基于改进认证测试理论的高效安全协议验证算法

李谢华 高春鸣

(湖南大学计算机与通信学院 长沙 410082)

摘 要 提出一种基于改进认证测试理论的高效安全协议验证算法——AAAP(Automatic Analyzer for Authentication Protocols)算法。利用认证测试理论中消息间的偏序关系避免状态空间爆炸的问题,通过算法实现改进认证测试中的三条定理,从而证明安全协议的安全属性。实验证明,该算法能够检测出 Neuman-Stubblebine 协议中存在的类型攻击,并在对 NSL (Needham-Schroeder-Lowe) 协议的证明中较其他几种经典算法具有更高的效率。

关键词 安全协议,改进的认证测试,Neuman-Stubblebine 协议,NSL 协议

中图法分类号 TP311 **文献标识码** A

Efficient Protocol-proving Algorithm Based on Improved Authentication Tests

LI Xie-hua GAO Chun-ming

(School of Computer and Communication, Hunan University, Changsha 410082, China)

Abstract A new efficient protocol-proving algorithm——AAAP(Automatic Analyzer for Authentication Protocols) had been proposed for verifying security protocols. This algorithm is based on the improved authentication tests model, which enhances the original model by formalizing the message reply attack. With exact causal dependency relations between messages in this model, the protocol-proving algorithm can avoid the state explosion caused by asynchronous. In order to get the straight proof of security protocols, three authentication theorems were exploited for evaluating the agreement and distinction properties. When the algorithm terminates, it outputs either the proof results or the potential flaws of the security protocol. The experiment shows that, comparing with other algorithm, the protocol-proving algorithm can detect the type flaw attack on Neuman-stubblebine protocol, and prove the correctness of NSL protocol more efficiently.

Keywords Security protocol, Improved authentication tests, Neuman-Stubblebine protocol, NSL protocol

安全协议作为保障网络信息安全的基础,被广泛地应用于无线通信、军用通信、电子商务等重要领域。由于安全协议的重要性,因此对于其自身安全性能的研究就显得尤为重要。以往,对于安全协议的验证大多通过手工推导完成,不仅准确度受到质疑,还可能人为地引入误差。鉴于此,安全协议的自动化验证技术一直都是安全协议研究中的一个重要方面,并且涌现出诸如 NRL 协议分析器[1]、Murφ[2]等自动化协议分析工具。这些工具多采用攻击构造法来验证协议的安全属性,通过构造和检测协议所有可能的状态空间来发现协议中潜在的安全漏洞以及相应的攻击方法。这类方法的优点在于自动化程度高,缺点则是会产生大量的中间状态,从而造成状态空间爆炸的问题。

为解决状态空间爆炸的问题,研究人员又提出了证据构造法,用于证明协议所能满足的安全属性而非构造攻击。在证据构造法中最具代表性的方法就是串空间模型[3]和认证测试方法[4],这两种方法以其严格的数学定义和理论推导大大降低了协议分析过程中产生的中间状态,并成功发现了一些经典安全协议的漏洞,因此被公认为是验证安全协议安全属性的最有效工具之一,Athena 系统[5]就是以串空间模型为理论基础,结合模型检测方法对安全协议进行设计、分析及代码生成。该系统利用串空间模型理论和剪切算法在一定程度上缩减了状态空间数量,但仍不能从根本上解决状态空间随协议规模成级数倍增长的问题[6]。此外,由于串空间模型和认证测试方法本身的不完善,导致无法检测由于消息重放而引起的攻击。为解决现有认证测试方法存在的问题,文献[7]中引入了消息类型的概念,并将消息类型的检测融入认证测试方法中,提出改进的认证测试方法,从而有效检测出安全协议中存在的类型攻击。

本文以改进的认证测试方法为理论基础,提出了一种高效的安全协议验证算法——AAAP (Automatic Analyzer for Authentication Protocols)算法,通过前向证明的方式验证安全协议的安全性能。此外,文中还针对认证测试方法的特点,增加了测试分量唯一性属性,从而完善了安全协议的正确性属性。

1 AAAP 算法中安全协议的描述方法

1.1 安全协议的结构描述

在串空间模型和认证测试方法中,安全协议的描述是通

到稿日期:2008-09-23 本文受国家高技术研究发展计划(863)项目(2007AA010404)资助。

李谢华(1977—),女,博士,讲师,主要研究方向为网络安全协议、形式化理论等,E-mail:lixiehua7899@sina.com;高春鸣(1957—),男,教授,主要研究方向为形式化模型等。

过偏序关系而构造节点间有向图来实现的。参加协议运行的主体被定义为不同的角色，如协议发起者、协议响应者、服务器以及攻击者。串代表了一个主体在协议运行过程所有接收和发送的消息集合。一个簇由一系列的串组成，包括正常主体串和攻击者串，用于代表一个协议的完整运行过程。AAAP算法的验证过程就是针对每个角色进行的，以NSL协议[8]为例，此协议中存在两个角色，发起者和响应者，因此在AAAP算法中NSL协议的协议格式表示为：

$$
\begin{array}{ccc}
\text{Init}[A,B,N_a,N_b] & & \text{Resp}[A,B,N_a,N_b] \\
1\langle +\{|A,N_a|\}_{K_B}\rangle & \rightarrow & \langle -\{|A,N_a|\}_{K_B}\rangle \\
\Downarrow & & \Downarrow \\
2\langle -\{|N_a,N_b,B|\}_{K_A}\rangle & \leftarrow & \langle +\{|N_a,N_b,B|\}_{K_A}\rangle \\
\Downarrow & & \Downarrow \\
3\langle +\{|N_b|\}_{K_B}\rangle & \rightarrow & \langle -\{|N_b|\}_{K_B}\rangle
\end{array}
$$

其中，协议发起者串中包含的3个状态及轨迹为$\langle +\{|A,N_a|\}_{K_B},-\{|N_a,N_b,B|\}_{K_A},+\{|N_b|\}_{K_B}\rangle$，协议响应者串中包含的3个状态及轨迹为$\langle -\{|A,N_a|\}_{K_B},+\{|N_a,N_b,B|\}_{K_A},-\{|N_b|\}_{K_B}\rangle$。符号“⇓”代表同一个串中不同状态间的转换边，符号“→”代表不同串之间的消息交换的转换边。

1.2 AAAP算法中串和簇的结构

在算法描述之前首先要定义协议中各个要素的结构，其中包括串、簇和协议目标集等。

定义1 一个簇$C=(N_c,E_c)$，其中$E_c\subset(\Rightarrow\cup\rightarrow)$代表边的集合，$N_c$是和$E_c$内的边相联系的节点集合，则簇$C$满足以下性质：

1)N_c是有限集合；

2)如果$n_1\in N_c$并且$term(n_1)$为负，则存在一个唯一的节点n_2使得$n_2\rightarrow n_1\in C$；

3)如果$n_1\in N_c$并且$n_2\Rightarrow n_1$，则$n_2\Rightarrow n_1\in C$；

4)C中没有环路。

定义2 一个串$S=(N_s\cup\Rightarrow)$，其中N_s是S中所有节点的集合，$\Rightarrow$是同一个串中两个相邻节点间的转换边集合。则串S满足以下属性：

1)节点$n=\langle s,i\rangle\in N_s$，对于$1\leqslant i\leqslant length(tr(s))$；

2)如果$n_2\in N_s$，则$n_1\Rightarrow n_2\in\Rightarrow$；

3)无符号消息项t起始于节点$n\in N_s$，当且仅当$term(n)$的符号为正并且$t\subset term(n)$，对于同一个串中所有节点n之前的节点n'都有$t\notin term(n')$；

4)无符号项t是唯一产生的，当且仅当t在一个唯一的节点$n\in N_s$中产生；

5)节点n上的消息t可以表示为一个复合结构$\langle data,type,key,inverkey\rangle$，其中$data$代表项$t$中的数据，它是一个复合结构，用来表示一条消息内多个消息项的连接；key和$invkey$代表该项使用的密钥对，对于非对称密钥对$key=K$则$inverkey=K^{-1}$，对于对称密钥$key=invkey$，对于明文消息$invkey=\varphi$；

6)消息项$data$是一个复合结构，表示为$\langle value,type,check\rangle$，其中$value$代表该消息项的数据，$type$用于指示该数据的数据类型，数据类型的定义参照4.3节；$check$是布尔型变量，用于指示该数据是否需要检测，若需要检测则值为true，不需检测则值为false；

7)$\Rightarrow^+$代表测试分量的转换边，并且$\Rightarrow^+\subset\Rightarrow$。

定义3 状态$s=(S,\sigma,t,n,\Rightarrow^+)$，其中$S$是一个指定主体的串，$\sigma$代表消息项的符号“+，−”，$t\in A$代表串中交换的项，$n$是当前节点，$\Rightarrow^+$是测试分量的转换边。

定义4 一个消息项是由一个或多个原子消息或复合消息连接而成的，消息项t的定义如下：

$$t=t_1||t_2||\cdots||t_i,i=1,2,3\cdots n$$

其中$t_i\in T\cup K\cup E\cup C$，符号“||”为连接符。集合$T,K,E,C$在2.3节中有详细的定义。

定义5 假设C是一个簇，且$n\in C$，对于协议主体P，$m\in A$是节点n上的原子消息，则$T(P,m,n)$是m在n上的消息类型。

定义6 协议目标表示为$\langle P,g\rangle$，其中P为主体标识，$g\in A$是协议目标。对于主体P所有的安全目标组成目标集合$G(P)$，$G(P)$的初始状态为协议需要满足的所有目标，算法执行过程中，每实现一个目标$G(P)$中的相应内容就会被删除，如果协议能够满足所有的安全目标，则$G(P)$的最终状态为空。

引理1 C为一个簇，且$n,n'\in C,n'\prec n$。$m,m'\in A$分别为节点n,n'上的消息。$\forall m'\in A$，如果$m'\in S(n,n\Rightarrow n')$，并且$T(P,m,n)\neq T(P,m'n')$，则$m\notin S(n'n'\Rightarrow)$。对于引理1的证明详见文献[7]。

1.3 安全协议的认证属性

Gavin Lowe在文献[9]中提出了使用一致性属性来描述安全协议的认证特性，一致性属性满足了认证协议对于通信主体身份唯一性的要求。一致性属性的描述如下：

每当响应主体B和主体A之间使用某一特定数据$\vec{x}$完成一次会话，则都存在唯一的一次协议运行，使得主体A作为协议发起者和B使用$\vec{x}$完成一次会话。

另外一种较弱的一致性属性为非单射一致性属性，它不保证协议的唯一性。非单射一致性属性不要求协议运行的唯一性，仅要求每当响应者B和A使用随机数$\vec{x}$完成一次会话，都存在一次协议运行，使得主体A作为协议发起者和B使用$\vec{x}$完成一次会话。非单射一致性属性可以描述为：

$$\forall C.\ \text{Resp}(\vec{x})\in C\Rightarrow\text{Init}(\vec{x})\in C$$

其中，$\text{Resp}(\vec{x})$和$\text{Init}(\vec{x})$代表使用数据$\vec{x}$的响应者串和发起者串。由于在每次协议运行过程中随机数的产生都是唯一的，因此一致性属性通常可以由非单射一致性属性替代，即如果一个安全协议被证明是满足非单射一致性属性的，则它也能够满足一致性属性。

非单射一致性属性使用改进的认证测试方法可以描述为：

$$\forall C.\ s_1(t)\in C\Rightarrow s_2(t)\in C$$

其中，s_1和s_2为使用t为测试分量的正常主体串，C为s_1和s_2所在的簇。

虽然一致性属性满足了认证协议对于通信主体身份唯一性的要求，但是不能够保证在一次协议运行中随机数产生的唯一性，因此无法检测由于测试分量的重放而引起的类型攻击。为满足随机数产生的唯一性，安全协议还需要满足测试分量的唯一性属性。测试分量的唯一性属性可以描述为：

在一次协议运行过程中，若t为测试分量，且t在节点n上唯一产生，则对于所有节点$n'\prec_s n$，都有$t\not\subset term(n')$。

测试分量唯一性属性使用改进的认证测试方法可以描述为：

$\rightarrow s_2(t) \sqsubset C.\ s_1$

其中，s_1 和 s_2 为使用 t 为测试分量的正常主体串，C 为 s_1 和 s_2 所在的簇。

通过检测测试分量的唯一性，可以验证协议中主体收到的一条消息是否是新鲜的，因而确定是否存在重放或类型攻击。由于测试分量是改进认证测试方法的证明基础，因此对于协议的验证首先要求测试分量满足唯一性属性，如果满足才能对其一致性属性进行证明，否则说明测试分量本身不合法，协议存在漏洞，验证也随即中断。

非单射一致性属性和测试分量唯一性属性可以用来评测认证协议的安全属性。

在 AAAP 算法中，通过实现文献[7]中的 3 个改进认证测试方法来证明协议满足的非单射一致性属性，并且通过实现引理 1 来证明协议满足的测试分量唯一性属性。

2 AAAP 算法的设计与分析

AAAP 算法是以改进的认证测试方法为工具，以认证协议的一致性属性和测试分量唯一性属性为标准建立的认证协议安全性验证算法。该算法针对协议中的每个正常主体分别对协议的测试分量唯一性属性和非单射一致性属性进行验证，从而证明协议的完整安全性。

2.1 AAAP 算法的结构

对于一个给定的协议，该算法针对每个正常主体分别进行证明，因此证明分为发起者端的证明和响应者端的证明两个部分，每个部分完成的程序流程均相同。算法首先构造测试分量并且根据测试分量的性质将其进一步分类为出测试、入测试和主动测试。接下来，算法对测试分量和构造测试分量之前的节点的数据项进行比较，从而证明测试分量的唯一性。若测试分量与前面节点中交换的数据项不相同则表明测试分量是唯一的，否则测试分量不唯一就表明协议中存在可能导致消息重放攻击的安全漏洞。在证明测试分量的唯一性后就使用改进的认证测试方法根据测试分量的类别进行协议一致性的证明。AAAP 算法的结构如图 1 所示。

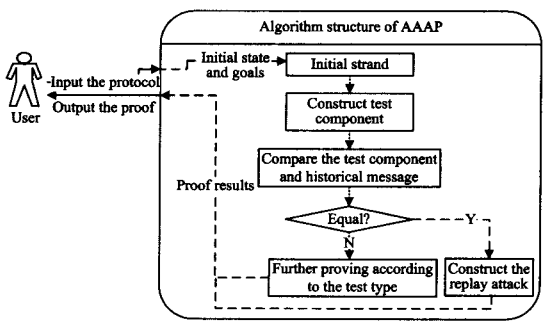


图 1 AAAP 算法结构图

2.2 算法描述及功能模块

从图 1 中可以看出，使用 AAAP 算法进行协议验证主要包括 3 个步骤，首先构造包含随机数的测试分量，并将测试分量分为入测试、出测试和主动测试；然后，通过比较测试分量和串中其他消息项的消息类型，验证测试分量的唯一性属性；最后，根据测试分量的类型证明一致性属性，从而证明协议的完整安全性。在 AAAP 算法中，*test-constructing* 模块用于构造测试分项，*test-proving* 模块用于根据测试分量的不同类型验证测试分量一致性属性和唯一性属性。在 AAAP 算法中 *st* 代表测试分量，α 代表在本次协议运行过程中唯一产生的随机数。算法中的描述都使用伪代码。AAAP 算法的主程序如算法 1 所示。

算法 1 AAAP (main)

```
S = {strand of principal P}; \\ 串 S 的初始状态
if (¬empty(S)) then \\若串为非空
  st = test-constructing(S,t, α)
  For (s∈S∧s.σ=+) do
    If (T(P,s.t,n′)≠T(P,s.t,n)) then \\唯一性证明
      test-proving(st) \\证明测试分量的一致性
    else 显示存在漏洞的部分
    end if
  end for
end if
```

在 AAAP 的主程序中 S 是一个正常主体的串，包含了该主体在协议运行过程中所有的状态和状态转换。算法中首先选择一个主体，然后调用 *test-constructing* 模块构造测试分量，并根据测试分量的类型进行分类。在构造完测试分量后要证明测试分量是否满足唯一性属性，因此比较主体串 S 中的测试分量 $T(P,s.t,n')$ 和串中其他消息分量 $T(P,s.t,n)$ 是否相同，从而确定测试分量的唯一性。如果唯一则调用 *test-proving* 模块，根据测试分量的类型进一步证明协议能够满足的一致性属性，否则测试分量可能被重放，从而导致类型攻击，算法结束并输出安全协议中可能存在的漏洞。

进行协议验证的第一步是利用 *test-constructing* 模块构造测试分量，并将测试分量根据类型不同分为出测试、入测试，以便于唯一性和一致性的证明。在改进的认证测试方法中，测试分量必须包含每次会话中唯一产生的随机数。根据随机数所在消息的形式又将测试分量分为出测试、入测试和主动测试。*test-proving* 模块则是用来验证协议中有哪些分量可以被证明，算法 2 分别给出了 *test-constructing* 和 *test-proving* 的伪代码，在该算法中 s,s' 代表串 S 中的不同状态，t_0,t_1 为代表消息的变量，K,K_X 分别代表通信主体和攻击者掌握的密钥集。算法 2 描述如下：

算法 2 protocol analyzing algorithm

```
Procedure test-constructing(S, st, α)
  if (¬empty(S)) then
    if (s.σ=+∧α∈s.t) then \\构造入测试、出测试
      t0 = s.t;
      if (α∈t0∧t0.type=ciphertext∧t0.inverkey∉KX) then
        st.t=t0; \\ st 为测试分量
        if (α∈s′.t∧s′·σ=−∧t1.type=ciphertext∧t1.inverkey
∈K) then
          t1=s′.t;t0⇒+t1 是 α 的出测试;
        end if
        if (α∈t0∧t0.type=plaintext) then
          if (α∈s′t.∧s′.σ=−∧t1.type=ciphertext) then
            st.t=t1;t0⇒+t1是 α 的入测试
          end if
        end if
      end if
    end if
  end if
```

```
end procedure
Procedure test-proving(st)
  if (¬empty(S)) then)
    g = proved item; \\已被证明的分量
    G = G\g; \\从目标集中删除已被证明的分量
    if (¬empty(G)) then
      t=select goal form G; \\从目标集中找出下一个要证明的分量
      test-constructing (S,t,α);
    else each item has been proved
    end if
  end if
end procedure
```

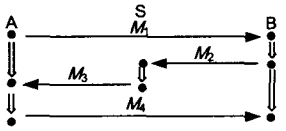
3 基于 AAAP 算法的安全协议验证结果分析

本节我们使用 AAAP 算法对 Neuman-Stubblebine 认证协议[9]和 NSL 认证协议进行实验验证，并根据验证结果和算法运行过程中产生的状态将 AAAP 算法和其他同类算法进行比较分析。

使用 AAAP 算法进行协议验证的过程中，所有的验证步骤都分为发起者端的验证和响应者端的验证两部分。

3.1 Neuman-Stubblebine 协议的实验验证

Neuman-Stubblebine 协议是由三方主体参与的身份认证及密钥分配协议，其协议交换过程如图 2 所示。其中 A，B 分别代表协议的发起者和响应者，S 代表密钥分发服务器；N_a，N_b 代表由 A 和 B 产生的随机数；K_{AS}，K_{BS} 代表 A，S 和 B，S 之间的共享密钥；K 是由 S 分发的后继会话密钥。



$M_1 = A, N_A$

$M_2 = B_1\{|A, N_A, T_B|_{K_{BS}}, N_B\}$

$M_3 = \{|B, N_A, K, T_B|\}_{K_{BS}}, \{|A, K, T_B|_{K_{BS}}, N_B\}$

$M_4 = \{|A, K, T_B|_{K_{BS}}, \{|N_B|\}\}K$

图 2 Neuman-Stubblebine 协议消息交换过程

发起者端的实验验证结果

使用 AAAP 算法对协议发起者端进行验证的结果如图 3 所示。

```
Protocol:Neuman-Stubblebine
This is the Initiator's Guarantee
Goal Set:
goal is A;
goal is B;
goal is Na;
goal is K;
goal is TB;
We use Incoming Test
We have proved A by Incoming Test;
We have proved B by Incoming Test;
We have proved Na by Incoming Test;
We have proved K by Incoming Test;
We have proved TB by Incoming Test;
The protocol is right,every component has been proved!
```

图 3 Neuman-Stubblebine 协议发起者端的验证

由以上验证结果可知，发起者能够对协议交换过程中使用的每一个变量进行验证，因此可以得到发起者端验证成功的结果。

响应者端的实验验证结果

使用 AAAP 算法对协议响应者端进行验证的结果如图 4 所示。

```
Protocol:Neuman-Stubblebine
This is the Responderr's Guarantee
Goal Set:
goal is A;
goal is B;
goal is Nb;
goal is K;
goal is TB;
We use Incoming Test
First,we use Incoming Test;
Second,we use Unsolicited Test;
The test component has the same type with former message;
Prove false,and there is something wrong in the protocol!
```

图 4 Neuman-Stubblebine 协议响应者端的验证

从验证结果可知，在验证过程中发现了测试分量和先前的某条消息类型相同，因此不能满足测试分量唯一性属性，从而导致协议存在漏洞，算法终止。

3.2 NSL 协议的实验验证

NSL 协议是对 Needham-Schroeder 公钥协议的改进，其功能是完成通信主体的双向身份认证。NSL 协议的消息交换过程如图 5 所示，其中 A，B 代表通信主体的身份标识；N_a，N_b 分别是 A 和 B 产生的随机数；K_A，K_B 分别是 A 和 B 的公钥。

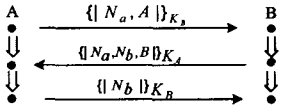


图 5 NSL 协议消息交换过程

发起者端的实验验证结果

用 AAAP 算法对 NSL 协议发起者端进行验证的结果如图 6 所示。

```
Protocol:Needham-Schroeder-Lowe
This is the Initiator's Guarantee
Goal Set:
goal is B;
goal is Nb;
goal is A;
goal is Na;
We use Outgoing Test
First,we use Outgoing Test Part One;
Second,we use Outgoing Test Part Two;
We have proved Na by Outgoing Test Part One;
We have proved A by Outgoing Test Part One;
We have proved B by Outgoing Test Part One;
We have proved Nb by Outgoing Test Part Two;
The protocol is right,every component has been proved!
```

图 6 NSL 协议发起者端的验证

(下转第 128 页)

结束语 主动网络管理体现了主动网络的思想，将一部分网络管理功能动态地分布在主动节点上，充分利用了主动节点的计算能力，使节点能够自动发现、解决问题，从而极大地优化了网络管理。本文讨论了一种主动网络的管理模型，该模型中各个模块相互独立、任务明确，而且在每个层都可以动态更新以适应主动网络中主动节点的易变性和主动应用的扩展性，因此网络管理的稳定性和扩展性都大为提高，适应了现代网络管理的需要。

参 考 文 献

[1] Di Fatta G, Gaglio S, Lo Re G, et al. Adaptive Routing in Active Networks// IEEE Openarch 2000. Tel-Aviv Israel, March 2000

[2] Di Fatta G, Lo Re G. Active Networks[J]. An Evolution of the Internet// Proc. of AICA2001-39th Annual Conference. Cernobbio, Italy, Sept. 2001

[3] Munir S. Active Networks: A Survey[EB/OL]. http://www.cse.ohio-state.edu/jain/cis788-97/ ftp/ activenets/index.htm, 2000-07-02

[4] Al Shaer E. Active Management Framework for Distributed Multimedia Systems [J]. Journal of Networks and Systems Management, 2000, 8: 49-72

[5] Brunner M, Stadler R. Service Management in Multi-party Active Networks [J]. IEEE Communications Magazine, 2000, 38 (3): 281-286

[6] Calvert K L. Directions in Active Networks [J]. IEEE Communications Magazine, 1998, 36 (1): 72-78

(上接第 76 页)

响应者端进行实验的验证结果

使用 AAAP 算法对 NSL 协议响应者端的验证结果如图 7 所示。

```
Protocol: Needham-Schroeder-Lowe
This is the Responder's Guarantee
Goal Set;
goal is B;
goal is Nb;
goal is A;
goal is Na;
We use Outgoing Test
First, we use Outgoing Test Part One;
Second, we use Outgoing Test Part Two;
We have proved Na by Outgoing Test Part One;
We have proved B by Outgoing Test Part One;
We have proved A by Outgoing Test Part One;
We have proved Nb by Outgoing Test Part Two;
The protocol is right, every component has been proved!
```

图 7 NSL 协议响应者端的验证

从以上验证结果可知，AAAP 算法证明了 NSL 协议能够满足完整的身份认证和消息保密的属性。

3.3 AAAP 算法的效率分析

AAAP 算法通过实验验证了 NSL 协议的正确性，将其算法效率同 Athena 算法和 Murφ 算法进行比较，表 1 列出了 3 种算法在对 NSL 协议进行证明过程中产生的状态数量。可以看出，当 NSL 协议仅有一对主体参与的情况下，AAAP 算法仅遍历 10 个状态就能够证明 NSL 协议所满足的认证属性，算法收敛性较好。

表 1 NSL 协议验证过程状态遍历情况

验证工具	Murφ	Athena	AAAP
遍历状态	1706	19	10

结束语 本文在改进的认证测试方法的基础上提出了安全协议自动化验证算法——AAAP 算法。该算法以 3 条改进的认证测试规则为依据实现了对于安全协议非单射一致性属性的正面。此外，本文还提出了测试分量唯一性属性，从而增强了 AAAP 算法检测重放攻击的能力。最后，通过对 Neuman-Stubblebine 协议和 NSL 协议的实验验证证明了 AAAP 算法的有效性，并通过将 AAAP 算法与 Athena 算法、Murφ 算法的遍历空间数进行比较，说明了 AAAP 算法在协议验证中的高效性。

参 考 文 献

[1] Meadows C, Syverson P F, Cervesato I. Formal specification and analysis of the Group Domain of Interpretation Protocol Using NPATRL and the NRL Protocol Analyzer. Journal of Computer Security[J], 2004, 12(6): 893-931

[2] Mitchell J C, Mitchel M, Stern U. Automated analysis of cryptographic protocols using Murφ // Proceedings of the 1997 IEEE Symposium on Security and Privacy[C]. Okland: IEEE Computer Society Press, 1997: 141

[3] Thayer F J, Herzog J C, Guttman Joshua D. Strand space : why is a security protocol correct [A] // Proceedings of the 1998 IEEE Symposium on Security and Privacy[C]. California, USA: IEEE Computer Society Press, 1998: 160-171

[4] Guttman J D, Fabrega T F J. Authentication tests [A] // Proceedings of 2000 IEEE Symposium on Security and Privacy [C]. Oakland CA: IEEE Computer Society Press, 2000: 96-109

[5] Song D X. Athena: A new efficient automatic checker for security protocol analysis // Proceedings of the 1999 IEEE Computer Security Foundations Workshop [C]. Washington DC, USA: IEEE Computer Society Press, 1999: 192-202

[6] Liu D X, Li X Y, Bai Y C. An attack-finding algorithm for security protocols. Journal of Computer Science and Technology[J], 2002, 17(4): 450-463

[7] Li Xiehua, Yang Shutang, Li Jianhua, et al. Security protocol analysis with improved authentication tests[A] // Proceedings of the 2nd Information Security Practice and Experience Comference [C]. vol. 3909 of Lecture Notes in Computer Society[C]. Hangzhou: Springer Verlag, 2006: 123-133

[8] Lowe G. Breaking and fixing the Needham-Schroeder public-key protocol using FDR // Proceedings of TACAS[C]. volume 1055 of Lecture Notes in Computer Science. Springer Verlag, 1996: 147-166

[9] Lowe G. A hierarchy of authentication specifications[A] // Proceedings of the 10th Computer Security Foundations Workshop [C]. Washington DC: IEEE Computer Society, 1997: 31-43

[10] Neuman B C, Stubblebine S G. A note on the use of timestamps as nonces[J]. Operating Systems Review, 1993, 27(2): 10-14