Computer Security Applications Conference, CA, USA, December 1996.

[79] G. Lowe, *Breaking and Fixing the Needham-Schroeder Public-Key Protocol Using FDR*, Software Concepts and Tools, Vol. 17, Pages 93-102, 1996.

[80] G. Lowe, *Towards a Completeness Result for Model Checking of Security Protocols*, Journal of Computer Security, Vol.7, No. 2/3, 1999.

[81] G. Lowe, B. Roscoe, *Using CSP to detect errors in the TMN Protocol*, IEEE Trans. Software Engineering, Vol.23, No.10, October 1997.

[82] G. Lowe, *Casper: A Compiler for the Analysis of Security Protocols*, Proc. 10th IEEE Computer Security Foundations Workshop, MA, USA, June 1997.

[83] W. Mao, *A Structural Operational Modeling of the Dolev-Yao Threat Model*, Report HPL-2002-218, Hewlett-Packard Company, UK, August 2002.

[84] W. Mao, C.Boyd, *Towards Formal Analysis of Security Protocols*, Proc. 6th IEEE Computer Security Foundations Workshop, New Hampshire, USA, June 1993.

[85] W. Mao, *An Augmentation of BAN-Like Logics*, Proc. 8th IEEE Computer Security Foundations Workshop, County Kerry, Ireland, June 1995.

[86] W. Marrero, E. Clarke, S. Jha, *Model Checking for Security Protocols*, Technical Report CMU-CS-97-139, School of Computer Science, Carnegie Mellon University, USA, May 1997.

[87] W. Marrero, E. Clarke, S. Jha, *A Model Checker for Authentication Protocols*, Proc. DIMACS Workshop on Design and Formal Verification of Security Protocols, NJ, USA, September 1997.

[88] N. Marti-Oliet, J.Meseguer, *Rewriting Logic: Roadmap and Bibliography*, Theoretical Computer Science, Volume 285 , Issue 2, August 2002.

[89] A. M. Mathuria, R. Safavi-Naini, P. R. Nickolas, *On the Automation of GNY Logic*, Proc. 18th Australian Computer Science Conference, Glenelg, Australia, February 1995.

[90] A. M. Mathuria, R. Safavi-Naini, P. R. Nickolas, *Some Remarks on the Logic of Gong, Needham and Yahalom*, Proc. International Computer Symposium, Hsinchu, Taiwan, December 1994.

[91] K. L. McMillan, *The SMV System*, The SMV Manual Available at http://www.cs.cmu.edu/~modelcheck/smv.html.

[92] C. Meadows, *Formal Methods for Cryptographic Protocol Analysis: Emerging Issues and Trends*, IEEE Journal on Selected Areas in Communication, Vol.21, No.1, January 2003.

[93] C. Meadows, *The NRL Protocol Analyzer: An Overview*, Journal of Logic Programming, Vol.26, No.2, February 1996.

[94] C. Meadows, *A Formal Framework and Evaluation Method for Network Denial of Service*, Proc. IEEE Computer Security Foundations Workshop, Mordano, Italy, June 1999.

[95] J. Millen, V. Shmatikov, *Constraint Solving for Bounded-Process Cryptographic Protocol Analysis*, Proc. 8th ACM Conference on Computer and Communication Security, Philadelphia, USA, November 2001.

[96] J. Mitchell, A. Ramanathan, A. Scedrov, V. Teague, *A Probabilistic Polynomial-time Calculus for Analysis of Cryptographic Protocols*, Electronic Notes in Theoretical Computer Science, Vol.45, 2001.

[97] J. C. Mitchell, M. Mitchell, U. Stern, *Automated Analysis of Cryptographic Protocols using Murφ*, Proc. IEEE Symposium on Security and Privacy, CA, USA, May 1997.

[98] J. C. Mitchell, V. Shmatikov, U. Stern, *Finite-State Analysis of SSL 3.0*, Proc. 7th USENIX Security Symposium, Texas, USA, January 1998.

[99] D. Monniaux, *Analysis of Cryptographic Protocls Using Logics of Belief: An Overview*, Journal of Telecommunications and Information Technology, Vol.4, Pages 57-67, 2002.

[100] L. E. Moser, *A Logic of Knowledge and Belief for Reasoning about Computer Security*, Proc. 2nd IEEE Computer Security Foundations Workshop, New Hampshire, USA, June 1989.

[101] T. Murata, *Petri Nets: Properties, Analysis and Applications*, Proceedings of IEEE, Vol.77, No.4, April 1989.

[102] D. M. Nessett, *A Critique of the Burrows, Abadi and Needham Logic*, ACM Operating System Review, Vol.24, No.2, April 1990.

[103] A. Nylen, *Modeling and Verification of Authentication Protocols*, MS Thesis, Department of Computer Systems, Uppsala University, Sweden, December 1997.

[104] J. Pang, *Analysis of a Security Protocol in μCRL*, Report SEN-R0201, CWI, The Netherlands, January 2002.

[105] L. C. Paulson, *Proving Properties of Security Protocols by Induction*, Proc. 10th Computer Security Foundations Workshop, MA, USA, June 1997.

[106] L. C. Paulson, *Proving Security Protocols Correct*, Proc. IEEE Symposium on Logic in Computer Science, Trento, Italy, July 1999.

[107] L. C. Paulson, *Inductive Analysis of the Internet Protocol TLS*, ACM Transactions on Computer and System Security, Vol.2, No.3, 1999.

[108] A. Pnueli, *Verification Engineering: A Future Profession*, A.M.Turing Award Lecture, San Diego, USA, 1997.

[109] R. Pucella, *Review of Communicating and Mobile Systems: The π-calculus*, ACM SIGACT News, Volume 31, Issue 4, December 2000.

[110] P. V. Rangan, *An Axiomatic Basis of Trust in Distributed Systems*, Proc. IEEE Symposium on Security and Privacy, CA, USA, April 1988.

[111] L. Ricciulli, P. Lincoln, P. Kakkar, *TCP SYN Flooding Defense*, Proc. Communication Networks and Distributed Systems Modeling and Simulation Conference, CA, USA, January 1999.

[112] A. W. Roscoe, *Modeling and Verifying Key-Exchange protocols Using CSP and FDR*, Proc. 8th IEEE Computer Security Foundations Workshop, County Kerry, Ireland, June 1995.

[113] M. Rusinowitch, M. Turuani, *Protocol Insecurity with Finite Number of Sessions is NP-Complete*, Proc. 14th IEEE Computer Security Foundations Workshop, Novia Scotia, Canada, June, 2001.

[114] D. B. Santhoshi, D. Shreyas, *Automated BAN Analysis of Authentication protocols*, Available at http://www.ics.uci.edu/~sdoshi.

[115] E. Saul, A. Hutchison, *Using GYPSIE, GYNGER and Visual GNY to Analyze Cryptographic Protocols in SPEAR II*, Proc. Conference on Information Security Management & Small Systems Security, Nevada, USA, 2001.

[116] B. Schenier, **Applied Cryptography**, Prentice Hall, 1994.

[117] S. Schneider, *Using CSP for Protocol Analysis: the Needham-Schroeder Public-Key Protocol*, Technical Report CSD-TR-96-14, Department of Computer Science, Royal Holloway University of London, UK, November 1996.

[118] V. Shmatikov, U. Stern, *Efficient Finite State Analysis for Large Security Protocols*, Proc. 11th IEEE Computer Security Foundations Workshop, MA, USA, June 1998.

[119] M. Sipser, **Introduction to the Theory of Computation**, PWS Publishing, 1997.

[120] E. Snekkenes, *Exploring the BAN Approach to Protocol Analysis*, Proc. IEEE Computer Society Symposium on Research in Security and Privacy, CA, USA, 1991.

[121] D. X. Song, *Athena: A New Efficient Automatic Checker for Security Protocol Analysis*, Proc. IEEE Computer Security Foundations Workshop, Mordano, Italy, June 1999.

[122] R. Stevens, **TCP/IP Illustrated: Volume I**, Addison-Wesley Pub Co, 1994.

[123] S. D. Stoller, *A Bound on Attacks on Payment Protocols*, Proc. 16th Annual IEEE Symposium on Logic in Computer Science, MA, USA, June 2001.

[124] P. F. Syverson, P. C. Van Oorschot, *On Unifying Some Cryptographic Protocol Logics*, Proc. IEEE Symposium on Security and Privacy, CA, USA, May 1994.

[125] P. F. Syverson, *Adding Time to a Logic of Authentication*, Proc. 1st ACM Conference on Computer and Communications Security, Virginia, USA, November 1993.

[126] P. Syverson, *The Use of Logic in the Analysis of Cryptographic Protocols*, Proc. 3rd IEEE Computer Security Foundations Workshop, New Hampshire, USA, June 1990.

[127] P. Syverson, *Formal Semantics for Logics of Cryptographic Protocols*, Proc. 3rd IEEE Computer Security Foundations Workshop, New Hampshire, USA, June 1990.

[128] P. Syverson, J.W.Gray, *The Epistemic Representation of Information Flow Security in Probabilistic Systems*, Proc. 8th IEEE Computer Security Foundations Workshop, County Kerry, Ireland, March 1995.

[129] P. Syverson, *Towards a Strand Semantics for Authentication Logic*, Electronic Notes in Theoretical Computer Science, Vol.20, 1999.

[130] A. S. Tanenbaum, **Computer Networks**, 3rd Edition, Prentice Hall, 1996.

[131] F. J. Thayer, J. Herzog, J. D. Guttman, *Strand Spaces: Why Is a Security Protocol Correct?*, Proc. IEEE Symposium on Security and Privacy, CA, USA, May 1998.

[132] F. J. Thayer, J. C. Herzog, J. D. Guttman, *Honest Ideals on Strand Spaces*, Proc. 11th IEEE Computer Security Foundations Workshop, MA, USA, June 1998.

[133] D. Van Dalen, **Logic and Structure**, 3rd Augmented Edition, Springer Verlag, 1997.

[134] P. C. Van Oorschot, *Extending Cryptographic logics of Belief to Key Agreement Protocols*, Proc. 1st ACM Conference on Computer and Communications Security, Virginia, USA, November 1993.

[135] C. Weidenbach, *Towards an Automatic Analysis of Security Protocols in First-Order Logic*, In Lecture Notes in Artificial Intellience Volume 1632, Springer Verlag, 1999.

[136] T. Y. C. Woo, S. S. Lam, *A Semantic Model for Authentication Protocols*, Proc. IEEE Symposium on Security and Privacy, CA, USA, May 1993.

[137] S. Yovine, *Kronos: A Verification Tool for Real-time Systems*, Available at http://www-verimag.imag.fr/TEMPORISE/kronos/

ABSTRACT

Cryptographic protocols should survive hostile environments, intruders, denial of service attacks and so on. In this way, proving a protocol secure is important and hard to achieve regarding *reactive* nature of the protocol itself. Formal verification methods can help us with getting some kind of assurance about the security of reactive facet of protocols. This thesis, at the first part will study theoretical and computational limits of a general formal verifier for security protocols. Then a survey of existing methods will come, categorized in algebraic and logical approaches. Existing solutions mainly consider security in the realm of secrecy and authentication, while *denial of service* attacks are quite common these days. In the second part we devise a novel framework to formally specify and define these kinds of attacks, which could be easily handled by typical model checkers. As a case study, SYN attack against TCP/IP protocol is studied and the correctness of SYN Cookie method (a usual remediation for SYN attack) is formally proved.

Keywords: **Formal Methods, Verification, Cryptographic Protocols, SYN Attack, Real-time Systems.**

Sharif University of Technology

Mathematical Sciences Department

Master of Science Thesis

Formal Methods in Verification Of Cryptographic Protocols

Mohammad Torabi Dashti

Supervisor: Dr. M. Ardeshir

November 2003