دانشگاه صنعتی امیرکبیر

دانشکده مهندسی کامپیوتر و فناوری اطلاعات

گزارش پروژه‌ی درس پروتکل‌های امنیتی

تحلیل صوری پروتکل مبادله‌ی کلید DASS

با استفاده از ابزار Proverif

استاد درس:

دکتر بابک صادقیان

ارائه‌دهندگان:

فائزه نصرآبادی

محمود اقوامی پناه

بهار ۹۶

چکیده

پروتکل‌های امنیتی پروتکل‌هایی هستند که خواسته‌های امنیتی مشخصی را برآورده می سازند. در میان پروتکل‌های مختلف پروتکل‌های تبادل کلید از اهمیت ویژه ای برخوردارند زیرا امکان تبادل محرمانه ی کلید بین طرفین را با وجود یک کانال ارتباطی ناامن ممکن می سازند. ما در این پروژه به تحلیل صوری پروتکل مبادله‌ی کلید DASS با استفاده از ابزار Proverif پرداخته ایم. ما نشان دادیم که این پروتکل هر چهار خواسته‌ی امنیتی محرمانگی، تصدیق اصالت موجودیت ،تازگی و تائید کلید را دارا هست. ضمن آنکه به بررسی دو خواسته‌ی دیگر یعنی توافق ضعیف و non injectivity نیز پرداخته ایم و برآورده شدن آن‌ها را نشان داده‌ایم.

کلمات کلیدی: تحلیل صوری، پروتکل‌های امنیتی،پروتکل تبادل کلید

فهرست مطالب

فصل اول :

مسئله ی تحلیل پروتکل‌های امنیتی

مقدمه

پروتکل‌های ارتباطی و پروتکل‌های امنیتی دو موضوع متفاوت موردبحث هستند. پروتکل‌های ارتباطی نحوه تعامل بین دو یا چند عامل در شبکه را تعریف می‌کنند و هدف آن‌ها انتقال داده‌ها در شبکه است. اما پروتکل‌های امنیتی هدفشان انتقال امن داده‌ها در شبکه ناامن است. پروتکل‌های امنیتی خواسته‌های امنیتی متفاوتی را طلب می‌کنند. عمده اهداف و خواسته‌های امنیتی یک پروتکل بدین ترتیب است:

- محرمانگی[1]: این خواسته درواقع همان جلوگیری از افشای یک پیام است. مهاجم نباید بتواند از محتوای پیام آگاه گردد. در نوع قوی‌تری از آن، هم‌چنین مهاجم نباید بتواند با مشاهده پیام هیچ اطلاعاتی راجع به آن به دست آورد. به‌عنوان‌مثال اگر دو پیام یکسان است مقدار رمز شده آن دو نباید برابر باشد که مهاجم با مشاهده آن دو بفهمد این دو پیام احتمالاً یکسان هستند.
- تصدیق اصالت[2]: شامل دو نوع است:
  - تصدیق اصالت موجودیت
  - تصدیق اصالت پیام
- بی‌نامی[3]: جلوگیری از شناسایی تعدادی رویداد خاص در میان یک مجموعه از رویدادها.
- عدم انکار[4]
- انصاف[5]

بیشتر ابزارهای توصیف صوری[6] پروتکل‌ها، دو خواسته‌ی امنیتی اول را پشتیبانی می‌کنند. ابزار موردبررسی در این گزارش کارهایی نیز درزمینه‌ی بی‌نامی انجام داده است، اما برای دو خواسته امنیتی عدم انکار و انصاف، هنوز ابزاری نیامده است.

در این گزارش در فصل اول به بررسی مسئله تحلیل پروتکل‌های امنیتی پرداخته‌شده است. در فصل دوم به معرفی اجمالی ابزار Proverif پرداخته‌ایم. سپس در فصل سوم در ابتدا به معرفی پروتکل‌های تبادل کلید و خواسته‌های امنیتی مورد انتظار از چنین پروتکل‌هایی پرداخته شده است. پس‌ازآن پروتکل موردبررسی یعنی پروتکل DASS

[1] Secrecy, Confidentially
[2] Authentication
[3] Anonymity
[4] Non-repudiation
[5] Fairness
[6] Formal

1991 معرفی شده است. روند پروتکل و پیام‌های مبادله شده در آن در ادامه موردبررسی قرار گرفته است. در فصل چهارم خواسته‌های امنیتی این پروتکل به‌طور دقیق موردبررسی قرار گرفته است. در فصل پنجم تکه‌هایی از کد به همراه خروجی و بررسی خواسته‌های برآورده شده آمده است.

مسئله تحلیل پروتکل‌ها

در تحلیل پروتکل‌های امنیتی قصد بر آن است که پروتکلی به شیوه‌ای صوری و با استفاده از یک سری ابزارهایی که بدین منظور طراحی‌شده‌اند ازنظر امنیتی موردبررسی قرار داده شوند. روش تحلیل دستی پروتکل‌ها روشی نادقیق است و ممکن است جنبه‌هایی از پروتکل مورد غفلت قرار گیرند. یعنی ممکن است پروتکلی امن تشخیص داده شود، درصورتی‌که واقعاً بدین گونه نیست.

در تحلیل پروتکل‌ها به روش صوری یک ابزار در اختیار داریم که ورودی‌هایی را دریافت کرده و با انجام پردازش‌هایی بر روی این ورودی‌ها خروجی را برای ما تولید می‌کند. ورودی‌های این ابزارها عبارت‌اند از:

- خود پروتکل که با استفاده از زبانی مدل شده است.
- توانمندی‌های مهاجم.
- خواسته‌های امنیتی مورد انتظار.

گام‌های رایج تحلیل صوری پروتکل‌ها

به‌منظور تحلیل یک پروتکل امنیتی چهار گام زیر برداشته می‌شوند:

۱. مدل کردن پروتکل با استفاده از یک زبان مدل‌سازی. زبان‌های مدل‌سازی مختلفی موجود هستند، نظیر:
   - Spi calculus
   - Strand Space
   - Logic

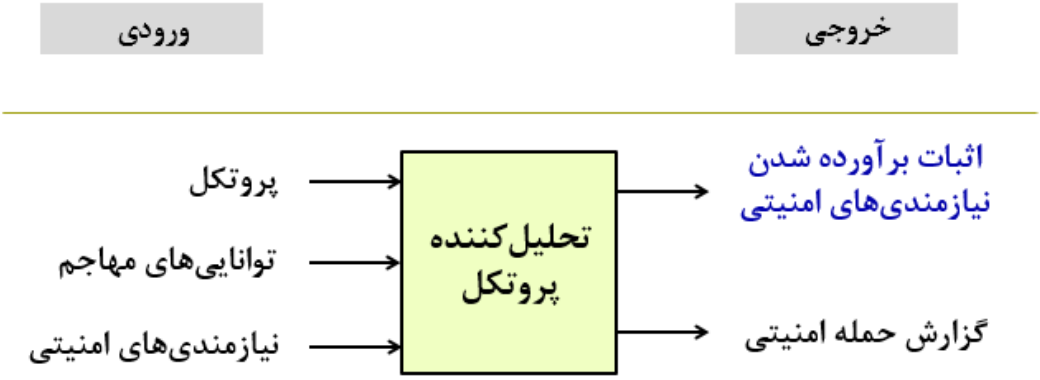
۲. توصیف دقیق خواسته‌های امنیتی و توانمندی‌های مهاجم

۳. استفاده از یک ابزار برای تحلیل خودکار پروتکل:

به‌طورمعمول این ابزار یک موتور استنتاج دارد که روش تحلیل صوری را پیاده‌سازی می‌کند.

۴. ارائه خروجی: که به دو طریق انجام می‌شود:

- اثبات امن بودن این پروتکل و برآورده شدن خواسته‌های امنیتی مورد انتظار از آن.
- گزارش حمله.



روش‌های تحلیل صوری پروتکل‌های امنیتی

این روش‌ها به دودسته تقسیم می‌شوند:

- اثبات‌کننده قضایا[1]
- وارسی گرهای مدل[2]

هرکدام از این دو روش مزیت‌ها و معایبی دارند و به‌طورکلی روشی وجود ندارد که در همه‌ی زمینه‌ها از سایر روش‌ها برتر باشد.

[1] Theorem prover
[2] Model checker

فصل دوم :

معرفی ابزارها

معرفی ابزارهای مختلف

ابزارهای مختلفی برای تحلیل صوری پروتکل‌ها موجود است، نظیر:

- Proverif : که در این گزارش با استفاده از آن تحلیل صوری پروتکل DASS صورت گرفته است. ابزار Proverif در سال ۲۰۰۱ ارائه گردیده است و یک اثبات‌کننده قضایا، مبتنی بر منطق هست. این ابزار یکی از بهترین ابزارهایی است که تابه‌حال درزمینه‌ی تحلیل صوری ارائه‌شده و مورداستفاده قرار می‌گیرد.
- Scyther : این ابزار در سال ۲۰۰۸ ارائه گردید که یک وارسی گر مدل بر مبنای Strand Spaces می‌باشد. استفاده از آن تقریباً نسبت به سایر ابزارها آسان‌تر است.
- Cryptic : این ابزار در سال ۲۰۰۶ ارائه گردید که یک اثبات‌کننده قضایا و وارسی گر مدل است. کار با این ابزار از همه ابزارهای موجود تقریباً دشوارتر است. برای مدل‌سازی پروتکل درواقع فراهم ساختن ورودی تحلیل صوری می‌توان از منطق یا Pi بهره برد.

ما در این گزارش از Pi و ابزار Proverif استفاده کرده‌ایم، به این صورت که مراحل اجرای پروتکل، خواسته‌های امنیتی، توانمندی‌های مهاجم را با استفاده از Pi مدل کردیم و به‌عنوان ورودی به ابزار تحویل دادیم.

ابزار Proverif

ورودی‌های سیستم برای این ابزار قواعد منطقی هستند. به‌طور مثال:

$$\text{taller}(x, y) \wedge \text{taller}(y, z) \Rightarrow \text{taller}(x, z) \qquad (1)$$

$$\text{taller}(A, B) \qquad (2)$$

$$\text{taller}(B, C) \qquad (3)$$

مثلاً پاسخ به این پرسش که آیا می‌توان نتیجه گرفت taller(A,C) ؟ مثبت است زیرا درخت اشتقاق زیر وجود دارد.

$$\dfrac{\dfrac{}{\text{taller}(A, B)}\ \text{by (2)} \qquad \dfrac{}{\text{taller}(B, C)}\ \text{by (3)}}{\text{taller}(A, C)}$$

البته برنامه اثبات‌کننده خودکار، ممکن است در زمان جستجوی DFS در حلقه بی‌نهایت گرفتار گردد. (Non-termination). ابزار proverif هم از این قاعده مستثنا نیست. البته سعی می‌شد تا با استفاده از خلاقیت و ابتکار (Heuristic) در برنامه اثبات‌کننده فقط در حالات بسیار خاصی برنامه گرفتار حلقه‌ی بی‌نهایت شود.

بررسی یک پروتکل ساده

در این پروتکل داریم:

$$B \rightarrow A \quad : \quad \{|s|\}_{\mathrm{enc}(sb)}$$

- B پیام s را با کلید خصوصی خود، enc(sb) ، رمز می‌کند و به سمت A می‌فرستد.
- A با استفاده از کلید عمومی B، dec(sb) ، پیام را ترجمه می‌کند.
- مبادله پیام بر روی کانال ارتباطی net صورت می‌گیرد که در دسترسی همگان ازجمله مهاجم است.
- کلید عمومی B در دسترس همگان ازجمله مهاجم است.

سؤالی که اینجا مطرح می‌شود آن است که آیا مهاجم می‌تواند از محتوای پیام s مطلع شود؟
در جدول زیر برای این پروتکل دانش اولیه مهاجم و اطلاعاتی را که در ادامه تبادل می‌شود را نشان داده‌ایم سپس مرحله‌به‌مرحله گذرهای پروتکل را بررسی کرده‌ایم:

مفهوم، تعبیر	Fact, Rule
مهاجم net را می‌داند. مهاجم به net دسترسی دارد.	$att(net)$
پیام y بر روی کانال x ارسال می‌شود.	$mess(x,y)$
اگر پیام y روی کانال x ارسال گردد و مهاجم به کانال x دسترسی داشته باشد، مهاجم y را می‌داند.	$mess(x,y) \wedge att(x) \Rightarrow att(y)$
اگر مهاجم به کانال x دسترسی داشته باشد و پیام y را بداند، می‌تواند پیام y را روی کانال x ارسال کند.	$att(x) \wedge att(y) \Rightarrow mess(x,y)$
اگر مهاجم به (جفت کلید) x دسترسی داشته باشد، به کلید رمزگذاری/تولید امضای enc(x) نیز دسترسی دارد	$att(x) \Rightarrow att(\mathrm{enc}(x))$
اگر مهاجم به (جفت کلید) x دسترسی داشته باشد به کلید رمزگشایی/وارسی امضای dec(x) نیز دسترسی دارد	$att(x) \Rightarrow att(\mathrm{dec}(x))$
اگر مهاجم پیام x را بداند و به کلید تولید امضای y دسترسی داشته باشد، می‌تواند x را با y امضا کند.	$att(x) \wedge att(y) \Rightarrow att(\mathrm{sign}(x,y))$
اگر مهاجم به پیام رمز/امضا شده x با کلید رمزگذاری enc(y) دسترسی داشته باشد و کلید رمزگشایی dec(y) را بداند، می‌تواند پیام x را استخراج کند.	$att(\mathrm{sign}(x,\mathrm{enc}(y))) \wedge att(\mathrm{dec}(y)) \Rightarrow att(x)$

مدل‌سازی پروتکل:

$$\Rightarrow mess(net, \mathrm{sign}(s, \mathrm{enc}(sb))) \quad (P1)$$

مدل‌سازی توانمندی‌های مهاجم:

$$att(x) \wedge att(y) \Rightarrow mess(x,y) \quad (A1)$$
$$mess(x,y) \wedge att(x) \Rightarrow att(y) \quad (A2)$$
$$att(x) \wedge att(y) \Rightarrow att(\mathrm{sign}(x,y)) \quad (A3)$$
$$att(x) \Rightarrow att(\mathrm{enc}(x)) \quad (A4)$$
$$att(x) \Rightarrow att(\mathrm{dec}(x)) \quad (A5)$$
$$att(\mathrm{sign}(x,\mathrm{enc}(y))) \wedge att(\mathrm{dec}(y)) \Rightarrow att(x) \quad (A6)$$

مدل‌سازی دانش اولیه مهاجم:

$$\Rightarrow att(net) \quad (I1)$$
$$\Rightarrow att(\mathrm{dec}(sb)) \quad (I2)$$

حالا باید ببینیم که آیا att(s) برقرار است یا خیر؟ یعنی درواقع آیا مهاجم s را می‌داند؟

در تصویر زیر کلیه عبارات فوق به‌صورت یکجا آمده است:

$$\Rightarrow att(net) \quad (I1)$$
$$\Rightarrow att(\mathrm{dec}(sb)) \quad (I2)$$
$$att(x) \wedge att(y) \Rightarrow mess(x,y) \quad (A1)$$
$$mess(x,y) \wedge att(x) \Rightarrow att(y) \quad (A2)$$
$$att(x) \wedge att(y) \Rightarrow att(\mathrm{sign}(x,y)) \quad (A3)$$
$$att(x) \Rightarrow att(\mathrm{enc}(x)) \quad (A4)$$
$$att(x) \Rightarrow att(\mathrm{dec}(x)) \quad (A5)$$
$$att(\mathrm{sign}(x,\mathrm{enc}(y))) \wedge att(\mathrm{dec}(y)) \Rightarrow att(x) \quad (A6)$$
$$\Rightarrow mess(net, \mathrm{sign}(s,\mathrm{enc}(sb))) \quad (P1)$$

برای پاسخ به سؤال بالا باید درخت انشقاق att(s) تشکیل شود و از روی آن بررسی شود که آیا به این عبارت می‌رسیم یا خیر. برای این منظور در ابتدا بررسی می‌شود که آیا att(s) جز fact های مسئله است یا خیر اگر نبود، به قواعد دیگر نگاه می‌شود و سعی می‌شود با استفاده از unifier هایی به هدف که att(s) است، برسیم. در انتها درخت انشقاق زیر حاصل می‌شود.

$$\dfrac{\dfrac{}{mess(net, \mathrm{sign}(s,\mathrm{enc}(sb)))}\,(P1)}{att(\mathrm{sign}(s,\mathrm{enc}(sb)))} \qquad \dfrac{}{att(\mathrm{dec}(sb))}\,(I2)$$
$$\dfrac{\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad}{att(s)}$$

گفتنی است که عامل انسانی از بالا به پایین بررسی می‌کند و عامل غیرانسانی از پایین به بالا. پس همان‌طور که مشاهده شد محرمانگی مقدار s رعایت نمی‌شود و مهاجم از مقدار s مطلع می‌گردد.

نحوه عملکرد proverif

این ابزار در ابتدا مجموعه قواعد تعریف‌شده را ساده‌سازی می‌کند و سپس عمل جستجوی عمقی را بر روی آن انجام می‌دهد. به‌منظور ساده‌سازی قواعد گام‌های زیر را انجام می‌دهد:

- Completion : درواقع ابزار عملیاتی بر روی قواعد انجام می‌دهد. به‌طور مثال انجام عملیاتی نظیر unify کردن عبارات و معادل‌سازی آن‌ها و حذف برخی از قواعدی که از روی قواعد دیگر قابل‌استخراج هستند.

این عملیات در زمان چندجمله‌ای قابل انجام است. به‌عنوان‌مثال، در تصویر زیر می‌خواهیم عبارت F' حاصل گردد:

$$\begin{array}{lll} R & \triangleq & G \Rightarrow F \\ R' & \triangleq & H \wedge F_0 \wedge H' \Rightarrow F' \end{array}$$

پس می‌توان دو عبارت را با یکدیگر یکی کرد و بجای آن حاصل زیر را نوشت:

$$R \circ_{F_0} R' \quad \triangleq \quad \sigma(H) \wedge \sigma(G) \wedge \sigma(H') \Rightarrow \sigma(F')$$

نماد σ نیز به مفهوم unify کردن استفاده‌شده است.

- حذف قواعد بدون استفاده: به‌عنوان‌مثال در دو قاعده زیر قاعده دوم اضافه است و می‌توان آن را حذف نمود، چون از طریق قاعده اول نیز می‌توان به F رسید و قاعده دوم مقداری عبارات اضافه‌تر دارد.

$$\begin{array}{lll} R & = & G_1 \wedge \cdots \wedge G_n \Rightarrow F \\ R' & = & \sigma(G_1) \wedge \cdots \wedge \sigma(G_n) \wedge H_1 \wedge \cdots \wedge H_k \Rightarrow \sigma(F) \end{array}$$

در قواعدی که مقدار حاصل در سمت راست عبارت در سمت چپ آن قاعده نیز آمده است، آن قاعده اضافی است و می‌توان آن را حذف کرد، به چنین قواعدی همواره درست[1] گویند.

$$R \quad = \quad \cdots \wedge F \wedge \cdots \Rightarrow F.$$

- اعمال جستجوی عمقی بر روی حاصل دو مرحله قبل.

در ادامه این سه گام را بر روی پروتکل ساده بخش قبلی اجرا می‌کنیم.

اجرای سه گام proverif بر روی پروتکل ساده مثال قبل

گام اول ساده‌سازی مجموعه دانش است.قواعد R, R' را از مجموعه دانش B و F_0 به شیوه زیر انتخاب می‌کند:

[1] tautology

- $R \circ_{F_0} R'$ exists.
- F_0 is not of the form $att(x)$.
- R's left-hand-side is of the form $att(x_1) \wedge \cdots \wedge att(x_n)$.

آن‌قدر این کار را انجام می‌دهد تا دیگر نتوان آن را انجام داد، سپس حاصل را به مجموعه دانش B اضافه می‌کند. در ادامه قواعد بی‌مصرف را از مجموعه B حذف می‌کند.

اگر مجموعه دانش B بدین ترتیب باشد:

$$\begin{aligned}
&\Rightarrow att(net) && (I1)\\
&\Rightarrow att(\mathrm{dec}(sb)) && (I2)\\
att(x) \wedge att(y) &\Rightarrow mess(x,y) && (A1)\\
mess(x,y) \wedge att(x) &\Rightarrow att(y) && (A2)\\
att(x) \wedge att(y) &\Rightarrow att(\mathrm{sign}(x,y)) && (A3)\\
att(x) &\Rightarrow att(\mathrm{enc}(x)) && (A4)\\
att(x) &\Rightarrow att(\mathrm{dec}(x)) && (A5)\\
att(\mathrm{sign}(x,\mathrm{enc}(y))) \wedge att(\mathrm{dec}(y)) &\Rightarrow att(x) && (A6)\\
&\Rightarrow mess(net, \mathrm{sign}(s,\mathrm{enc}(sb))) && (P1)
\end{aligned}$$

به‌عنوان‌مثال دو قاعده I2, A6 باهم قابل یکی شدن هستند و بجای این دو می‌توان حاصل آن‌ها یعنی R1 را قرار داد.

$$\circ_{att(\mathrm{dec}(y))} (A6):\quad att(\mathrm{sign}(x,\mathrm{enc}(sb))) \Rightarrow att(x) \qquad (R1)$$

یا اگر دو قاعده A1, A2 را باهم یکی نماییم، یک تاتولوژی حاصل می‌گردد و می‌توان آن را حذف نمود.

یا P1, A2 را می‌توان باهم ترکیب کرد و R3 را به دست آورد :

$$\circ_{mess(x,y)} (A2):\quad att(net) \Rightarrow att(\mathrm{sign}(s,\mathrm{enc}(sb))) \qquad (R3)\ (X)$$

یا I1, R3 را می‌توان باهم ترکیب کرد:

$\circ_{att(net)}$ (R3) :

$$\Rightarrow att(\text{sign}(s, \text{enc}(sb))) \qquad \text{(R4)}$$

درنهایت این مجموعه پس از اجرای گام اول بدین ترتیب درمی‌آید:

$$\begin{aligned}
&\Rightarrow att(net) && \text{(I1)}\\
&\Rightarrow att(\text{dec}(sb)) && \text{(I2)}\\
att(x) \wedge att(y) &\Rightarrow mess(x, y) && \text{(A1)}\\
mess(x, y) \wedge att(x) &\Rightarrow att(y) && \text{(A2)}\\
att(x) \wedge att(y) &\Rightarrow att(\text{sign}(x, y)) && \text{(A3)}\\
att(x) &\Rightarrow att(\text{enc}(x)) && \text{(A4)}\\
att(x) &\Rightarrow att(\text{dec}(x)) && \text{(A5)}\\
att(\text{sign}(x, \text{enc}(y))) \wedge att(\text{dec}(y)) &\Rightarrow att(x) && \text{(A6)}\\
&\Rightarrow mess(net, \text{sign}(s, \text{enc}(sb))) && \text{(P1)}\\
att(\text{sign}(x, \text{enc}(sb))) &\Rightarrow att(x) && \text{(R1)}\\
att(y) \wedge att(\text{sign}(x, \text{enc}(y))) &\Rightarrow att(x) && \text{(R2)}\\
&\Rightarrow att(\text{sign}(s, \text{enc}(sb))) && \text{(R4)}\\
&\Rightarrow att(s) && \text{(R6)}
\end{aligned}$$

در ادامه گام دوم را بر روی این قواعد اجرا می‌کنیم که شامل حذف قواعد بی‌استفاده است. قواعدی که مرتبط باهدف نیستند و آن را ارضاء نمی‌کنند، حذف می‌شوند. در قواعدی به فرم زیر

$$R \;=\; \cdots \wedge H \wedge \cdots \Rightarrow F$$

که H به شکل att(x) نیست باید حذف شوند، زیرا H از قواعد پایه قابل‌استخراج نیست.

به‌عنوان‌مثال قواعد A2, A6, R1, R2 حذف می‌شوند. در زیر حاصل اجرای گام دوم آمده است.

$$
\begin{aligned}
&\Rightarrow att(net) && (I1)\\
&\Rightarrow att(\mathsf{dec}(sb)) && (I2)\\
att(x) \wedge att(y) &\Rightarrow mess(x, y) && (A1)\\
att(x) \wedge att(y) &\Rightarrow att(\mathsf{sign}(x, y)) && (A3)\\
att(x) &\Rightarrow att(\mathsf{enc}(x)) && (A4)\\
att(x) &\Rightarrow att(\mathsf{dec}(x)) && (A5)\\
&\Rightarrow mess(net, \mathsf{sign}(s, \mathsf{enc}(sb))) && (P1)\\
&\Rightarrow att(\mathsf{sign}(s, \mathsf{enc}(sb))) && (R4)\\
&\Rightarrow att(s) && (R6)
\end{aligned}
$$

گام نهایی اجرای Goal Directed DFS بر روی مجموعه قواعد حاصل از دو مرحله قبلی است. هدف رسیدن به att(s) هست که همان‌طور که در تصویر فوق مشاهده می‌شود، این هدف خود یکی از fact های مسئله شده است. (قاعده R6)

نکاتی کلی در مورد ابزار Proverif

در اینجا لازم است برخی نکات در رابطه با این ابزار ذکر شوند:

- اجرای گام اول بر روی قواعد پایگاه دانش B ممکن است هرگز خاتمه پیدا نکند. برای حل این مسئله اثبات‌شده است که اگر پیام‌های برچسب دار استفاده شوند و هر پیام با یک برچسب یکتا از سایر پیام‌ها متمایز گردد مشکلی پیش نمی‌آید.
- خروجی این ابزار sound است ولی کامل نیست. یعنی آن‌که:
  - اگر بگوید خواسته‌های امنیتی برآورده می‌شوند، حتماً برآورده می‌شوند.
  - اگر بگوید خواسته‌های امنیتی برآورده نمی‌شوند، ممکن است برآورده شوند یا خیر.

دلیل آن این است که ممکن است پروتکل را به طریقی دیگر اجرا کرده باشند. یعنی به‌عنوان‌مثال اگر ترتیب مراحل انجام پروتکلی۱ و ۲ و ۳ است این ابزار پروتکل را به ترتیبی مثلاً به شکل ۱ و ۲ و ۱ و ۲ و ۱ و ۲ و ۳ اجرا کرده است. لازم به ذکر است که ترتیب مراحل رعایت شده است و مرحله دوم حتماً بعد از اجرای مرحله اول اجرا شده است اما ممکن یک مرحله چند بار تکرار شده باشد.

- چنانچه تعداد پیام‌های پروتکل زیاد شوند و ساختار این پیام‌ها پیچیده شوند زمان اجرای تحلیل توسط این ابزار ممکن است به‌طور چشمگیری به دلیل بزرگ شدن مجموعه B افزایش یابد.
- این ابزار مقیاس‌پذیر نیست، پروتکل‌های ساده را خیلی خوب و سریع تحلیل می‌کند و پروتکل‌های پیچیده را ممکن است ساعت‌ها طول بکشد تا حل نماید.

فصل سوم :

معرفی پروتکل انتخابی

پروتکل مبادله‌ی کلید DASS

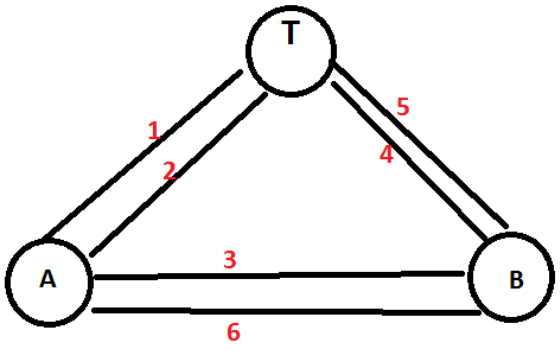
این پروتکل یک پروتکل شش گذرِ هست که سه نقش متفاوت در آن موجود است.

گذرهای زیر مطابق شکل به ترتیب رخ می‌دهد.

۱. A به S اعلام می‌کند که می‌خواهد با B در ارتباط باشد.
۲. S کلید عمومی و نام B را به‌صورت امضاشده با کلید خصوصی خود برای A ارسال می‌کند
۳. A یک پیام سه‌جزئی را برای B ارسال می‌کندکه به‌صورت زیر است:

Encks(ta), Sign prs(KUP,A,l) , Sign KRP(EncKUB(ks))

۴. B با ارسال یک پیام به S اعلام می‌کند که می‌خواهد با A در ارتباط باشد.
۵. Sکلید عمومی و نام A را به‌صورت امضاشده با کلید خصوصی خود برای B می‌فرستد
B به ترتیب جز دوم، سوم و جز اول گذر سوم را واضح کرده و کلید جلسه را استخراج می‌کند.
۶. B زمان tb را با کلید جلسه رمز می‌کند و برای A ارسال می‌کند.



فصل چهارم:

توصیف صوری پروتکل و خواسته‌های امنیتی

مقدمات توصیف پروتکل

برای پیاده‌سازی پروتکل از زبان pi استفاده کرده‌ایم به این صورت که طرفین پروتکل پیام‌های ارسالی خود را درون یک کانال عمومی قرار می‌دهند و طرف گیرنده پیام را دریافت می‌کند.برای این منظور یک کانال c به‌صورت عمومی علنی برای ارتباط ایجاد کردیم.

free c: channel.

برای تعریف متغیر به‌صورت کلی[1] از دستور free استفاده می‌کنیم و برای ایجاد نوع از دستور type استفاده می‌کنیم که در اینجا علاوه بر bitstring نیازمند نوع host و time نیز می‌باشیم.

type host.

type time.

علاوه بر تعریف نوع نیازمند تعریف توابع رمزنگاری و ترجمه رمز متقارن، رمزنگاری و ترجمه رمز نامتقارن، امضا رقمی و درستی سنجی هستیم که برای استفاده از این توابع نیازمند نوع key برای کلید رمزنگاری و ترجمه رمز متقارن و نوع skey برای تعریف کلید خصوصی رمزنگاری و ترجمه رمز نامتقارن و نوع pkey، برای تعریف کلید عمومی رمزنگاری و ترجمه رمز نامتقارن هستیم.

type key.

type pkey.

type skey.

برای تعریف تابع رمزنگاری و امضا رقمی از دستور استفاده fun می‌کنیم. درواقع با استفاده از این دستور سازنده[2] را برای رمزنگاری متقارن یا نامتقارن می‌سازیم

fun pk(skey):pkey[private].

fun sign(bitstring,skey):bitstring.

fun ksign(key,skey):bitstring.

[1] Global
[2] Constructor

fun aenc(bitstring,pkey):bitstring.

fun kadec(bitstring,skey):key.

fun aaenc(key,pkey):bitstring.

برای تعریف تابع ترجمه رمز و درستی سنجی از دستور reduc استفاده می‌کنیم. درواقع با استفاده از reduce به تعریفdestructor می‌پردازیم که تابع معکوس آن را نیز تعریف کنیم.

reduc forall m:bitstring,k:skey; adec(aenc(m,pk(k)),k)=m.

reduc forall m:key,k:skey; aadec(aaenc(m,pk(k)),k)=m.

reduc forall m: bitstring, k:skey ; getmess(sign(m,k))=m.

reduc forall m:bitstring,k:skey ; checksign(sign(m,k),pk(k))=m.

reduc forall m:key,k:skey ; kchecksign(ksign(m,k),pk(k))=m.

همچنین برای تعریف نوع طرفین A,B,S از نوع host استفاده می‌کنیم.

free A, B,S: host.

ازآنجاکه می‌خواهیم، هر ۳طرف پروتکل از بعضی از متغیرها استفاده کنند آن‌ها را global تعریف می‌کنیم به‌عنوان‌مثال می‌خواهیم ks که کلید جلسه است را به اشتراک بگذاریم بنابراین اگرچه به‌طور

global آن را در بالای processها تعریف کرده‌ایم اما از کلمه‌ی کلیدی Private بهره برده‌ایم تا محرمانه بماند. همچنین کلید خصوصی و کلید عمومی را برای طرفین تعریف کرده‌ایم ؛ که کلید خصوصی‌های آن‌ها با استفاده از کلمه‌ی کلیدی private فقط نزد خودشان محرمانه می‌ماند.

free ks:key[private].

free pkB:pkey.

free pkA:pkey.

free pkS:pkey.

free prB:skey[private].

free prA:skey[private].

free prS:skey[private].

توصیف پروتکل

برای تعریف عملیاتی که هر طرف بایستی انجام دهند برای هر طرف یک process تعریف می‌کنیم یعنی در این پروتکل سه process برای طرف A,B,S و یک process اصلی برای اجرای هم‌زمان ۳process بالا و تولید کلید عمومی طرفین از روی کلید خصوصی آن‌ها تعریف کرده‌ایم. به‌طورکلی برای تعریف process از دستور let استفاده می‌کنیم.

حال به تعریف هرکدام از طرفین و عملیاتی که بایستی انجام دهند می‌پردازیم:

let processA(pkS: pkey ,prA:skey,ks:key,pkB: pkey) =

این process کلید عمومیS، کلید خصوصی خود، کلید جلسه، کلید عمومیB را به‌عنوان ورودی می‌گیرد. کلید جلسه را خودش تولید می‌کند اما برای اینکه طرف B نیز بایستی در ProcessB به کلید جلسه دسترسی داشته باشد پس کلید جلسه را به‌صورت کلی تعریف کرده و به‌عنوان ورودی بهA می‌دهیم و عملیات را روی آن انجام می‌دهیم. در این قسمت ابتداA از روی کلید خصوصی خود کلید عمومی ایجاد می‌کند و بعد اسم B را روی کانال c می‌گذارد.

let pkA=pk(prA) in

out(c,B);

در اینجا بایستی A منتظر جواب S روی کانال باشد و زمانی که جواب درون کانال قرار گرفت جواب را از کانال دریافت کند.

in(c,x:bitstring);

جواب را با کلید عمومی S درستی آزمایی می‌کند و بایستی کلید عمومیB و اسم B باشد.

let (pkx:pkey,=B)= checksign(x,pkS) in

حال A یک جفت کلید خصوصی و کلید عمومی ایجاد می‌کند. ابتدا کلید خصوصی kRPرا تولید کرده و از روی آن کلید عمومی kUP را ایجاد می‌کند.

new kRP :skey ;

let kUP =pk(kRP) in

یک‌زمان ta و یک life time l از نوع زمان ایجاد می‌کند.

new ta:time;

new l:time;

حال یک پیام ۳جزئی را درون کانال قرار می‌دهد که این اجزا شامل رمز شده ta با کلید جلسه، امضاشده kUP,A,l با کلید خصوصی خودش، امضاشده با کلید خصوصیkRP رمز شده کلید جلسه با کلید عمومی B است.

out(c,(tencrypt(ta,ks),sign((kUP,A,l),prA),sign(aaenc(ks,pkx),kRP)));

حال منتظر جواب از سمت B می‌ماند و بلافاصله بعد از دریافت جواب از کانال آن را با کلید جلسه ترجمه کرده و زمانtbp را به دست می‌آورد.

in(c,y:bitstring);

let tbp= decrypt(y,ks) in

اگر در زمان L این tbp به دستش برسد آنگاه قبول می‌کند که کلید جلسه بین آن‌ها امن به اشتراک گذاشته‌شده است.

let processB(prB:skey,pkS:pkey,pkA:pkey) =

این process کلید خصوصی خودش، کلید عمومیS ، کلید عمومی A را به‌عنوان ورودی می‌گیرد. در ابتدا از روی کلید خصوصی خود کلید عمومی خود را ایجاد می‌کند و از کانال پیام ۳جزئی A را دریافت می‌کند.

let pkB=pk(prB) in

in(c, (m1:bitstring, m2: bitstring , m3: bitstring));

برای درستی سنجی پیام دریافتی کلید عمومی A را از S درخواست می‌کند و به این منظور اسم A را در کانال S قرار می‌دهد و منتظر جواب از سمت S می‌ماند و محض دریافت جواب آن را با کلید عمومی S درستی سنجی می‌کند و بایستی جواب کلید عمومی A و اسم A باشد.

out(c,A);

in(c,y:bitstring);

let (pkA:pkey,=A)= checksign(y,pkS) in

حال جز دوم پیام دریافتی از A را با کلید عمومی A درستی آزمایی می‌کند و بایستی کلید عمومی kUP و اسم A و L را به دست بیاورد.

let (kUP:pkey,=A,lp:time)= checksign(m2,pkA) in

حال جز سوم را با کلید عمومی kUP درستی آزمایی کرده و مقدار به‌دست‌آمده را با کلید خصوصی خود ترجمه می‌کند و کلید جلسه را به دست می‌آورد و جز اول را با کلید جلسه ترجمه کرده و زمان tap را به دست می‌آورد که اگر زمان درست بود کلید جلسه را قبول می‌کند.

let (temp:bitstring)= checksign(m3,kUP) in

let ks = kadec(temp,prB) in

let tap= decrypt(m1,ks) in

حال یک‌زمان tb ایجاد کرده و آن را با کلید جلسه رمز می‌کند و روی کانال قرار می‌دهد.

new tb:time;

out(c,tencrypt(tb,ks));

let processS(pkA:pkey,pkB:pkey,prS:skey)=

کلید عمومی A و کلید عمومی B و کلید خصوصی خود را به‌عنوان ورودی می‌گیرد. ابتدا از روی کلید خصوصی خود کلید عمومی خود را ایجاد می‌کند و پیام حاوی اسم B را از کانال دریافت می‌کند.

let pkS=pk(prS) in

in(c,=B);

حال کلید عمومیB و اسم B را با کلید خصوصی خودش امضا می‌کند و در کانال قرار می‌دهد.

out(c,sign((pkB,B),prS));

بعد دوباره از کانال اسمA را دریافت کرده و حال کلید عمومیA و اسم A را با کلید خصوصی خودش امضا می‌کند و در کانال قرار می‌دهد.

```
in(c,=A);
out(c,sign((pkA,A),prS));
```

Process

ابتدا کلید عمومی‌های طرفین از روی کلید خصوصی آن‌ها ساخته می‌شود و بعد processها به‌صورت موازی اجرا می‌شوند.

```
let pkA = pk ( prA) in
let pkB = pk ( prB) in
let pkS = pk ( prS) in
processA(pkS,prA,ks,pkB)) | (!processB(prB,pkS,pkA)) | !))
(!processS(pkA,pkB,prS)) )
```

بررسی خواسته‌های امنیتی

برای بررسی اینکه پروتکل خواسته‌های مدنظر ما را برآورده می‌کند یا نه از کوئری های خاص و event استفاده می‌کنیم که event برای زمانی است باوری برای ما حاصل می‌شود. ما برای هرکدام از خواسته‌های مدنظر توضیحات را در ادامه می‌آوریم:

محرمانگی

این خواسته محرمانه بودن کلید مشترک بین طرفین بررسی می‌کند که در اینجا با کوئری زیر این خواسته بررسی می‌شود:

```
query attacker(ks).
```

تعریف باورها

برای نشان دادن برآورده شدن خواسته‌های دیگر از event برای نشان دادن هر باور استفاده می‌کنیم. به این صورت که eventهای زیر را تعریف می‌کنیم:

event Asetkey(key). برای A باور اشتراک گذاشته شدن کلید حاصل می‌شود.

event Bsetkey(key). برای B باور اشتراک گذاشته شدن کلید حاصل می‌شود.

event Akeyconf(key). برای A باور تائید کلید حاصل می‌شود.

event Bkeyconf(key). برایB باور تائید کلید حاصل می‌شود.

event Abegintime(time). A زمانش را شروع می‌کند.

event Bbegintime(time). B زمانش را شروع می‌کند.

event Aendtime(bitstring). A زمان را خاتمه می‌دهد.

event Bendtime(bitstring). B زمان را خاتمه می‌دهد.

event Aaccept(key,pkey). طرف A کلید جلسه و کلید عمومی Bرا قبول می‌کند.

event Baccept(key). B کلید جلسه را قبول می‌کند .

event Aterm(key). Aکلید جلسه را در انتها پروتکل قبول می‌کند.

event Bterm(key,pkey).Bکلید جلسه و کلید عمومی Aرا در انتها پروتکل قبول می‌کند.

حال هرکجا که باور ایجاد می‌شود این eventهای بالا را قرار می‌دهیم و برای بررسی برآورده شدن باورها کوئری ایجاد می‌کنیم.

طریقه قرار گرفتن در هر process به‌صورت زیر هست:

```
let processA(pkS: pkey ,prA:skey,ks:key,pkB: pkey) =
let pkA=pk(prA) in
```

```
out(c,B);
in(c,x:bitstring);
let (pkx:pkey,=B)= checksign(x,pkS) in
event Aaccept(ks,pkx);
new kRP :skey ;
let kUP =pk(kRP) in
new ta:time;
event Abegintime(ta);
new l:time;
out(c,(tencrypt(ta,ks),sign((kUP,A,l),prA),sign(aaenc(ks,pkx),kRP)));
event Asetkey(ks);
in(c,y:bitstring);
let tbp= decrypt(y,ks) in
event Akeyconf(ks);
event Bendtime(tbp);
event Aterm(ks).     if pkx=pkB then


let processB(prB:skey,pkS:pkey,pkA:pkey) =
let pkB=pk(prB) in
in(c, (m1:bitstring, m2: bitstring , m3: bitstring));
out(c,A);
let (kUP:pkey,=A,lp:time)= checksign(m2,pkA) in
```

```
in(c,y:bitstring);
let (pkA:pkey,=A)= checksign(y,pkS) in
let (temp:bitstring)= checksign(m3,kUP) in
let ks = kadec(temp,prB) in
event Baccept(ks);
let tap= decrypt(m1,ks) in
event Bkeyconf(ks);
event Aendtime(tap);
new tb:time;
event Bbegintime(tb);
event Bsetkey(ks);
out(c,tencrypt(tb,ks));
event Bterm(ks,pkA);
.
```

خواسته‌ی تائید کلید

```
(* key confirm*)
query x:key; event(Bkeyconf(x))==>event(Asetkey(x)).
```

کوئری به این معنا است که اگر برای B باور تائید کلید حاصل شود قبل از این بایستی A یک کلید به اشتراک گذاشته باشد.

```
query x:key; event(Akeyconf(x))==> event(Bsetkey(x)) .
```

کوئری به این معنا است که اگر برای A باور تائید کلید حاصل شود قبل از این بایستی B یک کلید به اشتراک گذاشته باشد

خواسته‌ی تازگی

(*Freshness*)

query t:time,x:bitstring; event(Aendtime(x))==>event(Abegintime(t)).

کوئری بالا به این معنا است که اگر زمان A خاتمه داده شود بایستی قبل آن زمان A شروع شده باشد.

query t:time,x:bitstring; event(Bendtime(x))==>

event(Bbegintime(t)) .

کوئری بالا به این معنا است که اگر زمان B خاتمه داده شود بایستی قبل آن زمان B شروع شده باشد

خواسته‌ی تصدیق اصالت موجودیت

(*Etity Authentication*)

query x:key , y:pkey ; event(Bterm(x,y))==>event(Aaccept(x,y)) .

کوئری به این معنا است که اگر B کلید جلسه و کلید عمومی A را در انتها پروتکل قبول کند بایستی A کلید جلسه و کلید عمومی B را قبول کرده باشد.

query x:key ; event(Aterm(x))==>event(Baccept(x)).

کوئری به این معنا است اگر A کلید جلسه را در انتها پروتکل قبول کرد بایستی قبل از آن B کلید جلسه را قبول کرده باشد.

دو خواسته‌ی امنیتی دیگر

حال می‌خواهیم خواسته‌های بیشتری که پروتکل برآورده می‌کند را نشان دهیم. دو خواسته Week agreementو non injectiveauthenticity هستند. با توجه به اینکه ما Etity Authentication را برآورده می‌کنیم پس خواسته کلی را بررسی و برآورده می‌کنیم اما برای نشان دادن دو خواسته فوق Etity Authentication را در نظر نمی‌گیریم.

خواسته‌ی توافق ضعیف

(*Week agreement*)

query x:pkey ; event(Aend(x))==>event(Abegin(x)) .

کوئری به این معنا است که اگر A سر کلید عمومی خود توافق کرد بایستی A کلید عمومی را برای این منظور ایجاد کرده باشد.

query x:pkey ; event(Bend(x))==>event(Bbegin(x)) .

کوئری به این معنا است که اگر B سر کلید عمومی خود توافق کرد بایستی B کلید عمومی را برای این منظور ایجاد کرده باشد.

خواسته‌ی non injectiveauthenticity

این خواسته به این معنا است که پروتکل در مقابل حمله تکرار مقاوم است.

query event(Aendmsg)==>event(Bbeginmsg) .

کوئری به این معنا است که اگر A تبادل پیام را تمام کرد بایستی B تبادل پیام را شروع کرده باشد.

query event(Bendmsg)==>event(Abeginmsg) .

کوئری به این معنا است که اگر B تبادل پیام را تمام کرد بایستی A تبادل پیام را شروع کرده باشد.

طریقه قرار گرفتن در هر process به‌صورت زیر هست:

```
let processA(pkS: pkey ,prA:skey,ks:key,pkB: pkey) =
let pkA=pk(prA) in
    out(c,B);
in(c,x:bitstring);
let (pkx:pkey,=B)= checksign(x,pkS) in
event Abegin(pkx);
new kRP :skey ;
```

```
let kUP =pk(kRP) in
new ta:time;
event Abegintime(ta);
new l:time;
out(c,(tencrypt(ta,ks),sign((kUP,A,l),prA),sign(aaenc(ks,pkx),kRP)));
event Abeginmsg;
event Asetkey(ks);
in(c,y:bitstring);
event Aendmsg;
let tbp= decrypt(y,ks) in
event Akeyconf(ks);
event Bendtime(tbp);
event Aend(pkB).     if pkx=pkB then



let processB(prB:skey,pkS:pkey,pkA:pkey) =
let pkB=pk(prB) in
        in(c, (m1:bitstring, m2: bitstring , m3: bitstring));
out(c,A);
let (kUP:pkey,=A,lp:time)= checksign(m2,pkA) in
in(c,y:bitstring);
let (pkx:pkey,=A)= checksign(y,pkS) in
```

```
event Bendmsg;
let (temp:bitstring)= checksign(m3,kUP) in
let ks = kadec(temp,prB) in
event Bbegin(pkx);
let tap= decrypt(m1,ks) in
event Bkeyconf(ks);
event Aendtime(tap);
new tb:time;
event Bbegintime(tb);
event Bsetkey(ks);
out(c,tencrypt(tb,ks));
event Bbeginmsg;
event Bend(pkA);    if pkx=pkA then
0.
```

برای بررسی دو خواسته‌ی امنیتی جدید لازم است تا هر یک از باورهای ایجادشده در هر مرحله را به این صورت تعریف می‌کنیم:

```
event Abegin(pkey).
```

Aکلید عمومی را برای توافق ایجاد کرد.

```
event Bbegin(pkey).
```

Bکلید عمومی را برای توافق ایجاد کرد.

```
event Aend(pkey).
```

Aسر کلید عمومی خود توافق کرد.

event Bend(pkey).

Bسر کلید عمومی خود توافق کرد.

event Abeginmsg.

Aتبادل پیام را شروع کرد.

event Aendmsg.

Aتبادل پیام را تمام کرد.

event Bbeginmsg.

Bتبادل پیام را شروع کرد.

event Bendmsg.

Bتبادل پیام را تمام کرد.

فصل پنجم :

نتایج

برای تمام خواسته‌های امنیتی ابزار Proverif توانست به‌طور مستقیم ثابت کند که رابطه برقرار است.

همان‌طور که در تصویر زیر مشاهده می‌کنید، همه‌ی خواسته‌های امنیتی موردتوجه برآورده شده‌اند و ازآنجاکه درخت اشتقاقی برای رسیدن به آن‌ها موجود بوده است، خروجی همه‌ی آن‌ها True بوده است.

```
2nd process: Reduction ! 0 copy(ies)

1st process: Reduction ! 0 copy(ies)

New processes:

--------------------------------------------------------------
The attacker has the message pkS.
A trace has been found.
RESULT not attacker(pkS[]) is false.
-- Query not attacker(ks[])
Completing...
Starting query not attacker(ks[])
RESULT not attacker(ks[]) is true.
-- Query event(Bendtime(x_1489)) ==> event(Bbegintime(t))
Completing...
Starting query event(Bendtime(x_1489)) ==> event(Bbegintime(t))
RESULT event(Bendtime(x_1489)) ==> event(Bbegintime(t)) is true.
-- Query event(Aendtime(x_2231)) ==> event(Abegintime(t_2230))
Completing...
Starting query event(Aendtime(x_2231)) ==> event(Abegintime(t_2230))
RESULT event(Aendtime(x_2231)) ==> event(Abegintime(t_2230)) is true.
-- Query event(Akeyconf(x_3045)) ==> event(Bsetkey(x_3045))
Completing...
Starting query event(Akeyconf(x_3045)) ==> event(Bsetkey(x_3045))
RESULT event(Akeyconf(x_3045)) ==> event(Bsetkey(x_3045)) is true.
-- Query event(Bkeyconf(x_3785)) ==> event(Asetkey(x_3785))
Completing...
Starting query event(Bkeyconf(x_3785)) ==> event(Asetkey(x_3785))
RESULT event(Bkeyconf(x_3785)) ==> event(Asetkey(x_3785)) is true.
-- Query event(Bend(x_4467)) ==> event(Bbegin(x_4467))
Completing...
Starting query event(Bend(x_4467)) ==> event(Bbegin(x_4467))
RESULT event(Bend(x_4467)) ==> event(Bbegin(x_4467)) is true.
-- Query event(Aend(x_5286)) ==> event(Abegin(x_5286))
Completing...
Starting query event(Aend(x_5286)) ==> event(Abegin(x_5286))
RESULT event(Aend(x_5286)) ==> event(Abegin(x_5286)) is true.
mahmoud@ubuntu:~/Desktop/proverif1.96$
mahmoud@ubuntu:~/Desktop/proverif1.96$
mahmoud@ubuntu:~/Desktop/proverif1.96$
```

منابع

[1] Tardo, Joseph J., and Kannan Alagappan. "SPX: Global authentication using public key certificates." Journal of Computer Security 1.3-4 (1992): 295-316.

[2] Blanchet, Bruno, Ben Smyth, and Vincent Cheval. "ProVerif 1.93: Automatic cryptographic protocol verifier, user manual and tutorial." (2016).