ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.17, No.4, April 2006, pp.898−906
DOI: 10.1360/jos170898

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

SPVT:一个有效的安全协议验证工具*

李梦君[1+], 李舟军[2], 陈火旺[1]

[1](国防科学技术大学 计算机学院,湖南 长沙 410073)

[2](北京航空航天大学 计算机科学与工程学院,北京 100083)

SPVT: An Efficient Verification Tool for Security Protocol

LI Meng-Jun[1+], LI Zhou-Jun[2], CHEN Huo-Wang[1]

[1](School of Computer, National University of Defense Technology, Changsha 410073, China)

[2](School of Computer Science and Engineering, BeiHang University, Beijing 100083, China)

+ Corresponding author: Phn: +86-731-4576468, E-mail: mengjun_li1975@yahoo.com.cn

Li MJ, Li ZJ, Chen HW. SPVT: An efficient verification tool for security protocol. *Journal of Software*, 2006,17(4):898−906. http://www.jos.org.cn/1000-9825/17/898.htm

Abstract: This paper describes the security protocol verifier SPVT developed by Objective-Caml. In SPVT (security protocol verifying tool), the specification language is the π-like calculus extended with three appendixes, the Dolev-Yao model is described with Horn logic rules, the π-like calculus model of security protocol is transformed into the logic program model by abstract rules, the security properties are verified based on the calculus of the logic program's fixpoint, and the counter-examples on security properties are constructed from the process of the fixpoint calculus and the process of the property verification. The simplified Needham-Schroeder public-key authentication protocol is used to exemplify the automatic verification process of security protocol with SPVT, and the results show the validity of the verifier.

Key words: formal verification; security protocol; logic program

摘 要: 描述了基于 Objective Caml 开发的一个安全协议验证工具 SPVT(security protocol verifying tool).在 SPVT 中,以扩展附加项的类π演算作为安全协议描述语言,以扩展附加项的 Horn 逻辑规则描述协议攻击者的 Dolev-Yao 模型,通过一组抽象规则将安全协议的类π演算模型转换为逻辑程序模型,基于安全协议逻辑程序的不动点计算验证安全性质,从安全协议逻辑程序的不动点计算和安全性质的验证过程中构造不满足安全性质的安全协议反例.以简化的 Needham-Schroeder 公钥认证协议为例,描述了使用 SPVT 自动验证安全协议的过程,表明了 SPVT 用于安全协议验证的有效性.

关键词: 形式化验证;安全协议;逻辑程序

中图法分类号: TP309 文献标识码: A

安全协议使用加密技术实现开放互联网络的通信安全.协议攻击者的破坏和安全协议无穷会话的交叠运行,使得安全协议往往无法实现其设计目的.安全协议的形式化分析与验证是一项十分有意义的研究工作,并成为形式化研究领域的一个研究热点.协议攻击者的破坏和协议无穷会话的交叠,使得协议的手工验证十分复

* Supported by the National Natural Science Foundation of China under Grant Nos.60073001, 60473057, 90104026 (国家自然科学基金); the National High-Tech Research and Development Plan of China under Grant No.2002AA144040 (国家高技术研究发展计划(863))

Received 2004-09-28; Accepted 2005-11-08

杂.SPVT 是基于 Objective Caml 开发的一个安全协议验证工具,支持安全协议的自动验证和反例的自动生成.

安全协议本质上是分布式并发程序,使用类π演算将其描述为多个角色进程的并发合成.SPVT 中定义了一个类π演算,为每一个消息附加了 3 个项,用于标识消息的发送者、接收者和该消息是否为诚实的参与者发送或接收.模型检验方法能够自动验证安全协议的有穷会话和自动构造反例[8−13],使用抽象方法将安全协议多角色进程的并发合成转化为一组逻辑规则,通过计算安全协议逻辑程序的不动点能够对安全协议无穷会话的交叠进行验证[1−3].基于自定义的抽象规则,SPVT 自动地将安全协议多角色进程的并发合成转化为一组逻辑规则.同时,基于安全协议的建模方法,SPVT 能够从安全协议的验证过程自动构造不满足安全性质的安全协议反例,有助于辨识使用抽象方法导致的虚假攻击,有助于校正安全协议[5,7].

在基于模型检验的方法中,安全协议反例一般用踪迹描述,踪迹是输入动作和输出动作的交叠序列.文献[4]中,安全协议及其攻击用标准方式描述,例如简化的 Needham-Schroeder 公钥认证协议,采用标准方式描述如下:

$$A\rightarrow B:\{N_A,A\}_{pub(kB)};\quad B\rightarrow A:\{N_B,N_A\}_{pub(kA)};\quad A\rightarrow B:\{N_B\}_{pub(kB)}.$$

它的攻击采用标准方式描述如下:

$$A\rightarrow I:\{N_A,A\}_{pub(kI)};\quad I(A)\rightarrow B:\{N_A,A\}_{pub(kB)};\quad B\rightarrow A:\{N_B,N_A\}_{pub(kA)};$$
$$A\rightarrow I:\{N_B\}_{pub(kI)};\quad I(A)\rightarrow B:\{N_B\}_{pub(kB)}.$$

在标准描述方式中,每一个消息都给出了消息的实际(期待)发送者和实际(期待)接收者.显然,标准描述方式比基于踪迹的描述方式更为直观和简洁.

本文第 1 节给出 SPVT 中描述安全协议的类-π演算和协议攻击者的模型,同时给出保密性和认证性的描述.第 2 节将给出 SPVT 中的抽象规则,并描述 SPVT 中安全协议逻辑程序不动点的计算和安全性质的验证过程.第 3 节描述 SPVT 中安全协议反例的构造算法.第 4 节描述了使用 SPVT 对简化的 Needham-Schroeder 公钥认证协议自动验证的过程.第 5 节对全文进行总结.

1 安全协议描述语言与安全性质

1.1 安全协议描述语言及其语义

使用类π演算可以将安全协议描述为多角色进程的并发合成[6],SPVT 中安全协议描述语言如图 1 所示,描述语言的语义如图 2 所示..

tag::=	*Tag*
true,false	*Bool*
M,*N*,*U*,*V*,*S*,*T*::=	*Terms*
x, *y*, *z*	*Variable*
$a[M_1,...,M_n]$	*Name*
$f(M_1,...,M_n)$	*Function*

P,*Q*::=	Process
$\overline{c}\langle role(\langle M,N,tag\rangle,M')\rangle.P$	Output process
$c(role(\langle M,N,tag\rangle,x)).P$	Input process
0	Nil process
P|*Q*	Parallel process
!*P*	Replica process
(va)P	Restriction process
let $x=g(M_1',...,M_n')$ in *P*	Destructor process
if *M*=*N* then *P*	Condition process
begin(*M*,*M*′).*P*	Begin event
end(*M*,*M*′).*P*	End event
begin_*ex*(*M*,*M*′)	Begin_*ex* event
end_*ex*(*M*,*M*′)	End_*ex* event

Fig.1 The specification language of security protocol

图 1 安全协议描述语言

与文献[3]中的描述语言的区别在于:输出动作进程$\overline{c}\langle role(\langle M,N,tag\rangle,M')\rangle.P$和输入动作进程$c(role(\langle M,N,tag\rangle,x)).P$中附加了 3 个项.在输出动作进程$\overline{c}\langle role(\langle M,N,tag\rangle,M')\rangle.P$中 M 和 N 分别标识消息 M'的实际发送者和理想接收者,tag 取值为 true 表明消息 M'由安全协议的诚实参与者发送;在输入动作进程$c(role(\langle M,N,tag\rangle,x)).P$中 M 和 N 分别标识消息 x 的理想发送者和实际接收者,tag 取值为 true 表明消息 x 被安全协议的诚实参与者接收.若三元组$\langle M,N,tag\rangle$中的 M 和 N 取值为变量,则分别表示任意的发送者或接收者.协议攻

击者可以是一个安全协议会话中的诚实参与者,在 SPVT 中,协议攻击者被隐含地描述为安全协议会话中的诚实参与者.

其中,=是π演算中的结构同余.使用输入动作和输出动作中的 3 个附加项可以判定一个消息的实际(理想)发送者和实际(理想)接收者.例如,设 $P_0=\overline{c}\langle role(\langle M_1,N_1,tag_1\rangle,M')\rangle.P$ 和 $Q_0=c(role(\langle M_2,N_2,tag_2\rangle,x)).Q$ 是任意两个闭进程,则 P_0 中的输出动作和 Q_0 中的输入动作的交互使用标准方式可以描述如下:

if $tag_1\vee tag_2$=false,则 $statement=\varepsilon$(安全协议攻击者是实际的发送者和接收者);

if $tag_1\vee tag_2\neq$false,则 $statement=M_1(M_2)\rightarrow N_2(N_1):M'$.

$$\frac{}{\overline{c}\langle role(\langle M,N,tag\rangle,M')\rangle.P|c(role(\langle M,N,tag\rangle,x)).Q\rightarrow P|Q\theta,\theta=[M'/x]}$$

$$\frac{}{\text{let } x=g(M'_1,\ldots,M'_n) \text{ in } P\rightarrow P\theta,\theta=[M/x], \text{ if } g(M'_1,\ldots,M'_n)=M}$$

$$\frac{}{!P\rightarrow P|!P} \qquad \frac{P\rightarrow Q}{P|R\rightarrow Q|R} \text{ then } \qquad \frac{P\rightarrow Q}{(va)P\rightarrow(va)Q} \text{ then}$$

$$\frac{P\rightarrow Q,P\equiv P',Q\equiv Q'}{P'\rightarrow Q'} \text{ then } \qquad \frac{}{\text{begin}(M,M').P\rightarrow \text{begin}\_ex(M,M')|P} \qquad \frac{}{\text{end}(M,M').P\rightarrow \text{end}\_ex(M,M')|P}$$

Fig.2 The semantics of the specification language

图 2 描述语言的语义

1.2 攻击者模型与安全协议的安全性质

Dolev-Yao 模型[11]是安全协议形式化分析与验证研究中使用最广泛的协议攻击者模型,协议攻击者完全控制网络,它能够截获、转发、伪造开放互联网络上传送的消息;能够使用其关于安全协议的初始知识集合和截获的消息通过组合、分解、加密和解密等操作产生新的消息.Dolev-Yao 模型一般有两种描述方式,一种采用逻辑规则的描述方式,将协议攻击者的推理能力用逻辑规则的形式描述.在 SPVT 的协议攻击者模型中,每一个以 attacker 作为谓词符号的原子公式中都添加了 3 个原子消息 M,N,tag,具体地说:

对于安全协议初始知识集合中的每一个项 mes,对应一条逻辑规则:

$\rightarrow attacker(role(\langle host(k_I[]),host(v),\text{false}\rangle,mes))$;

对于安全协议中每一个 n 元构造算子 f,对应一条逻辑规则:

$attacker(role(\langle host(v_1),host(k_I[]),\text{false}\rangle,x_1))\wedge\ldots\wedge attacker(role(\langle host(v_n),host(k_I[]),\text{false}\rangle,x_n))\rightarrow attacker(role(\langle host(k_I[]),host(v),\text{false}\rangle,f(x_1,\ldots,x_n)))$;

对于安全协议中每一个 n 元析构算子 g 的每一个消减式 $g(M_1,\ldots,M_n)=M$,对应一条逻辑规则:

$attacker(role(\langle host(v_1),host(k_I[]),\text{false}\rangle,M_1))\wedge\ldots\wedge attacker(role(\langle host(v_n),host(k_I[]),\text{false}\rangle,M_n))\rightarrow attacker(role(\langle host(k_I[]),host(v),\text{false}\rangle,M))$;

协议攻击者的另外一种描述方式是将协议攻击者视为满足以下条件的任意闭子进程 Q,其中 S 是安全协议的初始知识集合,$fn(Q)$是进程 Q 的自由名集合:

(1) $fn(Q)\subseteq S$;

(2) 若 $\overline{c}\langle role(\langle M,N,tag\rangle,M')\rangle.P$ 出现于 Q 中,则 $M=host(k_I[]),tag=$false;若 $c(role(\langle M,N,tag\rangle,x)).P$ 出现于 Q 中,则 $N=host(k_I[]),tag=$false.

满足上述两个条件的任意闭子进程 Q 称为攻击者进程.安全协议的保密性和认证性可以描述为:

定义 **1**(保密性). 设 P 是安全协议多角色进程的并发合成,对 P 中每一个需保密的名 $secret$,若对任意一个攻击者进程 Q,均不存在进程 R 和 R',使得 $P|Q(\equiv\cup\rightarrow)^*\overline{c}\langle role(\langle M,N,tag\rangle,secret)\rangle.R|R'$,则称安全协议满足保密性.

定义 **2**(认证性). 设 P 是安全协议多角色进程的并发合成,如果对任意一个攻击者进程 Q,$P|Q(\equiv\cup\rightarrow)^*R'$,如果对每一个出现于 R'中的 end_$ex(M,M')$,begin_$ex(M,M')$也一定出现于 R'中,则称安全协议满足认证性.

2 抽象解释与安全性质验证

2.1 抽象解释

使用抽象方法,安全协议多角色进程的并发合成将转化为一组逻辑规则.下面是 SPVT 中的抽象规则:

$([[0]]\rho h)=\varphi$ $([[P|Q]]\rho h)=([[P]]\rho h)\cup([[Q]]\rho h)$

([[!P]]ρh)=([[P]](ρ[$i\to i$])h)$\cup$([[Q]]ρh) (i 是引入的会话标识符变量)

([[(va)P]]ρh)=([[P]]ρ[$a\to a$[$\rho(V_O)$,$\rho(V_S)$]])h)

([[c($role$($\langle M,N,tag\rangle$,x)).P]]ρh)=([[P]](ρ[$x\to x$])($h\wedge attacker(role(\langle\rho(M),\rho(N),tag\rangle,x))$)

([[$\bar{c}$ $\langle role(\langle M,N,tag\rangle,M')\rangle$.$P$]).$P$])$\rho h$)=([[$P$]]$\rho h$)$\cup$($h\to attacker(role(\langle\rho(M),\rho(N),tag\rangle,\rho(M'))))$

([[let $x=g(M_1,\ldots,M_n$ in $P)$]ρh]=$\cup$([[P]]($(\sigma\rho)$[$x\to\sigma' p'$])(σh))|如果 $g(M_1',\ldots,M_n')=p'$,并且 $\sigma\rho(M_1,\ldots,M_n)=(M_1',\ldots,M_n')\sigma'$,$Var(M_1,\ldots,M_n)\cap Var(M_1',\ldots,M_n')=\varphi$)

([[begin(M,M').P]]ρh)=([[P]]$\rho(h\wedge$begin(M,$\rho(M')$)) ([[end(M,M').P]]ρh)=([[P]]ρh)$\cup$($h\to$end(M,$\rho(M')$)).

本文的抽象规则与文献[3]中的抽象规则基本相同.但是,本文的抽象规则在以 *attacker* 作为谓词符号的每个原子公式中添加了 3 项.抽象规则将每个约束名 a 抽象为一个函数 $a[\rho(V_O),\rho(V_S)]$,其中 V_O 和 V_S 分别表示约束名 a 所在的角色进程中先于 a 出现的输入动作进程中引入的变量集合以及复制进程中引入的会话标识符集合.安全协议多角色进程的并发合成模型抽象得到的一组逻辑规则和 Dolev-Yao 模型对应的一组逻辑规则构成安全协议逻辑程序.经过抽象,安全协议保密性和认证性的验证转化为原子公式关于安全协议逻辑程序是否逻辑可推导的判定问题.

2.2 安全性质验证

定义 **3**(目标). 在安全协议逻辑规则前件中,形如 begin(M,M')和 $attacker(role(\langle M,N,tag\rangle,x))$($x$ 为任意变量)的原子公式称为假目标;形如 end(M,M')和 $attacker(role(\langle M,N,tag\rangle,M'))$($M'$不为变量)的原子公式称为目标.若 begin($M$,$M'$),end($M$,$M'$)和 $attacker(role(\langle M,N,tag\rangle,M'))$的项 M'中没有变量,则称其为闭原子公式.

定义 **4**(目标的解和正规解). 若逻辑规则 $H\to C$ 满足:H 中的原子公式都是假目标,则称 $H\to C$ 是一个解形式逻辑规则,用 SolvedForm 表示具有所有解形式逻辑规则构成的集合,用 UnSolvedForm 表示不具有解形式的所有逻辑规则构成的集合.

定义 **5**(规则蕴涵). 设 $R_1=H_1^1\wedge\ldots\wedge H_m^1\to C_1$ 和 $R_2=H_1^2\wedge\ldots\wedge H_n^2\to C_2$ 是两条逻辑规则,$C_1=attacker(role(\langle M_1,N_1,tag_1\rangle,M^1))$,$C_2=attacker(role(\langle M_2,N_2,tag_2\rangle,M^2))$,或者 C_1=end(M,M^1),C_2=end(M,M^2),则 R_1 逻辑蕴涵 R_2,记作 $R_1\Rightarrow R_2$,当且仅当:存在置换 θ,使得 $M^1\theta=M^2$,并且对每一个 $H_i^1=attacker(role(\langle M_i^1,N_i^1,tag_i^1\rangle,M_i'))\in\{H_1^1,\ldots,H_m^1\}$,存在 $H_j^2=attacker(role(\langle M_j^2,N_j^2,tag_j^2\rangle,M_j''))\in\{H_1^2,\ldots,H_n^2\}$,对每一个 H_i^1=begin(M, M_i')$\in\{H_1^1,\ldots,H_m^1\}$,存在 H_j^2=begin(M, M_j'')$\in\{H_1^2,\ldots,H_n^2\}$,使得 $M_i'\theta=M_j''$,称置换 θ 为 $R_1\Rightarrow R_2$ 的蕴涵置换.

定义 **6**(逻辑可推导). 设 F 是一个闭原子公式,B 是一组逻辑规则,F 关于 B 逻辑可推导当且仅当存在满足以下条件的一棵有限树:

(1) 除了根结点以外,每一个结点均用 B 中的一条逻辑规则标记,并且每一条边均用一个闭原子公式标记;

(2) 若树中的一个结点用 R 标记,结点的入边用闭原子公式 F_0 标记,n 条出边分别用闭原子公式 $F_1,\ldots,F_n$ 标记 ,则 $R\Rightarrow F_1\wedge\ldots\wedge F_n\to F_0$;

(3) 根结点有唯一一条出边并且用闭原子公式 F 标记.

满足上述条件的有限树称为 F 关于 B 的逻辑可推导树.在上述定义中,如果每一条边均用原子公式标记,而不要求都用闭原子公式标记,则称 F 关于 B 逻辑弱可推导,对应的有限树称为 F 关于 B 的逻辑弱可推导树.容易证明,F 关于 B 逻辑弱可推导,则 F 关于 B 逻辑可推导.

定义 **7**(消解). 设 $R=H\to F$ 和 $R'=H'\to F'$是两条逻辑规则,如果 $F=attacker(role(\langle M_1,N_1,tag_1\rangle,M_1'))$,并且 H'中存在满足 M_1'能够与 M_2'进行合一化的原子公式 $F_0=attacker(role(\langle M_2,N_2,tag_2\rangle,M_2'))$,如果 F=end(M_1, M_1'),并且 H'中存在满足 M_1'能够与 M_2'进行合一化的原子公式 F_0=end(M_1, M_2'),则 R 与 R'的消解,记作 $R'\cdot R$,定义为 $(H\wedge(H'-F_0))\theta\to F'\theta$,其中 $\theta=mgu(M_1',M_2')$是 M_1'和 M_2'的最一般合一化算子,并且称 F_0 是 R'的选择原子,记作 $F_0=selectedAtom(R')$,称 θ是 $R'\cdot R$ 的消解置换,记作 $\theta=sub(R,R')$.

定义 **8**(***X***-消解). 设 $R=H\to F$ 和 $R'=H'\to F'$是两条逻辑规则,$R\in SolvedForm$,$R'\in UnSolvedForm$,$F=attacker(role(\langle M_1,N_1,tag_1\rangle,M_1'))$,$F_0=attacker(role(\langle M_2,N_2,tag_2\rangle,M_2'))$是 H'中满足 M_1'能够与 M_2'进行合一化的目标,则 R

与 R'的 X-消解,记作 $R'\circ R$,定义为$(H\wedge(H'-F_0))\theta\rightarrow F'\theta$,其中$\theta=mgu(M_1', M_2')$是 M_1'和 M_2'的最一般合一化算子,F_0称为 R'的选择目标,记作 $F_0=selectedGoal(R')$,θ称为 $R'\circ R$ 的消解置换,记作$\theta=sub(R,R')$.

设 R 是一条逻辑规则,B 是一个逻辑规则集合,定义 $addRule(R,B)$如下:

If $\exists R'\in B, R'\Rightarrow R$, then $addRule(R,B)=B$;

else $addRule(R,B)=\{R\}\cup\{R'|R'\in B, R\not\Rightarrow R'\}\cup\{marked(R'')|R''\in B, R\Rightarrow R''\}$

并且定义 $addRule(\{R_1,...,R_n\},B)=addRule(\{R_2,...,R_n\},addRule(R_1,B))$.其中 $marked(R'')$表示逻辑规则 R'' 不再参与计算 X-消解.设 $R=F_1\wedge...\wedge F_n\rightarrow C$ 表示一个逻辑规则,$F_i=attacker(role(\langle M_i,N_i,tag_i\rangle, M_i'))(i=1,...,n)$,定义 $elimdup(R)$为满足以下条件的逻辑规则 R':

(1) 如果对任意的 $j<i$, $M_j'\neq M_i'$,则 F_i 是 R'逻辑前件中的一个原子公式;

(2) C 是 R'的逻辑后件.

设 P 是安全协议逻辑程序,归纳定义:

$Rule^{(0)}(P)=\{elimdup(R)|R\in P\}$

$T^{(0)}(P)=Rule^{(0)}\cap SolvedForm$ $C^{(0)}(P)=Rule^{(0)}\cap UnSolvedForm$

$X\_Resolution^{(0)}(P)=\{elimdup(R)|R=R''\circ R', R'\in T^{(0)}(P), R''\in C^{(0)}(P)\}$

$Rule^{(n+1)}(P)=addRule(X\_Resolution^{(n)}(P), Rule^{(n)}(P))$

$T^{(n+1)}(P)=Rule^{(n+1)}(P)\cap SolvedForm$ $C^{(n+1)}(P)=Rule^{(n+1)}(P)\cap UnSolvedForm$

$X\_Resolution^{(n+1)}(P)=\{elimdup(R)|R=R''\circ R', R'\in T^{(n)}(P), R''\in C^{(n)}(P)\}$

定义 **9**(解形式不动点). 设 P 是安全协议逻辑程序,定义 $fixpoint(P)=\{T^{(n)}(P)|n\geq 0\}\cap UnMarked$, $fixpoint(P)$称为 P 的解形式不动点.

设 P 是安全协议逻辑程序,R 表示一个逻辑规则,B 是一个逻辑规则集合,对于保密性,定义 $derivablerec(R,B,P)$如下:

if $\exists R'\in B, R'\Rightarrow R$, then $derivablerec(R,B,P)=\varnothing$

else if $R=\rightarrow C$, then $derivablerec(R,B,P)=\{\rightarrow C\}$

else $derivablerec(R,B,P)=\{derivablerec(elimdup(R'\cdot R),\{R\}\cup B,P)|R'\in fixpoint(P)\}$

对于认证性,定义$derivablerec(R,B,P)$如下:

if $\exists R'\in B, R'\Rightarrow R$, then $derivablerec(R,B,P)=\varnothing$

else if $R=\text{begin}(M_1, M_1')\wedge...\wedge\text{begin}(M_n, M_n')\rightarrow\text{end}(M,M')$, then $derivablerec(R,B,P)=\{R\}$

else $derivablerec(R,B,P)=\cup\{derivablerec(elimdup(R'\cdot R),\{R\}\cup B,P)|R'\in fixpoint(P)\}$

对于形如 $attacker(role(\langle M,N,tag\rangle,M'))$和 $\text{end}(M,M')$的闭原子公式 F,定义 $derivable(F,P)=derivablerec(F\rightarrow F,\varnothing,P)$.

定理 **1**. 设 P 是安全协议逻辑程序,则:(1) 保密性:若 F 是形如 $attacker(role(\langle M,N,tag\rangle,M'))$的闭原子公式,则 F 关于 P 逻辑可推导当且仅当 F 关于 $fixpoint(P)$逻辑可推导;(2) 认证性:若 F 是形如 $\text{end}(M,M')$的闭原子公式,则 F 关于 $P\cup B_b$ 逻辑可推导,并且 $\text{begin}(M,M')\in B_b$,当且仅当 F 关于 $fixpoint(P)\cup B_b$ 逻辑可推导,并且 $\text{begin}(M,M')\in B_b$.其中,$B_b=\{\rightarrow\text{begin}(M_1, M_1'),\ldots,\rightarrow\text{begin}(M_n, M_n')\}$,$\text{begin}(M_i, M_i')$是以 begin 作为谓词符号的原子公式.

定理 **2**. 设 P 是安全协议逻辑程序,则:(1) 保密性:若 F 是形如 $attacker(role(\langle M,N,tag\rangle,M'))$的闭原子公式,则 F 关于 $fixpoint(P)$逻辑可推导当且仅当$\rightarrow F\in derivable(F,P)$;(2) 认证性:若 F 是形如 $\text{end}(M,M')$的闭原子公式,则 F 关于 $fixpoint(P)\cup B_b$ 逻辑可推导并且 $\text{begin}(M,M')\in B_b$.当且仅当存在 $H_1\wedge...\wedge H_n\rightarrow F\in derivable(F,P)$,其中 H_i 均为形如 $\text{begin}(M_i, M_i')$的原子公式.

定理 1 和定理 2 给出了 SPVT 中验证保密性和认证性的方法.

3 安全协议反例的自动构造

定义 **10**(反例规则). 设 P 是安全协议逻辑程序,则:(1) 若 $secret$ 是 P 中需要保密的名,$F=attacker(role(\langle M,N,$

$tag\rangle$,$secret$)),并且$\rightarrow F\in derivable(F,P)$,则$\rightarrow F$ 是安全协议关于保密性的反例规则;(2) 如果 begin(M,M')和 end(M,M')是一个对应性断言,F=end(M,M')是一个闭原子公式,$H\rightarrow F\in derivable(F,P)$,并且 begin($M$,$M'$)$\notin H$,则 $H\rightarrow F$ 是安全协议关于认证性的反例规则.

定义 **11**(反例规则关于解形式不动点的推导树). 设 P 是安全协议逻辑程序,$H\rightarrow F$ 是一个反例规则,由 $H\rightarrow F\in derivable(F,P)$,则存在消解规则序列 $R^0=F\rightarrow F,R^1,...,R^n$ 和逻辑规则序列 $R_1,...,R_n$,反例规则 $H\rightarrow F$ 关于解形式不动点的推导树是按照如下步骤构造的一棵树 T:

令 $R^0=F\rightarrow F$,由 $derivable(F,P)=derivablerec(F\rightarrow F,\varphi,P)$和 $H\rightarrow F\in derivable(F,P)$,则 $H\rightarrow F\in derivablerec(F\rightarrow F,\varphi,P)$,存在 $fixpoint(P)$中的逻辑规则序列 $R_1,...,R_n$,消解规则序列 $R^0=F\rightarrow F,R^1,...,R^n$,使得 $R_1,...,R_n$ 中任意两条逻辑规则中的变量集合不相交,并且 R^i 与 R_{i+1} 中的变量集合不相交(i=1,…,n−1),其中 $R^{i+1}=elimdup(R_{i+1}\cdot R^i),\theta_{i+1}=sub(R_{i+1},R^i)$,$R^n=H\rightarrow F$,并且对于任意的 $i<j$,$R^i\not\Rightarrow R^j$,取$\theta=\theta_1...\theta_n$.设 $R^i=F_1^i\wedge...\wedge F_{f(i)}^i\rightarrow F$,$f(i)$表示 R^i逻辑前件中原子公式的数目,若 $f(i)$=0,则 $R^i=\rightarrow F$,显然 $f(n)$=0,并且设 $F_j^i=attacker(role(\langle M_j^i,N_j^i,tag_j^i\rangle,U_j^i))$,$attacker(role(\langle M_{h(i)}^i,N_{h(i)}^i,tag_{h(i)}^i\rangle,U_{h(i)}^i))$是 R^i 的选择原子.设 $R_i=A_1^i\wedge...\wedge A_{g(i)}^i\rightarrow C_i$,$g(i)$表示 R_i 逻辑前件中原子公式的数目,若 $g(i)$=0,则 $R_i=\rightarrow C_i$,显然 $g(n)$=0,并且设 $A_j^i=attacker(role(\langle S_j^i,T_j^i,tag_j'^i\rangle,V_j^i))$,$C_i=attacker(role(\langle S^i,T^i,tag^i\rangle,V^i))$.由 $R^{i+1}=elimdup(R_{i+1}\cdot R^i)$,则$\{V_1^{(i+1)}\ \theta_{i+1},...,V_{g(i+1)}^{(i+1)}\ \theta_{i+1},U_1^i\ \theta_{i+1},...,U_{h(i)-1}^i\ \theta_{i+1},U_{h(i)+1}^i\ \theta_{i+1},...,U_{f(i)}^i\ \theta_{i+1}\}=\{U_1^{(i+1)},...,U_{f(i+1)}^{(i+1)}\}$,则$\{V_1^{(i+1)}\ \theta,...,V_{g(i+1)}^{(i+1)}\ \theta,U_1^i\ \theta,...,U_{h(i)-1}^i\ \theta,U_{h(i)+1}^i\ \theta,...,U_{f(i)}^i\ \theta\}=\{U_1^{(i+1)}\ \theta,...,U_{f(i+1)}^{(i+1)}\ \theta\}$.定义 $class(U_j^1)$为$\{A_k^1\ \theta|V_k^1\ \theta_1=U_j^1,k=1,...,g(1)\}$,对于 i>1,定义 $class(U_j^i)$为

$$\{A|U_k^{i-1}\ \theta_i=U_j^i,k=1,...,h(i)-1,h(i)+1,...,f(i),A\in class(U_k^{i-1})\}\cup\{A_k^i\ \theta|V_k^i\ \theta_i=U_j^i,k=1,...,g(i)\}.$$

(1) $R^0=F\rightarrow F$,首先生成一个结点作为根结点,根结点有唯一的一条出边并且用闭原子公式 F 标记,将根结点的出边标记为未处理;

(2) 生成一个新结点,用逻辑规则 R_1 作为它的标记,根结点的出边作为新结点的入边,并将根结点的出边标记为已处理.生成 $g(1)$条边作为新结点的出边,并且分别用原子公式 $A_1^1\ \theta,...,A_{g(1)}^1\ \theta$标记;由 $M\theta_1=M=V^1\theta_1$,$M'=M'\theta=V^1\theta$,则 $R_1\Rightarrow A_1^1\ \theta\wedge...\wedge A_{g(1)}^1\ \theta\rightarrow F$.将新结点的 $g(1)$条出边标记为未处理,标记为未处理的出边用 $class(U_1^1),...,class(U_{f(1)}^1)$中的原子公式标记.

(3) 假设由消解规则序列 $R^0=F\rightarrow F,R^1,...,R^i$ 和逻辑规则序列 $R_1,...,R_i$ 构造出的部分树中未处理的出边用 $class(U_1^i),...,class(U_{f(i)}^i)$中的原子公式标记,由 $R^{i+1}=elimdup(R_{i+1}\cdot R^i)$,$attacker(role(\langle M_{h(i)}^i,N_{h(i)}^i,tag_{h(i)}^i\rangle,U_{h(i)}^i))$是 R^i 的选择原子,为每一个用 $class(U_{h(i)}^i)$中的原子公式标记的出边生成一个结点,结点以该出边作为入边,并将这些出边标记为已处理,生成的结点用逻辑规则 R_{i+1} 作为标记,用标记为 $A_1^{(i+1)}\ \theta,...,A_{g(i+1)}^{(i+1)}\ \theta$的 $g(i+1)$条边作为出边,并将这些出边标记为未处理.设 $A_j^{i-k}\ \theta\in class(U_{h(i)}^i)$,则 $V_j^{i-k}\ \theta_{i-k}...\theta_i=U_{h(i)}^i$,$V_j^{i-k}\ \theta=U_{h(i)}^i\ \theta$,由 $U_{h(i)}^i\ \theta_{i+1}=V^{i+1}\theta_{i+1}$,则 $V_j^{i-k}\ \theta=V^{i+1}\theta$,从而 $R_{i+1}\Rightarrow A_1^{(i+1)}\ \theta\wedge...\wedge A_{g(i+1)}^{(i+1)}\ \theta\rightarrow A_j^{i-k}\ \theta$.由$\{V_1^{(i+1)}\ \theta_{i+1},...,V_{g(i+1)}^{(i+1)}\ \theta_{i+1},U_1^i\ \theta_{i+1},...,U_{h(i)-1}^i\ \theta_{i+1},U_{h(i)+1}^i\ \theta_{i+1},...,U_{f(i)}^i\ \theta_{i+1}\}=\{U_1^{(i+1)},...,U_{f(i+1)}^{(i+1)}\}$,则增加新结点后得到的部分树中,未处理的出边用 $class(U_1^{(i+1)}),...,class(U_{f(i+1)}^{(i+1)})$中的原子公式标记.

(4) 设由消解规则序列 $R^0=F\rightarrow F,R^1,...,R^n$ 和逻辑规则序列 $R_1,...,R_n$ 构造的部分树为 T,由 $f(n)$=0,则 T 中所有出边都被标记为已处理,T 构造完毕.

设 $R_1=elimdup(R_2\circ R_3)$,$R_i=H_1^i\wedge\ldots\wedge H_{f(i)}^i\rightarrow C_i$,$R_2\in UnSolvedForm$,$R_3\in SolvedForm$,其中 $f(i)$表示 R_i逻辑前件中原子公式的数目,$H_j^i=attacker(role(\langle M_j^i,N_j^i,tag_j^i\rangle,U_j^i))$或 H_j^i=begin(M_j^i,U_j^i),$\theta=sub(R_2,R_3)$,对每一个 U_j^1,分别定义 $representativeAtom(U_j^1)$和 $deletedAtom(U_j^1)$如下:

如果 R_3 逻辑前件中存在原子公式 H_k^3 使得 $U_k^3\ \theta=U_j^1$,则定义:

$representativeAtom(U_j^1)=H_s^3$,其中 H_s^3 是 R_3 逻辑前件中满足 $U_k^3\ \theta=U_j^1$ (k=1,...,f(3))的第 1 个原子公式;

否则,$representativeAtom(U_j^1)=H_s^2$,其中 H_s^2 是 R_2 逻辑前件中满足 $U_k^2\theta=U_j^1$ (k=1,...,f(2))的第 1 个原子公式;

定义 $deletedAtom(U_j^1)=\{H_k^2 \mid U_k^2\theta=U_j^1\}\cup\{H_k^3 \mid U_k^3\theta=U_j^1\}-representativeAtom(U_j^1)$.

定义 **12**(支撑子树). 设 T 是一棵逻辑弱推导树,原子公式 F'是 T 中一条出边 e 的标记,T 中以 e 作为入边的结点为 *node*,以 *node* 作为根结点并且包含从 e 开始的所有分枝的 T 的子树 T',称为原子公式 F'的支撑子树.

原子公式 F'的支撑子树 T'的直观含义是:增加一个根结点 *node*′,*node*′与支撑子树 T'的根结点用边 e'连接,并用 F'作为 e'的标记,则得到的一棵新的树实际上是原子公式 F'的一棵逻辑弱推导树.

定义 **13**(反例规则关于安全协议逻辑程序的推导树). 设 P 是安全协议逻辑程序,$H\to F$ 是一个反例规则,T 是 $H\to F$ 关于解形式不动点的推导树,则 $H\to F$ 由 T 得到的关于安全协议逻辑程序 P 的推导树是按照如下步骤构造的一棵树:

设 *node* 是 T 中的任意一个结点,*node* 用逻辑规则 $R=H_1\wedge...\wedge H_n\to C\in fixpoint(P)$标记,$H_i=attacker(role(\langle M_i, N_i, tag_i\rangle, M_i'))$或 $\text{begin}(M_i, M_i')$,$C=attacker(role(\langle M,N,tag\rangle,M'))$,*node* 的入边用原子公式 $F'=attacker(role(\langle S',T',tag'\rangle, M''))$标记,*node* 所有的出边依次用原子公式 $F_1,...,F_k$ 标记,$F_i=attacker(role(\langle S_i,T_i,tag_i'\rangle, M_i''))$($i$=1,...,$k$),则存在置换 σ,使得 σ是 $R\Rightarrow F_1\wedge...\wedge F_k\to F'$的蕴涵置换,对于 R 逻辑前件中每一个原子公式 $H_i=attacker(role(\langle M_i,N_i,tag_i\rangle, M_i'))$,存在原子公式 $F_j=attacker(role(\langle S_j,T_j,tag_j'\rangle, M_j''))$,使得 $M_j''=M_i'\sigma$.若 R 是 P 中的逻辑规则,则不对结点 *node* 作变换;若 R 不是 P 中的逻辑规则,则存在逻辑规则 R_1 和 R_2,使得 $R=elimdup(R_2\circ R_1)$,$R_1\in SolvedForm$,$R_2\in UnSolvedForm$,设 $R_1=H_1^1\wedge...\wedge H_l^1\to C_1$, $H_i^1=attacker(role(\langle M_i^1, N_i^1, tag_i^1\rangle, U_i^1))$($i$=1,...,$l$),$C_1=attacker(role(\langle M^1, N^1, tag^1\rangle, T^1))$,$R_2=H_1^2\wedge...\wedge H_i^2\wedge...\wedge H_m^2\to C_2$, $H_i^2=attacker(role(\langle M_i^2, N_i^2, tag_i^2\rangle, V_i^2))$, i=1,…,m,$C_2=attacker(role(\langle M^2,N^2,tag^2\rangle,T^2))$, $H_i^2=selectedGoal(R_2)$,$\theta=sub(R_1,R_2)$,由 $R=elimdup(R_1\circ R_2)$,将结点 *node* 用两个结点 $node_1$ 和 $node_2$ 替换,并且 $node_1$ 和 $node_2$ 用标记为 $C_1\theta\sigma$的一条边连接,$node_1$ 用 R_1 作为标记,$node_2$ 用 R_2 作为标记.结点 $node_1$ 有 l 条出边,分别对应于 $H_1^1,..., H_l^1$,$node_2$ 有 m 条出边,分别对应于 $H_1^2,..., H_m^2$,如果 $H_j^t\in representiveAtom(M_i')$并且 $M_i'\sigma=M_s''$,则用 F_s 标记的边是 H_j^t 对应的边;如果 $H_j^t\in deletedAtom(M_i')$并且 $M_i'\sigma=M_s''$,则生成 F_s 的支撑子树的一个拷贝,支撑子树的根结点和结点 $node_t$ 用标记为 F_s 的边连接.$R_1\Rightarrow H_1^1\theta\sigma\wedge...\wedge H_l^1\theta\sigma\to C_1\theta\sigma$,$R_2\Rightarrow H_1^2\theta\sigma\wedge...\wedge H_{i-1}^2\theta\sigma\wedge H_i^2\theta\sigma\wedge H_{i+1}^2\theta\sigma\wedge...\wedge H_m^2\theta\sigma\to F'$,则变换后的逻辑推导树是一棵逻辑弱推导树.对变换后的逻辑推导树中的所有结点作上述变换,直到得到的逻辑弱推导树中所有结点都用 P 中的逻辑规则标记,因为 F 关于 $fixpoint(P)$的逻辑推导树是一棵有限树,上述变换过程一定终止,得到一棵 $H\to F$ 关于 P 的逻辑弱-推导树.

定义 **14**(反例树). 设 P 是安全协议逻辑程序,$H\to F$ 是反例规则,T 是 $H\to F$ 关于 P 的推导树,由 T 可以构造 $H\to F$ 关于 P 的反例树 T':T 和 T'具有相同的结点和边,只是边的标记不同.若 e 是 T 中用 F'标记的一条边,T 中以 e 作为入边的结点 *node* 的标记为 R,*node* 的所有出边为 $F_1,\ldots,F_n$,则 e 在 T'中用序偶$\langle F',G\rangle$标记,其中 $G=C\sigma$,C 是 R 的逻辑后件,σ是 $R\Rightarrow F_1\wedge...\wedge F_n\to F'$的蕴涵置换.

定义 11、定义 13、定义 14 定义了反例规则关于解形式不动点、安全协议逻辑程序的推导树以及反例树,同时也给出了它们的构造算法.

设 T 是反例规则 $H\to F$ 关于安全协议逻辑程序的一棵反例树,SPVT 中由 T 构造安全协议反例的算法简单描述如下:按照后序遍历的次序访问 T 中每一个结点 *node*,若 *node* 的入边用序偶$\langle F,G\rangle$标记,$F=attacker(role(\langle M_1,N_1,tag_1\rangle, M_1'))$,$G=attacker(role(\langle M_2,N_2,tag_2\rangle, M_2'))$,若 $tag_1\vee tag_2\neq$false,则输出 $M_2(M_1)\to N_1(N_2)$: M_1'.

4 实 例

使用 SPVT 中安全协议的描述语言,简化的 Needham-Schroeder 公钥认证协议可以描述为如下的多角色进程的并发合成,其中 *processA*, *processB* 分别对应角色 *A* 和角色 *B*:

$processA\triangleq \text{begin}(RoleB,host(v_1)).c(role(\langle host(v_1),host(k_A),\text{false}\rangle,host(v_1))).(\nu N_A)\,\overline{c}\,\langle role(\langle host(k_A),host(v_1),\text{true}\rangle),encrypt(2tuple(N_A,host(k_A),pub(v_1))))\rangle.c(role(\langle host(v_1),host(k_A),\text{true}\rangle.x\_M)$.let $x\_N=decrypt(x\_M,k_A)$ in let $x\_N=2Tuple(N_A,v_2)$ in $\overline{c}\,\langle role(\langle host(k_A),host(v_1),\text{true}\rangle,encrypt(v_2,pub(v_1))))\rangle$.

$processB \triangleq \overline{c}\langle role(\langle host(k_B),host(v_3),\text{false}\rangle,host(k_B))\rangle.c(role(\langle host(v_3),host(k_B),\text{true}\rangle,y\_M)).\text{let } y\_N=decrypt(y\_M,k_B) \text{ in let } y\_N=2Tuple(v_4,host(v_3)) \text{ in } (vN_B)\,\overline{c}\langle role(\langle host(k_B),host(v_3),\text{true}\rangle,encrypt(2Tuple(v_4,N_B),pub(v_3)))\rangle.c(role(\langle host(v_3),host(k_B),\text{true}\rangle,y\_M_1)).\text{let } y\_N_1=decrypt(y\_M_1,k_B) \text{ in if } y\_N_1=N_B \text{ then if } v_3=k_A \text{ then end}(RoleB,host(k_B)).$

$NS \triangleq (vk_A)(vk_B)(\overline{c}\langle role(\langle -,-,false\rangle),pub(k_A))\rangle)|(\overline{c}\langle role(\langle -,-,\text{false}\rangle),pub(k_B))\rangle)|(\overline{c}\langle role(\langle -,-,\text{false}\rangle),host(k_A))\rangle)|(\overline{c}\langle role(\langle -,-,\text{false}\rangle),host(k_B))\rangle)|(!processA)(!processB).$

下面是简化的 Needham-Schroeder 公钥认证协议对应的安全协议逻辑程序:

$\rightarrow attacker(role(\langle host(k_I), -,\text{false}\rangle,pub(k_I[])));$ $\rightarrow attacker(role(\langle host(k_I),host(k_I),\text{false}\rangle,k_I[]));$

$\rightarrow attacker(role(\langle host(k_I), -,\text{false}\rangle,host[k_I]));$ $\rightarrow attacker(role(\langle host(k_I), -,\text{false}\rangle,pub(k_A[])));$

$\rightarrow attacker(role(\langle host(k_I), -,\text{false}\rangle,host(k_A[])));$ $\rightarrow attacker(role(\langle host(k_I), -,\text{false}\rangle,pub(k_B[])));$

$\rightarrow attacker(role(\langle host(k_I), -,\text{false}\rangle,host(k_B[])));$

$attacker(role(\langle host(k_I), -,\text{false}\rangle,x_1))\wedge\ldots\wedge attacker(role(\langle host(k_I), -,\text{false}\rangle,x_n))\rightarrow attacker(role(\langle host(k_I), -,\text{false}\rangle,nTuple(x_1,\ldots,x_n)));$

$attacker(role(\langle -,host(k_I),\text{false}\rangle,ntuple(x_1,\ldots,x_n)))\rightarrow attacker(role(\langle host(k_I), -,\text{false}\rangle,x_i));i=1,\ldots,n;$

$\text{begin}(RoleB,host(v_1))\wedge attacker(role(\langle host(v_1),host(k_A[]),\text{false}\rangle,host(v_1)))\rightarrow attacker(role(\langle host(k_A[]),host(v_1),\text{true}\rangle,encrypt(2tuple(N_A[host(v_1),i^1],host(k_A[])),pub(v_1));$

$attacker(role(\langle host(v_3),host(k_B[]),\text{true}\rangle,encrypt(2tuple(v_2,host(v_3)),pub(k_B[]))))\rightarrow attacker(role(\langle host(k_B[]),host(v_3),\text{true}\rangle,encrypt(2tuple(v_2,N_B[encrypt(2tuple(v_2,host(v_3)),pub(k_B[]))],j^1)),pub(v_3));$

$\text{begin}(RoleB,host(v_4))\wedge attacker(role(\langle host(v_4),host(k_A[]),\text{false}\rangle,host(v_4)))\wedge attacker(role(\langle host(v_4),host(k_A[]),\text{true}\rangle,encrypt(2tuple(N_A[host(v_4),i^2],v_5),pub(k_A[])))\rightarrow attacker(role(\langle host(k_A[]),host(v_4),true\rangle,encrypt(v_5,pub(v_4)));$

$attacker(role(\langle host(k_A[]),host(k_B[]),\text{true}\rangle,encrypt(2tuple(v_6,host(k_A[])),pub(k_B[])))\wedge attacker(role(\langle host(k_A[]),host(k_B[]),\text{true}\rangle,encrypt(N_B[encrypt(2tuple(v_6,host(k_A[])),pub(k_B[])),j^2]),pub(k_B[]))\rightarrow \text{end}(RoleB,host(k_B[]));$

简化的 Needham-Schroeder 公钥认证协议的安全协议逻辑程序在 SPVT 中经过 10 次迭代计算终止,得到 49 条逻辑规则,其中第 48 条逻辑规则 $\text{begin}(RoleB,host(k_I[]))\rightarrow\text{end}(RoleB,host(k_B[]))$是反例规则.SPVT 自动生成的反例描述如下:

$host(k_A[])(host(k_A[]))\rightarrow host(k_I[])(host(k_I[])):\{N_A[host(k_I[]),z778],host(k_A[])\}_{pub(kI[])}$

$host(k_I[])(host(k_A[]))\rightarrow host(k_B[])(host(k_B[])):\{N_A[host(k_I[]),z778],host(k_A[])\}_{pub(kB[])}$

$host(k_A[])(host(k_A[]))\rightarrow host(k_I[])(host(k_I[])):\{N_A[host(k_I[]),z778],host(k_A[])\}_{pub(kI[])}$

$host(k_I[])(host(k_A[]))\rightarrow host(k_B[])(host(k_B[])):\{N_A[host(k_I[]),z778],host(k_A[])\}_{pub(kB[])}$

$host(k_B[])(host(k_I[]))\rightarrow host(k_A[])(host(k_A[])):\{N_A[host(k_I[]),z778],N_B[\{N_A[host(k_I[]),z778],host(k_A[])\}_{pub(kB[])}\},z777\}\}_{pub(kA[])}$

$host(k_A[])(host(k_A[]))\rightarrow host(k_I[])(host(k_I[])):\{N_B[\{N_A[host(k_I[]),z778],host(k_A[])\}\}_{pub(kB[])},z777]\}_{pub(kI[])}$

$host(k_I[])(host(k_A[]))\rightarrow host(k_B[])(host(k_B[])):\{N_B[\{N_A[host(k_I[]),z778],host(k_A[])\}]_{pub(kB[])},z777\}\}_{pub(kB[])}$

由于建模方面的原因,语句 $host(k_A[])(host(k_A[]))\rightarrow host(k_I[])(host(k_I[])):\{N_A[host(k_I[]),z778],host(k_A[])\}_{pub(kI[])}$ $host(k_I[])(host(k_A[]))\rightarrow host(k_B[])(host(k_B[])):\{N_A[host(k_I[]),z778],host(k_A[])\}_{pub(kB[])}$被输出两次,可以平凡地将安全协议反例中重复的语句删除.

5 结 语

本文描述了安全协议验证工具 SPVT 中的安全协议描述语言、攻击者模型、安全性质的验证方法、安全协议反例的自动构造方法.通过简化的 Needham-Schroeder 公钥认证协议的自动验证,表明了 SPVT 用于安全协议验证的有效性.在本文和已有研究工作的基础上[13,14],使用 SPVT 对复杂安全协议(如 Kerberos5)进行形式化分析与验证,是我们下一步的研究工作.同时,结合安全协议中原语(如异或算子)具有的代数性质(如异或算子具有可结合性、可交换性、有零元、满足幂零律),对安全协议进行形式化分析与验证是下一步的研究重点.

References:

[1] Abadi M, Blanchet B. Analyzing security protocols with secrecy types and logic programs. In: Jones ND, Leroy X, eds. Proc. of the 29th ACM SIGPLAN-SIGACT Symp. on Principles of Programing Languages (POPL 2002). Portland: ACM Press, 2002. 33−44.

[2] Blanchet B. An efficient cryptographic protocol verifier based on prolog rules. In: Pandya P, Radhakrishnan J, eds. Proc .of the 14th Computer Security Foundation Workshop (CSFW14). Cape Breton: IEEE Computer Society Press, 2001. 82−96.

[3] Blanchet B. From secrecy to authenticity in security protocols. In: Cousot P, ed. Proc. of the 9th Int'l Static Analysis Symp. (SAS 2002). LNCS 2477, Madrid: Springer-Verlag, 2002. 242−259.

[4] Clark J, Jacob J. A survey of authentication protocol literature, Web draft version 1.0. 1997. http://www.cs.york.uk/~jac

[5] Lowe G. Breaking and fixing the needham-schroeder public-key protocol using FDR. In: Steffen M, ed. Proc. of the Tools and Algorithms for the Construction and Analysis of Systems (TACAS). LNCS 1055, Passau: Springer-Verlag, 1996. 147−166.

[6] Abadi M, Gordon AD. A calculus for cryptographic protocols: The spi calculus. In: Matsumoto T, ed. Proc. of the 4th ACM Conf. on Computer and Communications Security. Zurich: ACM Press, 1997. 36−47.

[7] Heather J, Lowe G, Schneider S. How to prevent type attacks on security protocols. In: Syverson P, ed. Proc. of the 13th Computer Security Foundation Workshop. IEEE Computer Society Press, 2000. 255−268.

[8] Boreale M. Symbolic trace analysis of cryptographic protocols. In: Orejas F, Spirakis PG, Van Leeuwen J, ed. Proc. of the 28th Int'l Conf. on Automata, Language and Programming (ICALP 2001). LNCS 2076, Geneva: Springer-Verlag, 2001. 667−681.

[9] Mitchell JC, Mitchell M, Stern U. Automated analysis of cryptographic protocols using mur φ. In: Proc. of the 1997 IEEE Symp. on Security and Privacy. Okland: IEEE Computer Society Press, 1997. 141−153.

[10] Song DX. Athena: A new efficient automatic checker for security protocol analysis. In: Gorrieri R, ed. Proc. of the 12th IEEE Computer Security Foundation Workshop (CSFW12). Mordano: IEEE Computer Society Press, 1999. 192−202.

[11] Dolev D, Yao AC. On the security of public-key protocols. IEEE Trans. on Information Theory, 1983,29(2):198−208.

[12] Zhang YQ, Wang L, Xiao GZ, Wu JP. Model checking analysis of Needham-Schroeder public-key protocols. Journal of Software, 2000,11(10):1348−1352 (in Chinese with English abstract).

[13] Liu DX, Bai YC. Study on the proof method of authentication protocols based on Strand space. Journal of Software, 2002,13(7):1313−1317 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/13/1313.htm

[14] Li MJ, Li ZJ, Chen HW. A survey of security protocol verification based on process algebra. Journal of Computer Research and Development, 2004,41(7):1093−1103 (in Chinese with English abstract).

[15] Li MJ, Li ZJ, Chen HW. Verifying security protocol based on logic programming. Chinese Journal of Computers, 2004,27(10): 1361−1368 (in Chinese with English abstract).

附中文参考文献:

[12] 张玉清,王磊,肖国镇,吴建平.Needham-Schroeder 公钥协议的模型检测分析.软件学报,2000,11(10):1348−1352.

[13] 刘东喜,白英彩.基于 Strand 空间的认证协议证明方法研究.软件学报,2002,13(7):1313−1317. http://www.jos.org.cn/1000-9825/13/1313.htm

[14] 李梦君,李舟军,陈火旺.基于进程代数安全协议验证的研究综述.计算机研究与发展,2004,41(7):1093−1103.

[15] 李梦君,李舟军,陈火旺.基于逻辑程序的安全协议验证.计算机学报,2004,27(10):1361−1368.

李梦君(1975－),男,湖北云梦人,博士,讲师,主要研究领域为安全协议的形式化分析,形式化技术.

陈火旺(1936－),男,教授,博士生导师,中国工程院院士,主要研究领域为软件工程,软件理论.

李舟军(1963－),男,博士,教授,CCF 高级会员,主要研究领域为安全协议的形式化分析,进程代数理论,数据挖掘.