基于串空间的 Athena 分析技术研究*)

吴光伟　董荣胜
（桂林电子科技大学计算机系　桂林 541004）

摘　要　基于串空间模型的研究是当前安全协议领域的一个研究热点。Song 对串空间模型进行了扩展，将模型检验和定理证明结合起来，提出了一种取名为 Athena 的安全协议分析方法，并基于该方法开发了自动证明工具 APV，Song 的工作被认为是串空间理论发展的一个重要事件。本文对 Athena 进行了系统的分析，介绍了 Athena 的假设条件，给出了 Athena 的语法和语义，分析了该逻辑的优点和局限性，在此基础上，分析了 Athena 的核心算法，讨论了 Athena 算法自动高效的原因，以及该算法如何避免状态空间爆炸的技术，指出了该算法的缺陷，形成原因以及解决的一般方法。最后对 Athena 方法的发展方向进行了讨论。

关键词　串空间，Athena，安全协议，定理证明，模型检验

Athena Approach Based on Strand Spaces

WU Guang-Wei　DONG Rong-Sheng
(Department of Computer, Guilin University of Electronic Technology, Guilin 541004)

Abstract　Strand space model is an important research field in security protocol analysis scopes. Song extends strand space model, borrows some efficient techniques from both model checking and theorem proving, proposes an efficient automatic checking approach, Athena, for analyzing security protocols, and develops a tool named APV based on this approach. This paper discusses Athena systematically, introduces Athena's assumption, its syntax and semantics, and discusses merit and limitation of this logical. We detailedly describe the core verification algorithm of Athena, analysis what actually makes Athena successful in fast and automatic verification of security protocol. We also point out the defects of this algorithm, discuss the reason and give the way to solve it. Finally, the future direction in this field is discussed.

Keywords　Strand spaces, Athena, Security protocol, Theorem proving, Model checking

1997 年，Thayer，Herzog 和 Guttman 基于代数系统提出了串空间模型 SSM，它是一种结合定理证明和协议迹的混和方法[1]，具有高效、严谨、直观等特点，并受到了广泛的关注。

串空间理论发展过程中有两个重要事件：其一是 Guttman 提出的串空间模型认证测试方法[2]；其二是 Song 在串空间上提出的 Athena 方法[3]。

Athena 方法有效地结合了串空间模型以及模型检验、定理证明中的各种技术，在一定程度上克服了传统自动验证工具所面临的状态爆炸和参与者数量限制问题。

本文从四个方面对 Athena 系统进行分析，重点放在紧凑的状态表示方法以及核心的搜索算法上。最后以经典的 Needham-Schroeder 公共密钥认证协议为例，分别使用 APV 与传统的模型检验工具 SMV 进行验证，并对结果进行了比较。

1　Athena 系统的初始假设

Athena 系统的初始假设与文[1]中使用的假设一致，下面介绍两个主要的假设：加密假设和入侵者模型。

(1)Athena 系统使用的加密假设

$\{m\}_k=\{m'\}_{k'}\Leftrightarrow m=m'\wedge k=k'$

也就是，两个加密项相等的充分必要条件是：加密的消息相等，加密的密钥相等。

(2)Athena 系统使用的入侵者模型。入侵者 P 的初始知识 init-info(P)包括：主体的名字和入侵者初始密钥集合 Kp。Kp 通常包括公共密钥，所有入侵者的私有密钥，入侵者和其它主体通过运用协议规则分享的对称密钥。

入侵者能够截获消息，并且能够通过初始的知识和截获的消息得到新的知识，构造新的消息。Athena 通过入侵者角色对入侵者动作进行建模。入侵者的角色定义如下：

a)$M[t]$正文消息：$\langle+t\rangle$，$t\in$ init-info(P)并且 $t\in T$；

b)$F[g]$截获：$\langle-g\rangle$；

c)$T[g]$发送：$\langle-g,+g,+g\rangle$；

d)$V[g,h]$ 连接：$\langle-g,-h,+gh\rangle$；

e)$R[g,h]$分解：$\langle-gh,+g,+h\rangle$；

f)$K[k]$密钥：$\langle+k\rangle$，这里 $K\in Kp$；

g)$E[k,h]$加密：$\langle-k,-h,+\{h\}_k\rangle$；

h)$D[k,h]$解密：$\langle-k^{-1},-\{h\}_k,+h\rangle$

其中 g 和 h 是项，T 表示的是项的集合。+表示发送消息，−表示接收消息。

Athena 假设入侵者角色的集合是：$\Pi=\{M,F,T,V,R,K,E,D\}$。

Athena 通过修改入侵者角色和入侵者的初始知识，便可扩展入侵者模型。然而，这种修改可能导致 Athena 搜索算法中 split 步骤的不可判定[4]。

*)本文得到广西自然科学基金项目(编号：0542052)的资助。

2 Athena 系统的模型

2.1 语法

项的语法包括串常量($s1,s2,\cdots$)和丛变量($C,\cdots$)。一个串常量可以代表一个部分串,在特殊的情况下,一个节点可看成一个串常量。

命题公式可以递归定义如下:

(1)$s\in C$ 是一个(原子)命题公式;

(2)若 $f1$ 和 $f2$ 是命题公式,则 $\neg f1$ 和 $f1\wedge f2$ 是命题公式。

最后,合式公式(well-formed formulas),(wffs)可以递归定义如下:

(1)$f,\neg F1,F_1\wedge F_2$;

(2)若 f 是命题公式,并且除了 C 之外,f 中不含有其它变量,则 $\forall C.f$ 也是命题公式。

这里 f 是一个命题公式,F_1,F_2 是合成公式,C 是丛变量。

若 f 涉及的是入侵者串,则 f 中的原子公式必须以负的形式出现。在 f 中原子公式 τ 的出现称为负,是在这样的情况下:τ 在奇数次非的范围下(如:$\neg\tau$ 为一次非的范围下)。这种限制能够保证 Athena 算法产生有效的反例。

2.2 语义

节点的集合表示为 N。对于一个给定的协议 p,定义这个协议的所有串(包括正则串和入侵者串)为 $\sum_P$;它们的执行轨迹构成丛的集合 D_p。对于一个给定协议 p 的模型 Mp,它是一个元组 $Mp=(N,\sum_P,D_p,I)$,这里 I 是串常量和丛变量的解释(interpretation)。若 $M'=M[I(C)\leftarrow c]$,则除了 $I'(C)=c$,模型 M' 与 M 相同。逻辑中公式的语义如下:

(1)$I(s)\in\sum_P$ 和 $I(C)\in D_p$ 分别针对串和丛,通过解释 I 对常量 s 和变量 C 赋值;

(2)$M\vDash s\in C$ 当且仅当 $I(s)\in I(C)$;

(3)若 f 是命题公式或者是合成公式 wffs,则 $M\vDash\neg f$ 当且仅当 $M\not\vDash f$;

(4)若 $f1$ 和 $f2$ 是命题公式或者是合成公式 wffs,则 $M\vDash f1\wedge f2$ 当且仅当 $M\vDash f1$ 和 $M\vDash f2$;

(5)$M\vDash\forall C.f$ 当且仅当 $\forall_c\in D_p.\ M[I(C)\leftarrow c]\vDash f$。

2.3 Athena 的证明目标

Athena 验证目标描述成模型检验问题是:给定一个协议 P(作为一个模型),检验它是否满足公式 F(作为性质)。本文只考虑命题公式最为特殊的情况 $F\equiv\forall C.f$,其它情况可以依据该情况进行分析。

$\forall$ 量词可以用合取来表示(conjunction)。f 等价于合取范式 2.1:

$$f\equiv\wedge\forall C.(\wedge\Phi\Rightarrow\vee\gamma)\qquad(2.1)$$

其中,Φ,γ 分别是原子公式的集合。

则验证问题可用式 2.2 表示:

$$P;\Gamma\vdash\Delta\qquad(2.2)$$

其中,Γ 和 Δ 分别是从假设 Φ 和结论 γ 得到的串。即:

$$\Gamma=\{s|(s\in C)\in\Phi\},\Delta=\{s|(s\in C)\in\gamma.\}\qquad(2.3)$$

根据 Athena 逻辑,式 2.2 描述的语义是:

$$\lceil P;\Gamma\vdash\Delta\rfloor_P\equiv\forall C\in D_p.\ \Gamma\sqsubseteq C\Rightarrow\Delta\bigcap C\neq\phi(空)\qquad(2.4)$$

Athena 逻辑中的原子命题公式只有 $s\in C$,按照语义,直观的描述是某个串属于某个协议运行轨迹所构成的丛。因此,Athena 系统的逻辑十分简单。描述的是串属于丛的情况。这种逻辑十分适合进行搜索算法,并且能够对协议的认证性以及秘密性进行描述。然而,这种逻辑缺乏足够的能力对更多的性质进行描述。

3 Athena 系统的规则与算法

Athena 的规则与算法是 Athena 系统中的核心部分,包括:(1)Athena 的紧凑型状态表示方法;(2)Athena 系统的规则;(3)Athena 搜索算法。

3.1 Athena 的状态表示方法

Athena 系统建立在串空间理论基础上,并从两个方面进行了扩展。一个是半丛这个概念的提出,另外一个则是目标的提出,以及目标绑定。下面将给出半丛、目标以及目标绑定的形式化的定义。然后在这些概念的基础上给出 Athena 中状态的定义。

(1)半丛。一个半丛 $h=(Nh,Eh)$ 是 N 的一个子图,这里 Eh 是边的集合,Nh 是节点的集合,为 Eh 边两端的节点,半丛满足下面的性质:

1)Nh 是非空并且是有限的;

2)若 $n1\in Nh$ 并且 $n2\Rightarrow n1$,则 $n2\Rightarrow n1\in Eh$;

3)h 是无环的。

(2)目标。是一个二元组$(t,\ n)$,这里,$\mathrm{sign}(n)=-,t\sqsubset\mathrm{term}(n)$,并且 t 不是项的连接。丛 C 中的目标集合是 C 中所有目标的集合,用 $G(C)$ 来表示。

(3)目标绑定。若 C 是一个丛,n' 是下面集合的 $\leq_C$ 最小员($\leq_C$-minimal member)

$$\Psi=\{m\in C|t\sqsubset\mathrm{term}(m),\ \mathrm{sign}(m)=+,\text{and } m\leq n\}$$

则称目标(t,n)绑定到节点 n'。

用 $n'\overset{t}{\twoheadrightarrow}n$ 表示目标绑定关系。节点 n'称为(t,n)的被绑定者(binder)。若 t 在上下文中是清楚的,通常可简写成 $n'\twoheadrightarrow n$。

丛中任何一个目标都有被绑定者。

从上面对半丛,目标以及目标绑定的定义中,能够得到下面两点结论。

1)半丛相当于丛的一个子图。由丛的定义可以知道,半丛与丛的最大不同在于它没有$\rightarrow$关系,也就是缺少两个节点消息直接传递这个关系。

2)与串空间经典理论相比,它多了一个目标和目标绑定的概念。

目标绑定弥补了$\rightarrow$这个关系。$n2\twoheadrightarrow n1$ 与 $m2\rightarrow m1$ 的不同在于:在$\twoheadrightarrow$关系中,$n2$ 必须是 $\leq_C$ 最小员,$n2\twoheadrightarrow n1$ 可能代表 $n2\rightarrow n3,n3\rightarrow n4,\cdots,n'\rightarrow n$ 这样一个无穷序列。这里根据 $\leq_C$ 最小员这个条件,$n3,\cdots,n'$ 是类型为 R,V,T 的入侵者串。用$\twoheadrightarrow$代替$\rightarrow$是避免算法在无限转递情况陷入无穷搜索。

(4)状态。一个状态是一个元组$\langle S,\ G,\twoheadrightarrow\rangle$,这里:1)$S$ 是一个半丛;2)G 是 S 中没有绑定的目标集合;3)$\twoheadrightarrow$是目标绑定的关系。

G 是可以通过 S 和$\twoheadrightarrow$计算出来。

一个串 s 在状态 $l=\langle S,\ G,\twoheadrightarrow\rangle$中,用 $s\in l$ 来表示。扩展串空间理论中替换(substitution)的概念。对状态的替换为:

$$\sigma(\langle S,G,\twoheadrightarrow\rangle)=\langle\sigma(S),\sigma(G),\sigma(\twoheadrightarrow)\rangle,$$

其中:$\sigma(S)$和$\sigma(G)$是一些相应的串,这些串所有的项用 σ 来替换并且:

$$\sigma(\twoheadrightarrow)=\{\sigma(n'),\sigma(t),\sigma(n)|\text{这里 }n'\overset{t}{\twoheadrightarrow}n\}$$

也就是，$\sigma(n')\xrightarrow{\sigma(t)}\sigma(n)$。

(5)束集合。对一个协议 P，定义关于状态 l 的丛集合 $\Psi(l')$ 如下：

半丛 $C\in\Psi(l')$ 当且仅当 1)Sl 是 c 的子图；2)l 中$\twoheadrightarrow$关系与 c 中$\rightarrow$的关系一致，也就是说，对 Sl 中所有的 $n'\xrightarrow{t}\twoheadrightarrow n$ 存在一个路径 $n'=n1,n2,\cdots,nk=n$：

a)n'发送消息 m 并且 $t\subseteq m$，并且

b)c 中不存在一个 n''发送消息 m'有 $t\subseteq m'$，并且存在一个路径 n''到 n'。

直观讲，项 t 起源于 n'并且发送到 n。

从状态的形式化定义可知，Athena 用一个状态来表示参与者个体交错执行的顺序。这种状态结构有效地避免了状态空间因为异步组成成指数型增长。另外，从串的替换定义可知，一个串的替换可以包括自由变量，Athena 中串的替换沿用了这个特征。因此，一个没有确定的自由变量能够表示一系列无穷的协议参与者。这就进一步降低了搜索空间，因而在一定程度上克服了协议参与者数量限制，以及状态搜索空间爆炸等问题。但是需要注意，替换中使用自由变量，消除的只是对称冗余，两个不同角色的协议参与者不能用同一个串表示。这是导致 Athena 不能中止的根本原因。

3.2 Athena 系统的规则

Athena 系统的规则有两个特点：

(1)规则可逆，若规则中的一个前提被证明是错误的，则结论也是错误的；

(2)规则少，只有三条，即：init 规则，split 规则和 final 规则。

定理证明器不能自动运行的一个主要原因是规则较多，很难从中构造一个策略应对所有情况。Athena 规则少，有利于形成简单的搜索算法。另外，Athena 系统的规则是可逆的，而一般的定理证明器的许多规则不可逆。换言之，当定理证明器其中的一个子目标被证明是错误的时，不能说明初始目标是错误的，只能说明当前使用的规则(或者是策略)不能证明初始目标，因此需要从证明的某个步骤上运用不同的规则(策略)。这使得一般的定理证明器具有以下两个特征：

(1)要回溯，必须对证明的过程进行记录；

(2)若不能证明初始目标正确，一般不能说明初始目标是错误的，无法提出反例。

Athena 系统规则可逆，不需回溯，若一个子目标证明是错误的，初始目标就是错误的，这个子目标就是对反例的描述。

总的说来，Athena 规则的特点使得 Athena 方法能够有效自动搜索，并且能在协议不满足性质的情况下给出反例。三条规则的形式化定义如下。

Init 规则　将验证目标$(P;\Gamma\vdash\Delta)$转换成另外一个形式的序列 $l\vdash\Delta$，这里，l 是一个状态：

$$\frac{l_0(P,\Gamma)\vdash\Delta}{P;\Gamma\vdash\Delta}\tag{3.1}$$

初始状态 $l_0(P,\Gamma)$表示的是任何一个满足下面条件的半丛：它必须满足协议 P 并且包括了 Γ 中的所有串。初始状态 $l_0(P,\Gamma)=\langle S_\Gamma,G_\Gamma,\phi\rangle$，这里，$S_\Gamma$ 表示包括了 Γ 的最小半丛，也就是说，S_Γ 是 Γ 用关系$\Rightarrow$构成的回溯闭包(backward closure)；G_Γ 是由 S_Γ 计算出来的。初始状态没有进行的目标绑定，因此关系$\twoheadrightarrow$为空。

从初始状态 $l_0=l_0(P,\Gamma)$的结构以及其语义可以看到：它满足下面的性质：

$\Gamma\subseteq l_0$

$\forall C.\ \Gamma\subseteq C\Rightarrow C\in\Psi(l_0)$

这条性质显然是可靠并且可逆的。证明省略。

Final 规则　若 $\Delta\bigcap l\neq\phi$，$l\vdash\Delta$ 称为叶序列(leaf sequent)。当 $l\vdash\Delta$ 称为叶序列的时候，运用 final 推理规则：

$$\frac{\Delta\bigcap l\neq\emptyset}{l\vdash\Delta}\tag{3.2}$$

可逆性是很明显的，可靠性证明：对于任何的半丛 C，$C\in\Psi(l)$，那么 C 是 S_l 的超图(supergraph)，又因为 $\Delta\bigcap l\neq\phi$，则 Δ 与 C 的交集(intersection)也是非空的。$l\vdash\Delta$ 得证。

若 $G=\phi$，并且 final 规则不能运用，则当前的式子 $l\vdash\Delta$ 是错误的，或者说是悖论。

原因：若 $G=\phi$，则对任何一个正则串 s，$s\in S_l$ 当且仅当 $s\in C$。现在存在一个半丛 C，$C\in\Psi(l)$，因为在 Δ 中没有一个串是在 S_l，那么它也不可能在 C 中，这意味着 $\exists C\in\Psi(l')$。$\Delta\bigcap l=\phi$，因此 $l\vdash\Delta$ 是错误的。因为所有的规则可逆，那么，初始序列也是错误的，这个悖论序列就是一个反例，它展示了针对协议的一次成功攻击。

Split 规则　若 final 规则不能运用并且 $G\neq\phi$，运用 split 规则：

$$\frac{l_1\vdash\Delta\cdots l_n\vdash\Delta,\text{where}\{l_1,\cdots,l_n\}=F(l)}{l\vdash\Delta}\tag{3.3}$$

这个规则使用后继状态函数 F 将当前状态 l 划分成后继状态，形成新的工作序列。注意：$F(l)$可以是空的，在这种情况下，split 推理规则成功地完成了对这个序列的证明。原因：如果 $F(l)$为空，则 l 中包括了相矛盾的地方，所以丛集合是空的，那么，序列显然正确。

若 F 是完全包含(complete-inclusive)时，Split 规则可靠并且可逆。

若 F 是完全包含的，则在任何一个状态，它必须满足下面的性质：

(1)$F(l)$是有穷的；

(2)$\Psi(l')=\Psi(l)$，这里 $L'=F(l)$，$\Psi(l')=\bigcup_{u\in UP(l)}\Psi(l')$。

根据该定义，Split 规则的可靠性显然满足。

现在证明，当 F 是完全包含时，split 规则是可逆的。也就是：L 的任意一个后继状态 l_i，若 $l_i\vdash\Delta$ 不成立，那么 $l\vdash\Delta$ 也不成立。证明的思路：一个序列 $l_i\vdash\Delta$ 是错误的，那么必然存在一个 C，$C\in\Psi(l')$，它与 Δ 没有任何共同的串。又因为 $\Psi(l')\subseteq\Psi(l)$，那么 $C\in\Psi(l)$，则 $l\vdash\Delta$ 不成立。

3.3 Athena 搜索算法

本文将 Athena 搜索算法分为两个部分：(1)搜索算法流程；(2)后继状态函数 F。

3.3.1 搜索算法流程

Athena 搜索算法分为三步。

第一步，使用 init 规则，将$(P;\Gamma\vdash\Delta)$转换成初始工作序列 $l\vdash\Delta$。进入第二步。

第二步，按照宽度优先的原则选择当前工作序列 $l'\vdash\Delta$ 进行分析。若无新的工作序列，则初始目标 $l\vdash\Delta$ 得证，算法中止。

(1)若当前的 $l'\bigcap\Delta\neq\phi$，根据 final 规则，当前工作序列 $l'\vdash\Delta$ 得证。转入第二步。

(2)若 $l' \cap \Delta = \phi$,且当前目标绑定 $G=\phi$,根据 final 规则,当前工作序列不成立,则初始目标 $l \vdash \Delta$ 不成立。当前状态为反例。算法中止。

(3)若 $l' \cap \Delta = \phi$,并且当前状态 $G \neq \phi$,则当前工作序列无法进行判断,进入第三步。

第三步,运用 split 规则,对当前的工作序列 $l' \vdash \Delta$ 进行划分。

后继状态函数 F 将当前状态划分成有限个后继状态,形成新的后继工作序列,后继状态可以为空,根据 split 规则,$l' \vdash \Delta$ 得证。转入第二步。

算法显然只需对首次出现的状态进行搜索。

下面从算法角度讨论搜索算法不能中止的原因。搜索中止的条件是:(1)当前状态不满足性质;或者(2)所有搜索状态满足性质。

若当前工作序列不能证明满足性质且绑定目标不为空,则运用 split 规则将当前工作序列划分成后继工作序列,又若后继状态存在一个状态有效,则搜索算法继续。如果这种情况始终出现,将导致搜索算法不能终止。要使算法中止,就要做一定的限制,若状态中参与者串达到一定数量,则认为该状态满足性质,避免状态无限划分下去。换言之,就是限定协议参与者角色数量,从而限制状态规模,使得搜索算法中止。

若该性质不满足,Athena 给出不满足性质的状态作为反例,这时 $G=\phi$。可以由该状态构造相应的协议运行。方法:如果 h 是一个图,它包括了半丛 S 以及用关系"$\twoheadrightarrow$"定义的边。对每一个 $n1 \twoheadrightarrow n2$,添加关于类型 R,V,T 的入侵者串,并配上相应的"$\rightarrow$"边,由此构造由边"$\rightarrow$"和"$\Rightarrow$"序列组成的路径。最后删除掉边 $n1 \twoheadrightarrow n2$。用相同的方法删除掉所有的 $\twoheadrightarrow$,得到一个图 h',图 h' 只包括了边"$\rightarrow$"和"$\Rightarrow$"。容易证明 h' 是一个丛。丛是一个协议的运行轨迹集合。因为只是添加了关于类型为 R,V,T 的入侵者串,h' 和 h 包含有相同的正则串。

3.3.2 后继状态函数 F 的实现

搜索算法第三步,需要使用后继状态函数 F 对当前状态进行划分,形成新的后继状态。后继状态函数 F 的形式化描述如下:

定义 3.1 一个替换 $\sigma=[t/x]$ 是变量 x 到项 t 的映射(mapping)。替换项不一定闭包,可包含自由变量。

定义 3.2 位置是一个对 $[r,i]$,这里 r 是一个角色(role)(一个参数化的串),它代表了给定协议 P 中的合法参与者或者是入侵者,i 是 r 中节点的索引,换言之,位置指定(specifies)了一个特殊(particular)角色的一个特殊节点。

定义 3.3 对于 $[r,i]$,若替换 σ 称为一个关于项 t 统一者,则相应的节点 $n=\langle r,i\rangle$ 有这样的形式 $n=\langle +t'\rangle$,对于 t 和 t'($t' \sqsubset t''$),替换 σ 是统一者。换言之,节点 $\sigma(n)$ 发送了一个项,这个项包括了 $\sigma(t)$。

定义 3.4 对于 $[r,i]$,若替换 σ 是一个关于 t 的最为普遍的统一者 MGU,则对其它的统一者 σ' 来说,存在一个替换使得 $\sigma'=\sigma \circ \gamma$。

对于给定的协议 P 和项 t(term),定义统一者的集合 $UP(t)$ 如下:

$UP(t)=\{([r,i],\sigma) \mid \sigma$ 是关于 t 和 $[r,i]$ 的一个 MGU,其中 $[r,i] \in P \cup \Pi\}$,

Π 是入侵者模型。因为任何的协议都有有限的正则串和入侵者串,并且每一个串都有有限个节点,集合 $UP(t)$ 有限。

计算协议 P 中状态 $l=\langle S,G,\twoheadrightarrow\rangle$ 的下一个状态:首先选定一个没有绑定的目标 $g \in Gl$,然后,计算关于项 t 的统一者集合 $UP(t)$,对任意一个项 $u=([r,i],\sigma) \in UP(t)$,构造下一个状态如下:

(1)使得 $Su=(\Rightarrow^{-1})*[\sigma(\langle r,i\rangle)]$ 是一个(可能是部分)串,这个串以 $\sigma(\langle r,i\rangle)$ 为结束节点,并且在 $\Rightarrow$ 上回溯闭包;它包括了 $\sigma(\langle r,i\rangle)$ 节点以及所有 $\sigma(r)$ 的前驱节点。

(2)对于任意开始于初始状态 l 的串 $s \in S$,若存在一个替换 γ_s 使得 $\gamma_s(Su) \cap \sigma(s) \neq \phi$,即 $\gamma_s(Su)$ 和 $\sigma(s)$ 有相同的节点,那么 $\sigma(s)$ 是 Su 的一个实例化。构造出一个新的状态 $l'=\langle S',G',\twoheadrightarrow'\rangle$,这里:

1)$S'=\sigma(s) \cup \{\gamma_s(Su)\}$;

2)$\twoheadrightarrow'=\sigma(\twoheadrightarrow) \cup \{\langle s',i\rangle \overset{\sigma(t)}{\twoheadrightarrow} \sigma(g)\}$,这里 $S'=\sigma(s) \cup \{\gamma_s(Su)\}$

(g 被串 s' 的第 i 个节点绑定);

3)G' 根据上面两条进行更新。

(3)将 Su 作为一个新的串,构造了附加的下一个状态 l''(additional next state)$=\langle S'',G'',\twoheadrightarrow''\rangle$,

1)$S''=\sigma(s) \cup Su$;

2)$\twoheadrightarrow''=\sigma(\twoheadrightarrow) \cup \{\langle Su, i\rangle \overset{\sigma(t)}{\twoheadrightarrow} \sigma(g)\}$

(g 绑定在 Su 上的最后一个节点);

3)G'' 根据上面两条进行更新;

所有后继状态的集合定义为 $L'=\bigcup_{u \in UP(t)} L'u$。

直观地说:后继状态函数 F 的功能是对 G 中的目标进行目标绑定,由此改变三元组的 S,G 和 $\twoheadrightarrow$,形成新的后继状态。对 G 中的目标进行绑定,目标绑定的节点有两种可能:1)绑定到原有的节点上;2)构造一个新的节点,将目标绑定到新的节点上。对于第 2 种情况,有两种可能:1)新的节点是入侵者节点;2)新的节点是合法参与者的节点,这种情况下,需要对该节点进行扩展,形成串,这个串以该节点为结束节点,并且在 $\Rightarrow$ 上回溯闭包。S,G 和 $\twoheadrightarrow$ 都被改变而形成了新的状态。这些新状态可以理解为原有状态根据目标绑定进行划分后的某个部分。

后继状态函数 F 对 G 中的目标进行绑定,考虑到所有的可能性,且目标绑定的可能性有限,根据前一节对完全闭包的定义,F 是完全包含时,那么,Split 规则是可靠并且可逆的。

对目标进行绑定时,还需注意一个细节:替换问题。

替换问题是对串中的自由变量进行赋值的问题。初始目标 $l \vdash \Delta$ 中,包括目标节点的串的参数可能已经进行了替换,称这个替换为 σ。在进行目标绑定的时候,因为 $\sigma(n') \overset{\sigma(t)}{\twoheadrightarrow} \sigma(n)$,那么,被绑定节点 n' 中的某些参数会根据 $\sigma(n)$、$\sigma(t)$ 进行替换,也就是 $\sigma(n')$。如果该节点是合法参与者的节点,那么此合法参与者角色的相应参数也要得到替换,形成特殊参与者串,这称为例化。需要注意的是,这种例化后形成的参与者串仍可以保持自由变量(参数),直观地说来,每次替换,只对串中必须替换的变量进行赋值,而其它的自由变量依然是自由变量,原来的常量保持不变。

后继状态可能无效,检验一个状态 l' 是否满足下面的条件,满足则无效:

(1)$\leq_c$ 最小员($\leq_c$ − minimal member)条件不满足。比如说,存在一个发送节点 $n1$,t 绑定在这个节点,并且有另外一个节点 $n2$,$t \sqsubset \text{term}(n2)$ 并且 $n2 \leq n1$。

(2)最后的状态图中存在关于 $\twoheadrightarrow$ 和 $\Rightarrow$ 的闭包。

(3)"唯一产生"(unique origination)的性质不满足。比如,当一个现实 Na 是由一个特定的角色中节点 $n1$ 中唯一产生的,若在状态 l 中,$Na \sqsubseteq \text{term}(n2)$ 且 $n2 \preceq n1$,则这个状态无效。

否则 l' 状态是有效的。

4 状态空间的修剪

为了进一步减少 Athena 搜索的状态,需要对状态空间进行修剪,分析不可达状态。

由 split 规则定义可知:若 $F(l)$ 为空,则在工作序列 $l \vdash \Delta$ 中,l 是矛盾的。也就是 $\Psi(l)=\phi$,l 中包括了不可达的状态。这时,工作序列 $l \vdash \Delta$ 成立。用式 4.1 表示。

$$\frac{F(l)=\emptyset}{l \vdash \Delta} \tag{4.1}$$

通过使用修剪定理,Athena 能够容易、系统的结合这些来自于定理证明的结果,有效地对证明树的状态空间进行修剪。修剪定理如下:

$$\frac{\theta(l) \text{ is true}}{l \vdash \Delta} \tag{4.2}$$

θ 是判断谓词,若 θ 为正确,则状态 l 是矛盾的。

Athena 同样提供了方便的接口,以供专家级别的用户针对他们自己的协议,提出不同的判断谓词。目前,Athena 拥有两个内建的判断谓词。在清楚的情况下,可以将判断谓词称为修剪定理。

定理 4.1 C 是一个丛,$K \in K \backslash Kp$,是合法参与者(legitimate principal)的钥匙。若 k 从未在正则节点中产生,则对于任何的入侵者节点 $p \in C$,$K \sqsubseteq \text{term}(p)$。

定理 4.1 的详细证明见文[1]。

若协议中具体的加密消息都是良构的,换言之,协议中不包括带有自由变量(可以是任何类型)的加密,可以计算由合法参与者产生消息的加密嵌套的最大深度,用 Em 来表示最大深度。

定理 4.2 l 是矛盾的,若状态满足下面的条件:

(1)若节点 $n'=\langle +h\rangle D[k,h]$ 属于入侵者串,类型为 D,并且绑定了一些目标(t,n),而且

(2)若嵌套深度比 Em 要大或者是相等。

定理 4.2 中,协议的加密消息必须是良构的。一些协议不满足这个条件,在这种情况下,必须一开始指明加密嵌套的最大深度。相似的限制出现在模型检验器中,比如 FDR,Murϕ,Brutus。

5 实例分析

Athena 方法的优势在于分析任意配置安全协议的高效性[5]。为了更直观地说明,我们选择了主频为 P4 1.5G,内存 128M 的主机,以经典的 Needham-Schroeder 公共密钥认证协议[6]为实例,分别使用的 Athena 自动验证工具 APV(操作系统为 red hat linux9.0)和传统的模型检验器 SMV(操作系统为 windows2000)进行了验证。为了保证有效地对工具本身进行比较,避免出现因为程序设计不同而造成的误差,我们使用了公开的源码。其中 APV 使用的源程序为该工具自带实例,SMV 的源程序则采用了由 Ilya Issenin 于 2001 年提出的程序[7]。

Needham-Schroeder 公共密钥认证协议描述如下:

Message 1. A→B :$\{N_a, A\}_{Kab}$

Message 2. B→A :$\{N_a, N_b\}_{Kab}$

Message 3. A→B :$\{N_b\}_{Kab}$

试验数据表明 Athena 系统有效地克服了因为多对协议运行而造成的状态爆炸问题,试验数据如表 1 所示。

表 1

用户时间(user time)	系统时间(system time)	状态(state)
70ms	0ms	27

使用 SMV 的源程序,在两对协议运行的情况下,出现了状态爆炸问题。在一对协议运行的情况下,实验数据如表 2 所示。

表 2

用户时间(user time)	系统时间(system time)	分配的 BDD 节点
17.1406 s	0.1875 s	1692528

结论 Athena 未来的发展方向包括两个方面:

(1)对 Athena 中的搜索算法进行修改[8,9],包括对 Athena 的假设条件进行修改,这方面目前有 Sergey Berezin 等人的工作[4]。同时,Athena 算法的 split 规则在一定程度上有不清晰的情况,特别是在对入侵者模型进行修改的条件下,这意味着核心搜索算法在一个基本步骤上是不可判定的。因此,对 split 规则进一步进行修改是必须的。

(2)对 Athena 建立的理论基础进行进一步的完善和扩展。Athena 所建立的逻辑能够有效的表示一些安全性质,包括秘密性,认证性,但同时有着一定的局限性,难以表达其它的性质。对 Athena 的逻辑进行修改,进一步扩展对串空间理论,增加更多的密码学原语,使其能应对更为复杂的协议,使得 Athena 算法能够更多地运用到实际领域。

最后需要说明的是,本文所提的 Athena 系统,与由麻省理工学院的 Konstantine Arkoudas 开发的取名为 Athena 的交互式定理证明系统不同[10]。

参 考 文 献

1 Thayer F J, Herzog J C, Guttman J D. Strand spaces: Why is a security protocol correct? In: Proc. of 1998 IEEE Symposium on Security and Privacy, 1998

2 Guttman J D, F'abrega F J T. Authentication tests. In: Proc. 2000 IEEE Symposium on Security and Privacy. May, IEEE Computer Society Press, 2000

3 Song A P D, Berezin S. Athena: a novel approach to effcient automatic security protocol. analysis. Journal of Computer Security, 2001(9):47~74

4 http://www.sergeyberezin.com/publications.php

5 卿斯汉. 安全协议. 北京:清华大学出版社,2005

6 Needham R, Schroeder M. Using Encryption for Authentication in Large Networks of Computers. Communications of the ACM, 1978

7 http://www.ics.uci.edu/~isse/index.html

8 Perrig A, Song D X. Looking for diamonds in the desert: Extending automatic protocol generation to three party authentication and key agreement protocols. In: Proc. of the 13th IEEE Computer Security Foundations Workshop, IEEE Computer Society Press, July 2000

9 Berezin S. Model Checking and Theorem Proving: a Unified Framework: [PhD thesis]. Carnegie Mellon University, 2002

10 http://www.cag.csail.mit.edu/~kostas/dpls/athena/