改进Athena算法的多协议攻击自动化验证方法

刘 威[1,2] 郭渊博[1,2] 雷新锋[3] 李俊锋[4]
(解放军信息工程大学 郑州 450001)[1] (数学工程与先进计算国家重点实验室 郑州 450001)[2]
(中国人民解放军第61840部队 北京 100097)[3] (太原卫星发射中心 太原 036300)[4]

摘 要 多协议环境下协议安全性问题是安全协议形式化分析验证领域的一个公开问题。针对此问题,在分析Athena算法的基础上提出了一种多协议攻击自动化验证方法。该方法扩展了Athena状态表示方法和后继状态生成算法,使得攻击者具备截取其它协议交互消息和计算生成当前协议消息的能力,能够以自动化的方式验证协议是否存在多协议攻击。实验结果表明,提出的方法能够实现多协议攻击的自动化验证。

关键词 多协议攻击,自动化验证,安全属性,Athena算法,逆向搜索
中图法分类号 TP393 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2014.12.024

Automatic Verification for Multi-protocol Attacks by Improving Athena

LIU Wei[1,2] GUO Yuan-bo[1,2] LEI Xin-feng[3] LI Jun-feng[4]
(The PLA Information Engineering University, Zhengzhou 450001, China)[1]
(State Key Laboratory of Mathematical Engineering and Advanced Computing, Zhengzhou 450001, China)[2]
(Unit 61840 of the PLA, Beijing 100097, China)[3] (Taiyuan Satellite Launch Center, Taiyuan 036300, China)[4]

Abstract Protocol security in multi-protocol environments is an open issue in formal analysis for security protocols. Aiming at this problem, an automatic verification for multi-protocol attacks was proposed based on Athena algorithm. The state representation and successor state generation algorithm of Athena are extended, and the attacker can intercept messages from one protocol and insert messages generated by it to another protocol. Some state reduction rules are introduced. The method can verify whether there is a multi-protocol attack. The experiment results show that the method can implement automatic verification for multi-protocol attacks.

Keywords Multi-protocol attacks, Automatic verification, Security properties, Athena, Backwards search

在安全协议形式化分析领域,近些年来出现了很多成功的安全协议形式化分析方法,如BAN逻辑[1]、AVISPA[2]、Paulson归纳法[3]和串空间模型[4]等,这些方法利用数学或计算机理论能够证明协议安全性或发现协议存在的攻击,但一般只限于验证独立运行环境下协议的安全性。在很多实际的协议运行环境中,尤其是在Internet应用环境中,经常会出现多个协议同时运行的场景,不同的协议或相同协议按不同的轮次同时并行运行,这就是所谓的多协议环境。此外,在大型安全协议中,经常会使用多个子协议实现预期的安全目标,从而导致多个子协议的并行运行。在多协议环境中,攻击者能够参与多个协议运行,截取其中协议交互消息和计算生成适合当前协议的消息,从而导致协议面临单独运行时不会出现的攻击。文献[5,6]指出了多协议攻击检测在未来安全协议分析中的必要性和重要性。文献[7]列举了6种多协议攻击实例并总结了阻止多协议攻击的3种方法。

然而,验证协议在多协议环境中的安全属性十分困难,原因在于安全协议的安全属性不具有可组合性。近年来,国内外学者在这方面做了一些研究,但总体上讲,现有工作还存在着一些不足。例如,文献[8]提出了一种自动识别多协议攻击的新方法,但其仅限于发现协议可能出现的类型缺陷攻击,而且不能给出已发现攻击对协议的影响。文献[9]提出了一个多协议攻击自动化检测系统ADMA,系统可以在多协议环境中检测某个协议是否存在多协议攻击,但是仅能检测目标协议与已建协议库中的协议是否存在多协议攻击,从而极大地限制了其应用范围。

Athena[10,11]检测工具是一种基于串空间模型[4]开发的自动化协议分析工具,结合了定理证明方法和模型检验方法。Athena采用了紧凑型状态表示方法和逆向搜索状态空间方法,算法规则较少并且可逆,适合自动搜索并能够有效解决状态空间爆炸问题。本文在分析Athena算法的基础上提出一种多协议攻击自动化验证方法,该方法扩展了Athena算法的状态表示方法,在分析攻击者存在情况下消息的多种构造途径的基础上实现了后继状态生成函数,并通过定义解密序列能够有效解决算法终止性问题。提出的方法能够对任意的多

到稿日期:2014-01-24 返修日期:2014-04-10 本文受国家部委基金项目(9140C130103120C13062)资助。
刘 威(1989-),男,硕士生,主要研究方向为信息与网络安全,E-mail:liuwcy@163.com;郭渊博(1975-),男,博士,副教授,硕士生导师,主要研究方向为信息与网络安全;雷新锋(1973-),男,博士,主要研究方向为信息安全、形式化方法;李俊锋(1972-),男,工程师。

协议系统进行安全性分析，且可以准确地找到攻击者对协议的攻击路径。

本文第1节描述协议验证模型，作为验证算法的基础；第2节介绍状态概念；第3节介绍算法思想，本质上是消息构造的逆向搜索算法；第4节介绍算法的具体步骤；第5节是实验结果与分析；最后总结全文。

1 协议验证模型

Athena算法中定义的协议模型建立在串空间理论上，为了利用模型检测技术和实现协议自动化验证，从两个方面对串空间进行了扩展。一个是提出半丛概念，另一个是提出目标和目标绑定概念。串空间和Athena是著名的形式化分析方法，详细内容可参见文献[4,11]。接下来，根据Athena算法中的协议模型定义一个通用的协议验证模型，该协议验证模型与基于完美加密假设的大多数协议形式化分析与验证方法是兼容的。

协议验证模型里的基本内容是消息项，用来表示协议角色间交互的消息，消息项简称为项。消息项集合Term包括基本项和组合项，其中基本项集合包括Const(表示全局常量集合，包括主体名等)；Fresh(表示新鲜值集合，包括随机数、临时密钥等)；Key(表示长期密钥集合)；Var(表示用于存储接收到的消息项的变量集合)。消息项中组合项集可由加密、连接运算和散列函数运算等操作得到。定义辅助函数 $base$，用于分解连接项：若 $t=(t_1,t_2)$，则 $base(t)=base(t_1)\cup base(t_2)$；否则，$base(t)=t$。

下面给出协议规范的描述方法。在文中的协议验证模型中，以协议事件的形式来静态刻画协议交互中的消息发送和接收行为，把协议角色描述为多个协议事件的序列，整个协议的描述则是所有角色描述的并集。协议事件定义为 $Prevent::=send.I.R.m|receive.I.R.m$，其中，发送协议事件 $send.I.R.m$ 代表协议角色 I 向角色 R 发送消息 m，接收协议事件 $receive.I.R.m$ 表示协议角色 R 收到来自 I 的消息 m。在下文中，用 ev 或者 ev 加下标来表示一个协议事件。

协议规范静态地刻画协议正确的执行流程，但是在协议的实际执行过程中，如果网络环境中存在攻击者，协议的执行很可能会与协议规范不一致。协议的自动化验证正是基于这一点，对合法主体和攻击者所有可能的执行路径进行穷尽搜索，以期找到协议可能存在的错误。因此，需要首先对执行路径进行形式化的描述，然后设计一种搜索算法实现协议的自动化验证。

在文中的协议验证模型中，分别用运行事件和攻击者事件刻画合法主体和攻击者在协议实际执行中可能的行为。攻击者事件定义为 $Intrevent::=encr|decr|app|init|iknows$，包括处理加解密操作的 $encr$ 和 $decr$ 事件、表示函数运算的 app 事件、表示攻击者初始知识的 $init$ 事件和表示攻击者在协议执行中的某个位置获知了某消息的 $iknows$ 事件。这种定义方法便于扩展攻击者能力，使得模型具有很好的扩展性。下面给出运行事件的定义。

在协议实际运行时，一个主体能够并行执行多个协议，而且能够以不同的角色参与到协议的运行。为了区分不同的运行，定义运行标识 id，来代表主体运行的唯一标识。运行标识 id 是全局符号，不同主体的运行和同一主体的多次运行对应的运行标识均不相同。运行事件是在协议事件中加入运行标识 id，这样一个运行事件对应于唯一的运行，利用运行事件能够描述主体运行中的消息发送和接收行为。

运行事件定义为 $Runevent::=id_1.send.A.B.m|id_2.receive.A.B.m$，其中，发送运行事件 $id_1.send.A.B.m$ 表示协议主体 A 在运行 id_1 中向 B 发送消息 m，接收运行事件 $id_2.receive.A.B.m$ 表示协议主体 B 在运行 id_2 中收到来自 A 的消息 m。这里，A 和 B 表示协议的实际执行者，而不是协议描述规范中的角色。

运行事件和攻击者事件通称为状态事件。在文中，用 e 或者 e 加下标等来表示一个状态事件。下一节中将根据状态事件和Athena中的目标绑定关系概念给出状态定义，利用状态能够表示和推导协议所有可能的执行路径。下面给出文中使用到的两个概念。

定义1 函数 in 和 out 是从状态事件到项集合的映射，用于计算状态事件和攻击者知识的关系。函数 in 表示攻击者知识中需要包含使状态事件发生的项；函数 out 表示状态事件 e 如 $send$ 事件执行后攻击者知识中增加的项，也就是说状态事件 e 发生后首次出现的项(term)。

定义2 替换 $\sigma=[t/x]$ 是变量 x 到项 t 的映射。替换项可以包含自由变量。替换 σ 称为项 t_1 和 t_2 的合一，当且仅当 $\sigma(t_1)=\sigma(t_2)$。σ 称为项 t_1 和 t_2 最一般合一，当且仅当对于任意合一 σ'，存在替换 σ''，使得 $\sigma'=\sigma\circ\sigma''$，记为 $\sigma=MGU(t_1,t_2)$。

2 状态

在给出状态定义之前，首先给出目标和目标绑定概念。目标是一个二元组 (e,t)，e 是一个状态事件，t 是消息项，且 $t\in in(e)$。

对于目标 (e,t)，若项 t 在状态事件 e' 的 out 集合中，则称目标 (e,t) 能够被状态事件 e' 绑定。目标 (e,t) 绑定到 e'，记为 $e'\xrightarrow{t}e$。

这里给出的目标和目标绑定的定义与Athena中的目标和目标绑定概念在出发点上是一致的，形式上有所不同。下面给出状态的定义。

定义3 状态是一个三元组 $\langle E,G,\rightarrow\rangle$，其中，$E$ 为状态事件的集合，G 是 E 中未绑定的目标的集合，$\rightarrow$ 是 E 中状态事件的关系。一般用小写字母 q 表示一个状态。

G 中的元素满足下面条件：$\forall e\in E:\forall t\in in(e)$：若 $\rightarrow$ 中不存在关系 $e'\xrightarrow{t}e$，则 $(e,t)\in G$。$\rightarrow$ 是 E 中状态事件的关系，包括无标签的边(记为 $e\rightarrow e'$)和以项 t 作为标签的边(记为 $e\xrightarrow{t}e'$)。无标签边表示属于同一协议运行的运行事件的执行先后顺序关系，带标签边表示状态事件的目标绑定关系，目标绑定关系中的事件若都是运行事件，则应属于不同运行的事件。

给定状态$\langle E,G,\rightarrow\rangle$，定义$\rightarrow^*$是$\rightarrow$(包括无标签边和带标签边)的自反传递闭包。通过$\rightarrow^*$可以得到由无标签边和带标签边共同作用下的状态事件关系。

对于状态事件e和一个替换σ，$\sigma(e)$表示对e中所有的项t利用σ进行替换后得到的状态事件。

定义4 协议P的状态q需要满足下列条件：

(1)目标绑定关系表示消息因果关系：

$$e_1 \xrightarrow{t} e \Rightarrow t \in out(e_1) \wedge t \in in(e)$$

(2)项绑定到最早可能的状态事件：

$$\forall e_1,e_2,e,t:(e_1 \rightarrow^* e_2 \wedge e_2 \xrightarrow{t} e \wedge t \in out(e_1)) \Rightarrow e_1 = e_2$$

(3)对于状态中运行事件e_1，设该运行事件属于运行id_1，那么运行id_1中发生在该运行事件之前的运行事件必须存在，且所有运行事件次序关系与协议P中角色的描述一致，即$\forall e_1 \in Runevent \cap E:(\exists r \in P, ev_2 \in r, \sigma:(e_1 = id_1.\sigma(ev_2) \wedge (\forall ev_3 : ev_3 <_r ev_2 \Rightarrow id_1.\sigma(ev_3) \rightarrow^* e_1)))$，$<_r$表示协议$P$中角色$r$上所有协议事件的次序。

(4)在同一个协议运行中，运行事件是唯一的：

$$(ev \in Protevent, e = id.\sigma(ev) \wedge e' = id.\sigma'(ev)) \Rightarrow e = e'$$

根据状态中的状态事件集合和状态事件关系，可以生成状态的有向无环图。利用状态中的状态事件关系能够推导出状态事件上的偏序关系，进而可以生成协议执行路径中状态事件的全序关系。

定义5 状态可看作协议P路径的筛选，代表与状态匹配的协议路径集。定义函数$trace(P,q)$表示与状态q匹配的协议路径集合：

$trace(P,q)=\{tr \mid tr \in trace(P) \wedge \exists \sigma:((e \in E \Rightarrow \sigma(e) \in tr) \wedge (e_1 \rightarrow e_2 \Rightarrow \sigma(e_1) <_{tr} \sigma(e_2)))\}$，$<_{tr}$表示协议路径中状态事件关系。

在第4节的验证方法中，将协议机密性和认证性验证转化为判断协议的初始状态是否存在与之对应的协议路径，根据状态的定义可知，状态只有满足一定条件才能根据其构造出对应的协议路径。

定义6 对于一些状态，可以构建包含这些状态的协议路径，把这些状态称为可实现状态。可实现状态q满足条件：

$$\forall e_1 \in E: \forall t \in in(e_1): \exists e_2 \in E, t \in out(e_2) \wedge e_2 \rightarrow e_1$$

根据定义6，可实现状态中不包含未绑定的目标，所有状态事件的输入项均存在与之绑定的状态事件。与可实现状态相反，有些状态代表的协议路径集为空，称为空状态。引理1和引理2描述两种典型的空状态，可消除验证算法中的空分支，减少状态搜索空间。

引理1 设q为协议P的状态，若q中存在环，那么q代表的协议路径集为空。形式化表示为：

若存在e_1,e_2,e_3满足$e_1 \rightarrow^* e_2 \xrightarrow{t} e_3 \wedge e_1 \neq e_2 \wedge t \in in(e_1)$，则$trace(P,q)=\emptyset$。

引理2 设q为协议P的状态，项t是一个随机数且状态事件e_1第一次发送t，那么在q中不存在更早的状态事件e_2把t作为子项发送，这是由随机数的性质决定的。形式化表示为：

若存在e_2满足$e_2 \rightarrow^* e_1 \wedge e_2 \neq e_1 \wedge t \in in(e_2) \cup out(e_2)$，则$trace(P,q)=\emptyset$。

3 算法思想

提出的多协议攻击自动化验证方法，可用于验证多协议环境下协议的机密性和认证性。在多协议环境中，由于协议交互的复杂性，攻击者能够利用一个协议的消息攻击另一个协议，并且大型协议各子协议发出的消息可能不会对自身的安全性造成影响，但却能够影响其它子协议的安全性。因此，分析多协议环境下协议的安全性，即验证协议是否存在多协议攻击，验证系统需要具备攻击者截取其它协议消息和计算生成适合当前协议消息的能力。现有的目标绑定方法只能在同一协议内进行目标绑定，文中的方法主要利用扩展目标绑定实现攻击者参与其它协议运行以及截取其中协议交互消息和计算生成适合当前协议消息的能力。下面详细介绍扩展的目标绑定过程。

如果一个状态$q=\langle E,G,\rightarrow\rangle$不是可实现的，则$q$中存在尚未绑定的目标，即$G$不为空。选择目标$(e,t) \in G, e \in E, t \in in(e)$，则$q$中不存在$e' \xrightarrow{t} e$。目标项$t$可能来源于诚实主体发送的消息，或者攻击者利用自己知识通过加解密或其他函数运算生成此消息，这其中隐含了消息的连接和分解操作。分别就目标项的各种构造途径分析相应的目标绑定方法。

途径一，目标项来源于一个发送协议事件中的消息，为了确定目标项的来源，需要搜索协议中所有的发送协议事件。这种情况下，既要考虑当前协议中的发送协议事件，也要考虑来自其它协议的发送协议事件。因此，利用最一般合一判断协议集中所有发送协议事件out集合中是否存在与目标项能够合一的项。若存在，为该发送协议事件分配一个未使用的运行标识符生成对应的运行事件，将目标绑定到该运行事件并生成新的状态。而且，如果状态q中存在该发送协议事件对应的运行事件，也需要将目标绑定到该运行事件并生成新的状态。

途径二，若攻击者知道相关密钥，则通过对发送协议事件中的消息进行多次解密和分解操作能够得到目标项，这种情况下进行目标绑定，需要搜索协议中所有的发送协议事件，针对每个发送协议事件e'，若e'中的消息通过多次解密和分解操作能够得到此目标项，为该发送协议事件e'分配一个未使用的运行标识符生成对应的运行事件，将目标绑定到该运行事件并生成新的状态。而且，如果状态q中存在e'对应的运行事件，也需要将目标绑定到该运行事件并生成新的状态。

途径三，目标项由攻击者通过加密生成。假设t是$\{m\}_k$形式，攻击者知识集合包含项m和k，则可以将t绑定到攻击者加密事件。具体做法是添加加密事件$encr(m,k,\{m\}_k)$到当前状态，将目标绑定到此加密事件，生成新的状态。

途径四，目标项由攻击者通过函数运算生成。假设目标项$t=app(f(t_1,\cdots,t_n))$，函数f在攻击者知识集合中，则可

以将 t 绑定到函数运算事件。具体做法是添加该函数运算事件到当前状态中，将目标绑定到该函数运算事件，生成新的状态。这里，$f(t_1,\cdots,t_n)$限定为可逆函数。

根据目标项的所有构造途径，对状态中的所有目标进行绑定，若最终生成的状态中不包含未绑定的目标，则得到的状态是可实现的。下面分别分析以上 4 种途径：对于途径一，由于协议中发送协议事件是有限的，故可能的发送协议事件必然是有限的，经过有限次搜索能够完成目标绑定；攻击者通过加密操作构造目标项，根据完美加密假设，攻击者需要知道对应的明文和密钥，并且加密事件只有一种可能的形式，因此目标绑定是唯一的；攻击者通过函数运算构造目标项，由于限定攻击者能够执行的函数运算为常用的可逆运算，如随机数自加一或自减一等，因此目标绑定同样是唯一的。对于途径一、途径三、途径四，能够直接将目标绑定到合适的发送协议事件、加密事件或者函数运算事件。

然而，对于途径二，考虑到协议发送事件中的消息可能是嵌套加密的，为了从中得到目标项，可能需要进行多次解密和分解操作，此过程比较复杂，因此，引入解密序列概念，利用解密序列对此过程进行描述。

定义 7 状态 q 是可实现状态，e 是 q 中的状态事件，t 是一个项，$t\in in(e)$。那么存在项 $t_1,t_2,\cdots,t_N$ 和状态事件 $e_1,e_2,\cdots,e_{N+1}$，使得 $e_{N+1}\xrightarrow{t_N}e_N\xrightarrow{t_{N-1}}\cdots\xrightarrow{t_1}e_1\xrightarrow{t}e$，其中，$e_{N+1}$是一个发送运行事件，对于 $\forall i:0<i<N+1,t_i=\{m_i\}_{ki},t_{i-1}\in base(m_i)$。把 $e_{N+1},e_N,\cdots,e$ 称为目标(e,t)的解密序列。

利用解密序列能够合并目标绑定方法中的途径一和途径二，途径一对应于 $N=0$ 时的解密序列。在 MGU 概念的基础上定义最一般解密合一和 $decrs$ 函数，利用两者能够计算出两个项对应的解密序列。

定义 8 σ 是一个替换，Seq 是零或多个消息项的连接。(σ,Seq)称为项 t_1 和 t_2 的解密合一者，记为$(\sigma,Seq)\in DU(t_1,t_2)$，当且仅当：若 Seq 为空，存在替换 σ 使得 $\sigma(t_1)\in base(\sigma(t_2))$；否则 $Seq=Seq'\cdot[\{m\}_k]$，存在替换 σ 使得$\{m\}_k\in base(\sigma(t_2))$并且$(\sigma,Seq')\in DU(t_1,m)$。

解密合一的集合 DS 为 t_1 和 t_2 最一般解密合一，记为 $DS=MGDU(t_1,t_2)$，当且仅当：

(1)对于所有的$(\sigma,Seq)\in DS$，有$(\sigma,Seq)\in DU(t_1,t_2)$；

(2)对于任意解密合一$(\sigma,Seq)\in DU(t_1,t_2)$，存在$(\sigma',Seq')\in MGDU$和替换$\sigma''$，使得 $\sigma'=\sigma\circ\sigma''$。

用函数 $decrs$ 计算项 t_1 和 t_2 的最一般解密合一集合：

$$decrs(t_1,t_2)=\{(\sigma,[])\mid t\in base(t_2)\wedge\sigma=MGU(t_1,t)\}\cup\{(\sigma,Seq'\cdot\{m\}_k)\mid\{m\}_k\in base(t_2)\wedge(\sigma,Seq')\in decrs(t_1,m)\}$$

例如，u 和 v 是基本变量，即两个变量仅能用基本项进行赋值，而不能用组合项进行赋值。项 ni 和$\{u,\{v\}kir\}$的最一般解密合一为：

$$MGDU(ni,\{u,\{v\}kir\})=\{(\{u\rightarrow ni\},[]),(\{v\rightarrow ni\},[\{v\}kir])\}$$

在验证算法中，需要生成目标项和状态事件 out 集合中所有项的解密序列。为了方便，定义函数 $Edecrs$ 用于生成目标项和状态事件 out 集合中项的解密序列。

$$Edecrs(t_1,E)=\{(e,\sigma,Seq)\mid e\in E\wedge t_2\in out(e)\wedge(\sigma,Seq)\in decrs(t_1,t_2)\}$$

4 多协议攻击自动化验证算法

在状态的基础上，提出了多协议攻击自动化验证算法，算法的核心是目标绑定和后继状态生成算法。

4.1 安全属性定义

密码协议运行在一个分布式的系统中，主体通过网络交换消息，协议中的每个主体只能看到自己发送和接收的消息，协议主体依据这些消息，对协议的安全性形成一个局部的认识，并且从不同主体的角度看，协议安全性往往不同。因此，本文从主体的角度对协议的机密性和一致性[12]属性进行验证。

(1)机密性

如果要求 m 具有秘密性，则 m 不能被正常通信者以外的用户得到。秘密性用 $secret(r,m)$表示，具体含义是角色 r 要求 m 具有秘密性。

(2)一致性

一致性表示一个主体想要确认它是否和它所期望的主体进行了协议交互，使用 $wagree(r,r')$表示主体 r 对主体 r'进行的弱一致性验证，$sagree(r,r')$表示主体 r 对主体 r'进行的强一致性验证。

4.2 主算法结构

对于给定的协议集，验证算法会对其中定义的安全属性进行自动化验证。下面介绍验证算法的主要流程：验证算法以协议描述 P 为输入，由协议描述 P 中定义的机密属性和一致性属性，验证算法能够自动地生成初始状态 q_0，然后将 q_0 作为参数传入到递归算法 iterate()进行迭代，该算法最终会输出安全属性的验证结果。

对于机密性，初始状态 q_0 表示违背机密性的所有路径。q_0 构造方法是：若角色 R 中包含待验证的机密性目标，则 q_0 包含角色 R 中所有协议事件对应的运行事件，并且这些运行事件的顺序关系与协议规范定义的一致；q_0 还应包含代表攻击者获知了秘密 t 的事件 $iknows(t)$，以及攻击者的初始知识，包括主体名和所有公钥等。对于一致性，对应的初始状态构造方法类似。

递归算法 iterate()是多协议攻击自动化验证算法的主要算法。在该算法中，iterate()首先利用剪枝函数 prune_theorems()判断当前状态是否符合剪枝条件，若符合剪枝条件，表示由当前状态不能生成可实现状态，中止 iterate()执行；若不符合剪枝条件，表示由当前状态生成可实现状态是可能的，再利用 runcount()判断状态 q 中的运行数是否超过了设置的值 x，若超过了值 x，则输出安全属性有限制验证并中止 iterate()执行，若状态 q 中的运行数不大于值 x，首先从当前状态 q 中选择目标(e,t)，若(e,t)为空，则说明当前状态是可实现状态，利用 property_check()对状态 q 进行检测，给出安全属性的验证结果，下文中将具体说明此函数；若(e,t)不为空，则说明当前状态还包含未绑定的目标，使用第 3 节中叙述的 3 种目标绑定方法对该目标进行绑定(下一小节具体说明这 3 种

目标绑定方法),在目标绑定完成后会生成新的状态 q',然后将 q'传入到 iterate()递归调用该算法。下面是 iterate()算法的伪代码:

```
iterate (P,q)
{
if (! prune_theorems (P,q)){
  if (runcount(q)<=x){
    //从状态 q 中选择一个未绑定的目标
    (e,t)=select_goal (q);
    if ((e,t)==NULL){
      //q 为一个可实现状态,验证安全属性
      property_check (q);
    }
    else{
      //利用所有的目标绑定方法对(e,t)进行绑定
      if ∃m,k:t={m}k{
        q'=q∪{encr(m,k,{m}k)--t-->e};
        iterate (P,q');}
      else if ∃f,t' :t=f(t') {
        q'=q∪{app(f(t'))--t-->e};
        iterate (P,q');}
      for all (e',σ,Seq)∈Edecrs(t,Eqs){
        q'=q∪{e'--Seqn-->…decr(Seqi)…--t-->e};
        iterate (P,q');}
      for all (e',σ,Seq)∈Edecrs(t,ev(P)) {
        q'=q∪{e'--Seqn-->…decr(Seqi)…--t-->e}∪q'';
        iterate (P,q');}
      }
    else{
      output the property is verified limitedly;}
    }
  else
    return;
}
```

property_check()以一个可实现状态 q 为输入,用于对 q 进行检测,给出安全属性的验证结果。对于机密性,由于验证算法针对机密性构造的初始状态表示违背机密性的所有路径,因此协议是不安全的,根据可实现状态 q 构造对应的攻击路径。对于一致性属性,根据一致性属性的定义来检测可实现状态是否满足一致性属性的要求,如果不满足则协议不能实现一致性属性,由可实现状态 q 构造出对应的攻击路径;如果满足一致性属性的要求,只能说明在可实现状态 q 下协议能够实现一致性属性,验证算法不能对一致性属性进行完全验证。

4.3 后继状态生成算法

后继状态生成算法根据目标项的所有构造途径,对状态中的所有目标进行绑定,并生成后继状态。对于一个状态 $q=\langle E_q,G,\rightarrow\rangle$,$G$ 是目标项的集合,若 G 不为空,则 G 中存在还未绑定的目标。从 G 中选择一个未绑定的目标,假设选中(e,t),利用后继状态生成算法对目标进行绑定生成后继状态。

根据目标绑定的 3 种方式,分别说明对状态 q 中的目标(e,t)进行绑定以及生成后继状态 $q'=\langle E_q',G',\rightarrow'\rangle$的过程:

(1)将目标(e,t)绑定发送运行事件。设 E_{qs} 是状态 q 中所有发送运行事件的集合,对于 $\forall(e',\sigma,Seq)\in Edecrs(t,E_{qs})$,$\rightarrow'=\rightarrow\cup\{e'\xrightarrow{Seq_n}\cdots decr(Seq_i)\cdots\xrightarrow{t}e\}$,然后根据$\rightarrow'$更新 E_q'和 G',生成后继状态 q'。

设 $ev(P)$是输入协议集中所有发送协议事件对应的发送运行事件的集合,对于 $\forall(e',\sigma,Seq)\in Edecrs(t,ev(p))$,$\rightarrow'=\rightarrow\cup\{e'\xrightarrow{Seq_n}\cdots decr(Seq_i)\cdots\xrightarrow{t}e\}\cup\rightarrow''$,其中状态事件关系$\rightarrow''$满足:如果 $ev\in role(e')\wedge id.\sigma(ev)=e'$,则 $\forall ev':ev'<_{role(e')}ev\Rightarrow id.\sigma(ev')(\rightarrow'')^*e'$,$id$ 是当前状态 q 中不包含的运行标识符。然后根据$\rightarrow'$更新 E_q'和 G',生成后继状态 q'。

(2)攻击者通过加密生成项 t。如果 t 是$\{m\}_k$ 形式,并且 m 和 k 在攻击者知识集合中,则 $E_q'=E_q\cup\{encr(m,k,\{m\}_k)\}$,$\rightarrow'=\rightarrow\cup\{encr(m,k,\{m\}_k)\xrightarrow{t}e\}$,根据 E_q'和$\rightarrow'$更新 G',生成后继状态 q'。

(3)攻击者通过函数运算生成项 t。若存在 $f(t_1,\cdots,t_n)$使得 $t=f(t_1,\cdots,t_n)$,则 $E_q'=E_q\cup\{app(f(t_1,\cdots,t_n))\}$,$\rightarrow'=\rightarrow\cup\{app(f(t_1,\cdots,t_n))\xrightarrow{t}e\}$,根据 E_q'和$\rightarrow'$更新 G',生成后继状态 q'。

4.4 验证算法的讨论

在安全协议形式化分析领域,一个很重要的形式化分析模型是 Dolev-Yao 模型,在这之后的安全协议自动化验证工具中一般采用 Dolev-Yao 模型来刻画攻击者的能力。本文提出的多协议攻击自动化验证算法本质上也是基于该模型的,验证算法中目标项的几种构造途径都是遵循 Dolev-Yao 模型而来的。

在验证算法中,目标项能够绑定到所有可能的发送协议事件,对应于 Dolev-Yao 模型的攻击者消息转发能力;目标项绑定到攻击者加密和解密事件,对应于攻击者的加解密运算能力;目标的定义以及项的合一定义,从另一种形式实现了攻击者的消息连接和分解能力;目标不仅能够绑定到正常协议交互对应的事件,也能绑定到其他的发送协议事件或者攻击者事件,实现了攻击者丢弃协议交互消息的能力。总之,Dolev-Yao 模型刻画的攻击者能力验证算法均得到实现,并且为了分析多协议攻击,还赋予了攻击者参与其它协议运行以及截取其中协议交互消息和计算生成适合当前协议消息的能力,从而能够保证验证算法的正确性和有效性。而且,本文提出的自动化验证算法能够找到协议可能存在的多个攻击,其他同类型工具如著名的安全协议自动化验证工具 AVISPA 在找到协议存在的一个攻击后就会停止运行,因此本文提出的验证算法能够更好地验证协议的安全性。

5 实验结果与分析

5.1 实验结果

从 SPORE[13] 中选取 9 个协议,利用实现的验证算法对

它们其中两个协议组成的多协议系统进行安全性验证，部分验证结果如表1所列。其中，协议1是被攻击的协议，安全属性是协议1要实现的安全目标，协议2是被攻击者利用来攻击协议1的协议。

表1 多协议攻击检测结果

协议1	安全属性	协议2	是否存在多协议攻击
Andrew-Lowe-BAN	wagree(R,I)	Yahalom	是
Denning-Sacco	wagree(I,R)	Andrew	是
Denning-Sacco	secret(I,kir)	Andrew	否
Denning-Sacco-Lowe	secret(R,kir)	Yahalom	是
KaoChow-3	secret(R,kir)	WooLamPi-1	是
Needham-Schroeder	wagree(R,I)	Yahalom-BAN	是
Needham-Schroeder	secret(R,kir)	Yahalom-BAN	是

从表1可以看出，Andrew-Lowe-BAN和Yahalom协议并行运行时，Andrew-Lowe-BAN协议的wagree(R,I)验证不满足。Denning-Sacco和Andrew协议并行运行时，Denning-Sacco协议的wagree(I,R)满足，而秘密性secret(I,kir)不满足。Needham-Schroeder和Yahalom-Ban协议并行运行时，Needham-Schroeder的安全属性wagree(R,I)和secret(R,kir)均不满足。

经分析发现，造成上述多协议攻击的主要原因是协议2中的消息没有足够的附加信息，导致协议参与者对接收到的消息无法判断消息来源和做出有效检查，这样攻击者就能够利用协议2获得协议1中的主体所期待的加密等消息，进而能够扮演该主体预期的通信者，完成与该主体的协议交互步骤，最终达到获得密钥或者实现非法认证等目的。

5.2 多协议攻击实例

本节对表1中的最后一个多协议攻击作具体说明。

Yahalom-BAN协议描述：

Msg1 $I \rightarrow R: I, ni$

Msg2 $R \rightarrow S: \{R, nr, \{I, ni\}_{k(R,S)}\}$

Msg3 $S \rightarrow I: \{nr, \{R, kir, ni\}_{k(I,S)}, \{I, kir, nr\}_{k(R,S)}\}$

Msg4 $I \rightarrow R: \{\{I, kir, nr\}_{k(R,S)}, \{nr\}_{kir}\}$

Needham-Schroeder协议：

Msg1 $I \rightarrow S: I, R, ni$

Msg2 $S \rightarrow I: \{ni, R, kir, \{kir, I\}_{k(R,S)}\}_{k(I,S)}$)

Msg3 $I \rightarrow R: \{kir, I\}_{k(R,S)}$

Msg4 $R \rightarrow I: \{nr\}_{kir}$

Msg5 $I \rightarrow R: \{dec(nr)\}_{kir}$

利用算法进行机密性验证，需要首先生成相应的初始状态q_0。设$q_0=(E,G,\rightarrow)$，则

$E=\{receive.\ I.\ R.\ \{kir, I\}_{k(R,S)}, send.\ R.\ I.\ \{nr\}_{kir}, receive.\ I.\ R.\ \{dec(nr)\}_{kir}\} \cup \{iknows(kir)\}$；

$\rightarrow=\{receive.\ I.\ R.\ \{kir,I\}_{k(R,S)} \rightarrow send.\ R.\ I.\ \{nr\}_{kir} \rightarrow receive.\ I.\ R.\ \{dec(nr)\}_{kir}\}$；

$G=\{(receive.\ I.\ R.\ \{kir, I\}_{k(R,S)}, \{kir, I\}_{k(R,S)}), (receive.\ I.\ R.\ \{dec(nr)\}_{kir}, \{dec(nr)\}_{kir}), (iknows(kir), kir)\}$

利用实现的验证算法对协议初始状态进行逆向搜索，最终生成可实现状态，再构造出其对应的攻击路径。验证结果显示，Needham-Schroeder协议和Yahalom-BAN协议并行运行时存在多协议攻击，检测到的攻击路径如下：

(1) $fake.\ A.\ C.\ \{A,B\}$；

(2) 2. $receive.\ A.\ C.\ \{A,B\}$；

(3) 2. $send.\ C.\ S.\ \{C, nr2, \{A,B\}_{k(C,S)}\}$；

(4) $intercept.\ C.\ S.\ \{C, nr2, \{A,B\}_{k(C,S)}\}$；

(5) $fake.\ B.\ C.\ \{A,B\}_{k(C,S)}$；

(6) 1. $receive.\ B.\ C.\ \{A,B\}_{k(C,S)}$；

(7) 1. $send.\ C.\ B.\ \{nr1\}_A$；

(8) $intercept.\ C.\ B.\ \{nr1\}_A$；

(9) $fake.\ B.\ C.\ \{dec(nr1)\}_A$；

(10) 1. $receive.\ B.\ C.\ \{dec(nr1)\}_A$。

在这个攻击中，步骤(1)表示在Yahalom-BAN中攻击者冒充主体A向C发送消息$\{A,B\}$，步骤(2)表示主体C接收了A发来的消息$\{A,B\}$，并按照Yahalom-BAN的协议步骤生成随机数$nr2$并向S发送了消息$\{C, nr2, \{A,B\}_{k(C,S)}\}$(步骤(3))，步骤(4)和(5)表示攻击者截获了这条消息，并冒充Needham-Schroeder协议的主体B向C发送消息$\{A, B\}_{k(C,S)}$，在步骤(6)中主体C接收了此消息，并按照Needham-Schroeder的协议步骤生成了随机数$nr1$并向B发送了消息$\{nr1\}_A$(步骤(7))，步骤(8)和(9)表示攻击者截获了此消息，并冒充B向C发送了消息$\{dec(nr1)\}_A$，在步骤(10)中主体C接收了此消息，并认为它与主体B完成了Needham-Schroeder协议，但实际上B完全没有参与该协议交互。攻击者利用Yahalom-BAN协议产生了Needham-Schroeder协议中主体C期望的消息，并与C完成了Needham-Schroeder协议。

5.3 算法复杂度分析

状态空间爆炸问题是安全协议自动化验证工具的固有问题，模型状态空间随并行系统规模的增长呈指数增长，当并行系统的规模超过一定范围时，状态空间超出可检测的范围，从而导致分析不彻底。而且，由于本文验证算法的目标是能够分析验证多协议系统的安全性，多协议系统涉及到多个协议，形成的状态空间更加庞大，因此在算法实现中需要尽可能地降低算法的空间复杂度。

分析目标绑定过程，可以发现此过程中状态的改变主要是因为添加了新的目标和新创建的运行，因此，在算法中只定义了一个全局状态q，从而能够极大地降低算法执行时所占用的存储空间。为了在iterate()执行完成后恢复到调用iterate()前的状态，使用的解决方案是针对上述目标绑定过程对状态做的改变，相应地修改全局状态q以恢复到调用iterate()算法前的状态，具体要进行的操作是移除目标绑定过程中添加的目标，以及从状态中移除目标绑定过程新创建的运行等。

在算法时间复杂度方面，通过验证SPORE协议库中的12个安全协议20个安全属性，将系统运行数分别设置为2—7，统计出单个安全属性的平均验证时间，如表2所列。由表2得出，在运行数不大于5时，平均验证时间小于1s，在运行数为7时，平均验证时间是4.9s。结果表明，实现的验证算法具有良好的运行效率。

(下转第132页)

LNCS 8043. Berlin:Springer,2013:1-17

[7] Yan Huang,Katz J,Evans D. Efficient Secure Two-Party Computation Using Symmetric Cut-and-Choose[C]// Advances in Cryptology-Crypto,LNCS 8043. Berlin:Springer,2013:18-35

[8] 孙茂华,罗守山,辛阳,等. 安全两方线段求交协议及其在保护隐私凸包交集中的应用[J]. 通信学报,2013,34(1):30-42

[9] 李顺东,戴一奇,游启友. 姚氏百万富翁问题的高效解决方案[J]. 电子学报,2005,33(5):769-773

[10] Li Shun-dong,Wang Dao-shun,Dai Yi-qi,et al. Symmetric cryptographic solution to Yao's millionaires' problem and an evaluation of secure multiparty computations[J]. Information Sciences,2008,178(1):244-255

[11] Cachin C. Efficient private bidding and anctions with an oblivious third party[C]// 6th ACM Conference on Computer and Communications Security. Singapore,1999:120-127

[12] Li Rong-hua, Wu Chuan-kun, Zhang Yu-qing. A fair and efficient protocol for the millionaires' problem[J]. Chinese Journal of Electronics,2009,18(2):249-254

[13] Gordon S D,Hazay C,Katz J,et al. Complete fairness in secure two-party computation[C]//40th ACM Synmposium on Theory of Computing (STOC). New York:ACM Press,2008:413-422

[14] Pinkas B. Fair secure two-party computation[C]// In Advances in Cryptology-Eurocrypt 2003, LNCS 2656. Berlin: Springer, 2003:87-105

[15] Garay J,MacKenzie P,Prabhakaran M,et al. Resource Fairness and Composability of Cryptographic Protocols[C]// Proc of the 3rd Theory of Cryptography Conference (TCC), LNCS 3876. Berlin:Springer,2006:404-428

[16] Halpern J, Teague V. Rational Secret Sharing and Multiparty Computation[C]// Proceedings of the 36th Annual ACM Symposium on Theory of Computing (STOC). New York: ACM Press,2004:623-632

[17] 张恩,蔡永泉. 基于双线性对的可验证的理性秘密共享方案[J]. 电子学报,2012,40(5):1050-1054

[18] Zhang E, Cai Y Q. Rational Multi-Secret Sharing Scheme in Standard Point-to-Point Communication Networks[J]. International Journal of Foundations of Computer Science,2013,24(6):879-897

[19] Zhang E,Cai Y Q. Collusion-free Rational Secure Sum Protocol [J]. Chinese Journal of Electronics,2013,22(3):563-566

[20] Zhang Z F,Liu M L. Rational secret sharing as extensive games [J]. Science China Information Sciences,2013,56(3):1-13

[21] Tian Y L,Ma J F,et al. Fair (t,n) threshold secret sharing scheme[J]. IET Information Security,2013,7(2):106-112

[22] 谢识予. 经济博弈论(第二版)[M]. 上海:复旦大学出版社,2002:138-158

(上接第 117 页)

表 2 单个安全属性平均验证时间

系统运行数	平均验证时间
2	0.003s
3	0.016s
4	0.07s
5	0.27s
6	1.20s
7	4.90s

结束语 本文描述的协议自动化验证方法,能够用于多协议环境下协议安全性验证。验证算法思想来源于 Athena 算法,在分析攻击者存在情况下消息的多种构造途径的基础上,提出了消息构造的逆向搜索方法,用以准确地找到攻击者对协议的攻击路径。通过改进消减规则和采用新的方法处理攻击者知识推导,算法具有良好的运行效率,能够实现多协议攻击的自动化验证。下一步工作是引入其它状态削减规则,进一步提高算法效率,并检测更多的多协议攻击。

参 考 文 献

[1] Burrows M,Abadi M,Needham R. A logic of authentication[J]. Mathematical and Physical Sciences,1989,426(1871):233-271

[2] Vigano L. Automated Security Protocol Analysis With the AVISPA Tool[J]. Electronic Notes in Theoretical Computer Science,2006,155:61-86

[3] Paulson L C. The inductive approach to verifying cryptographic protocols[J]. Journal of computer security,1998,6(1):85-128

[4] Fábrega F J T,Herzog J C,Guttman J D. Strand spaces:Proving security protocols correct[J]. Journal of computer security, 1999,7(2):191-230

[5] Bella G. What is correctness of security protocols? [J]. Journal of Universal Computer Science,2008,14(12):2083-2106

[6] Khoury P,Hacid M,Sinha S K,et al. A Study on recent trends on integration of security mechanisms[M]//Ras Z W,Dardzinska A. Advances in Data Management. Berlin: Springer-Verlag, 2009:203-224

[7] Mathuria A,Singh A R,Sharavan P V,et al. Some new multiprotocol attacks[C]// Proc of the 15th Int Conf on Advanced Computing and Communications. Washington: IEEE Computer Society Press,2007:465-471

[8] Genge B,Haller P. A Syntactic Approach for Identifying Multi-Protocol Attacks[C]// Ultra Modern Telecommunications and Workshops. Washington:IEEE Computer Society Press,2009:1-5

[9] 杨元原,马文平,刘维博,等. 有效的多协议攻击自动化检测系统[J]. 重庆大学学报,2012,35(2):71-77

[10] Song D,Perrig A,Berezin S. Athena:a novel approach to efficient automatic security protocol analysis[J]. Journal of Computer Security,2001,9(1):47-74

[11] Song D. An Automatic Approach for Building Secure Systems [D]. Berkeley:University of California at Berkeley,2002

[12] Lowe G. A hierarchy of authentication specifications[C]// Proc of The 10th Computer Security Foundations Workshop. Washington:IEEE Computer Society Press,1997:31-43

[13] Security protocols open repository[EB/OL]. 2012-02-11[2013-11-17]. http://www.lsv.ens-cachan.fr/Software/spore/table.html