基于类 pi 演算的电子支付协议安全性形式化研究*

顾永跟[1,2]，李国强[1]，王国钧[2]

(1. 上海交通大学 计算机系，上海 200030；2. 湖州师范学院 计算机系，浙江 湖州 313000)

摘 要：设计一个满足安全需求的协议非常困难，并且极易出错，因此利用形式化方法来检验安全协议引起了人们极大的关注。使用了类 pi 演算来验证电子支付协议的认证性和匿名性。

关键词：类 pi 演算；电子支付协议；认证性；匿名性

中图法分类号：TP301；TP309 **文献标识码**：A **文章编号**：1001-3695(2006)03-0022-03

Formal Research of Electronic Payment Protocol Based on pi-like Calculus

GU Yong-gen[1,2], LI Guo-qiang[1], WANG Guo-jun[2]

(1. *Dept. of Computer Science, Shanghai Jiaotong University, Shanghai* 200030, *China*; 2. *Dept. of Computer Science, Huzhou Normal University, Huzhou Zhejiang* 313000, *China*)

Abstract: Design of a protocol is a difficult and error-prone task, thus the use of formal methods that allow for the verification of such protocols has received increasing attention. In this paper, using pi-like calculus to verify authentication and anonymity properties of electronic payment protocols.

Key words: pi-like Calculus; Electronic Payment Protocol; Authentication; Anonymity

电子商务是以计算机互联网络和信息技术为平台的经济交易活动，目前已经渗透到社会生活的诸多方面，成为当今经济增长的重要力量。作为电子商务的关键，电子支付协议对于安全的性质和功能有着特殊且更高的要求，除了保密和认证的安全要求以外，性能不同的电子支付协议还有其他的安全要求，如不可否认性、公平性、原子性、匿名性等。目前，越来越多的电子支付协议正在不断地涌现，如何保证它们的实现达到既定安全要求成为计算机科学家主要研究的问题。当前的研究主要集中在对电子支付协议的形式化描述和验证方面，一般采用逻辑方法、数学证明方法和进程代数方法等。

pi 演算[1]是 Robin Milner 在 CCS 等移动并发计算模型的基础上提出的一种移动计算模型。它的通道具有确定的作用域，作用域以外的进程不能对该通道进行存取，这在一定程度上确保了通道通信的安全性。为了适合于安全协议的形式化分析，人们扩展了 pi 演算(如安全 pi 演算)、应用 pi 演算等，这里统称为类 pi 演算。应用 pi 演算[2]增加了传值、函数原语和项的代数相等关系，并且提出了观察等价和静态等价两种等价关系，使之更好地描述和验证具体的安全协议[3~5]。本文利用应用 pi 演算来验证典型电子支付协议——IBM 公司提出的 *i*KP 协议族的认证性及其匿名性。

1 典型的电子支付协议——*i*KP 协议

*i*KP 协议族[6,7]由 IBM 公司在 1995 年提出，是一组基于公钥密码体制，应用于 Internet 上信用卡的交易。按照需要数字签名主体数量的不同分为 1KP, 2KP, 3KP 三个协议，分别满足不同的安全要求。*i*KP 协议族一经发明便在欧洲投入了广泛的实际应用中，著名的 SET 协议就是根据其中的 3KP 改进而来。*i*KP 协议族参与协议的主体由购买方 Buyer、销售方 Seller 和转账方 Acquirer 组成。在这里我们主要来研究讨论 1KP 协议的安全性。为了描述 1KP 协议，我们首先定义加密原语：用 SK_X 来表示 X 的秘密密钥，$PK(SK_X)$ 表示其对应的公开密钥。同时，我们用 $H(.)$ 来表示单向散列函数；$E_X(.)$ 表示用公开密钥 PK_X 进行加密；$S_X(.)$ 表示用秘密密钥 SK_X 进行数字签名。

我们定义了一些用来简化协议描述的符号和它们所代表的含义如下：

符号	代表内容
Desc	购买方的信息，如地址、电话、信用卡账户名等
$Salt_B$	购买方产生的随机数，用来传递给销售方加密 Desc
Authprice	购买方所购买商品的数量和价格
Date	销售方产生的时间戳，用来防止重复攻击
$Nonce_S$	销售方产生的随机数，用来防止重复攻击
ID_S	销售方的标志码，用来区别不同的销售方
TID_S	销售方产生的交易标志码，用来区别不同的交易
Ban	购买方的信用卡账号
Expiration	购买方信用卡的期限
R_B	购买方用来产生自己标志码的随机数
ID_B	购买方的标志码，产生方法：$ID_B = H_k(R_B, Ban)$
Despcode	转账方的应答，表示完成或未完成转账

同时，由于 1KP 协议的消息内容十分烦琐，为了简明表示，我们对其消息内容进行了如下的简写：

消息	代表内容
Common	$Authprice, ID_S, TID_S, Date, Nonce_S, ID_B, H(Salt_B, Desc)$
Clear	$ID_S, TID_S, Date, Nonce_S, H(Common)$
Slip	$Authprice, H(Common), Ban, R_B, Expiration$
EncSlip	$E_A(Slip)$
Sig_A	$S_A(Respcode, H(Common))$

1KP 协议是由六步组成：①购买方向销售方发起请求 Initiate；②销售方响应销售方请求 Invoice；③购买方对销售方提起支付 Payment；④销售方向转账方申请转账 Auth-Request；

收稿日期：2005-03-16；修返日期：2005-05-16

基金项目：国家自然科学基金资助项目(60473006)

⑤转账方向销售方响应转账 Auth-Response; ⑥销售方向购买方确认转账结果 Confirm。协议流如下:

Initiate: B→S: $Salt_B$, ID_B
Invoice: S→B: Clear
Payment: B→S: EncSlip
Auth-Request: S→A: Clear, $H(Salt_B, Desc)$, EncSlip
Auth-Response: A→S: Respcode, Sig_A
Confirm: S→B: Respcode, Sig_A

(1) 购买方向销售方发起购买请求: 发送消息 $Salt_B$, ID_B。购买方产生随机数 R_B, 使用公式 $ID_B = H_k(R_B, Ban)$ 计算出 ID_B, 连同产生的另一个随机数 $Salt_B$ 一起送往销售方, 发起购买请求。

(2) 销售方响应销售方请求: 发送消息 Clear。销售方记录当前时间 Date, 并产生随机数 $Nonce_S$, 这两个元素一起来标志信息的唯一性。随之产生 TID_S, 计算 $H_k(Salt_B, Desc)$; 加上销售方的标志码 ID_S 和用户购买商品的数量和价格信息 Authprice 建立起 Common 消息, 并且用单向散列函数将之压缩。随后, 将 ID_S, TID_S, Date, $Nonce_S$ 和 H(Common) 一起发送给购买方。

(3) 购买方对销售方提起支付: 发送消息 EncSlip。购买方得到销售方的反馈后, 通过得到的 ID_S, TID_S, Date, $Nonce_S$ 和自己原有的信息得出 H(Common), 与销售方传来的 H(Common) 作比较, 若相等则可以提起支付。购买方将 Authprice、H(Common)、用户的信用卡账号 Ban、用来产生用户标志码的随机数 R_B、信用卡使用期限 Expiration 这些信息用转账方的公开密钥加密后发送给销售方。

(4) 销售方向转账方申请转账: 发送消息 Clear, $H_k(Salt_B, Desc)$, EncSlip。销售方将消息 Clear 和 $H_k(Salt_B, DESC)$, 连同购买方传来的消息一起发送给转账方, 申请转账。

(5) 转账方向销售方响应转账: 发送消息 Despcode, Sig_A。转账方首先通过接收到的 Clear 得到 ID_S, TID_S, Date, $Nonce_S$; 接着得到采用自己的秘密密钥解开 EncSlip, 得到 Authprice, Ban, Expiration, R_B; 然后用这些信息重构 H(Common), 并且与得到的 H(Common) 进行比较。若相等, 则将交易的结果和自己的数字签名反馈给销售方。

(6) 销售方向购买方确认转账结果: 发送消息 Despcode, Sig_A。销售方将收到的信息发送给购买方, 告知交易是否成功。

2　应用 pi 演算

应用 pi 演算的语法和 pi 演算大致相同, 只是在 pi 演算当中, 通道传递的只是名 name, 而在应用 pi 演算中, 通道传递的是项 term, 项包括了名、变量和函数原语三种。

2.1　应用 pi 演算的语法

我们首先定义 ∑ 为一函数集合, 其中包含有限个函数项, 则给定一个 ∑, 应用 pi 演算的项定义如下:

$L, M, N ::= n \mid x \mid f(M_1, M_2, \cdots, M_l)$

其中, n 表示名, x 表示变量, $f(M_1, M_2, \cdots, M_l)$ 表示函数项。

接着定义应用 pi 演算的普通进程(Plain Processes) 如下:

$P, Q, R ::= 0 \mid P \mid Q \mid !P \mid (\nu n).P \mid \text{if } U = V \text{ then } P \text{ else } Q \mid u(x).P \mid \underline{u}\langle M\rangle.P$

在这里, 0 表示空进程, P|Q 表示并行复合, !P 表示复制, $\nu n.P$ 表示受限, if U = V then P else Q 表示条件, $\underline{\ }x).P$ 表示输入, $\underline{\ }\langle M\rangle.P$ 表示输出。

普通进程加入了主动替换(Active Substitution) 原语, 可以被扩展成为扩展进程(Extended Processes), 定义如下:

$A, B, C ::= P \mid A \mid B \mid (\nu n).A \mid (\nu x).A \mid \{x = V\}$

其中, P 表示普通进程, A|B 表示并行复合, $(\nu n).A$ 表示名受限, $(\nu x).A$ 表示变量受限, $\{x = V\}$ 表示主动替换。

2.2　等价关系

如果 A 通过通道 a 送出一个消息, 也即如果存在某个上下文(Context) (这里的上下文与传统意义上的不同, 它只表示在并行复合、受限和主动替换下的上下文) $C[\ ]$, 使得 $A \rightarrow^* C[\underline{a}\langle M\rangle.P]$, 我们可定义为 $A \mathrm{T}\ a$。

定义 1(观察等价)　观察等价($\approx$) 是闭扩展进程上关系 R 的最大对称关系。R 的定义如下, 如果 $A R B$ 则有:

(1) 如果 $A \mathrm{T}\ a$, 则 $B \mathrm{T}\ a$;

(2) 如果 $A \rightarrow^* A'$, 则对于某些 B' 有 $B \rightarrow^* B'$, 并且 $A' \mathrm{R} B'$;

(3) 对于所有的闭上下文, 有 $C[A] \mathrm{R} C[B]$。

静态等价($\approx_s$) 表示两个进程的任何框架(Frame) 都不能被任何其他进程所区分。我们首先定义框架相等来形式化地描述什么叫做在框架下不可区分。

定义 2(框架相等)　$(M = N)_\phi$, 表示两个项 M 和 N 在框架 ϕ 下相等, 当且仅当对于某些名 n 和替换 σ, 有 $\phi \equiv \nu n.\sigma$, $M\sigma = N\sigma$ 和 $\{n\} \cap (f_n(M) \cup f_n(N)) = \Phi$。

定义 3(静态等价)　两个框架 ϕ 和 ψ 静态等价 $\phi \approx_s \psi$, 即 $dom(\phi) = dom(\psi)$, 并且对于所有的项 M 和 N, $(M = N)_\phi$, 当且仅当 $(M = N)_\psi$。

两个闭扩展进程静态等价 $A \approx_s B$, 就是指它们的任何框架都是静态等价的。

3　1KP 协议的形式化分析

3.1　1KP 协议形式化表示

在 1KP 协议的形式化描述中, 我们需要定义它的项(包括变量、名和函数原语) 如下:

$T, U, V, V_0, \cdots ::=$	项
$A, B, S, x_1, x_2, \cdots$	变量
$C_{BS}, C_{SB}, C_{SA}, C_{AS}, \cdots$	通道名
$Nonce_A, K_A$, Date, $\cdots$	随机数、密钥、时戳名
$f(T_1, \cdots, T_n)$	函数

其中, 函数集 ∑ 包括了三类型的函数:

(1) 有关密码和检测的函数, 其中 $H(u_1, \cdots)$ 表示单项散列函数; Pk(u) 表示公开密钥; $\{T\}_V$ 表示用公钥加密或者用私钥数字签名; Decrypt(w, u) 表示用私钥解密; Checksig(w, u) 表示用公开密钥检测数字签名; Checktime(Time, Expiration) 表示检测时间戳是否过期。

(2) 函数集包括构造消息的函数, 包括 Initiate(u_1, u_2), Clear(u_1, u_2, u_3, u_4, u_5), Common(u_1, u_2, $\cdots$, u_7), Slip(u_1, u_2, u_3, u_4, u_5), ARequest(u_1, u_2, u_3), AResponse(u_1, u_2) 和 Confirm(u_1, u_2)。

(3) 函数集还包括投影函数, 即选择构造消息函数中第 j 个消息, 如 Initiate.j(Initiate(u_1, u_2)) 表示选择 Initiate( ) 函数中第 j 个消息。这类函数还包括 Clear.j(Clear(u_1, u_2, u_3, u_4, u_5)), Common.j(Common(u_1, u_2, $\cdots$, u_7)), Slip.j(Slip(u_1,

u_2, u_3, u_4, u_5)), ARequest. j(ARequest(u_1, u_2, u_3)), AResponse. j(AResponse(u_1, u_2)) 和 Confirm. j(Confirm Confirm(u_1, u_2))。

由上述函数,我们有如下项的代数相等关系:

Decrypt($\{x\}_{Pk(z)}$, z) = x

Checksig($\{x\}_z$, Pk(z)) = true

Checktime(time, Expiration) = true if time∈Expiration

false otherwise

Tuple. j(Tuple(x_1, ⋯, x_1)) = x_j for $j < i$, Tuple({Initiate, Clear, Common, Slip, ARequest, AReponse, Confirm}

有了项的定义和其代数项等关系,我们就可以形式化地描述 1KP 协议了。形式化描述分两步:消息定义和主体定义。

(1) 消息定义。我们通过主动替换来定义协议的消息。

σ_1: = {x_1 = Initiate($Salt_B$, ID_B)}

σ_2: = {x_2 = Clear(ID_S, TID_S, Date, $Nonce_S$, H(Authprice, ID_S, TID_S, Date, $Nonce_S$, Initiate. 2(x_1), H(Initiate. 1(x_1), Desc)))}

σ_3: = {x_3 = {Slip}$_{Pk(SK_A)}$}

σ_4: = {x_4 = ARequest(x_2, H(Initiate. 1(x_1), Desc), x_3)}

σ_5: = {x_5 = AResponse(Respcode, {Respcode, H(Common)}$_{SK_A}$)}

σ_6: = {x_6 = Confirm(AResponse. 1(x_5), AResponse. 2(x_5))}

(2) 主体定义。下面的代码是来表示协议主体,它们分别形式化地定义了购买方 P_B、销售方 P_S 和转账方 P_A。

P: = (γAuthprice)(γDesc)(P_B | P_S | P_A)

P_B: = (γ$Salt_B$)(γR_B)($\underline{C_{BS}}$ < x_1 σ_1 > |! (C_{SB}(x_2). if Clear. 5(x_2) = H(Authprice, Invoice. 1(x_2), Clear. 2(x_2), Clear. 3(x_2), H(R_B, Ban_B), H($Salt_B$, Desc)) then $\underline{C_{BS}}$ < x_3 σ_3 > |! (C_{SB}(x_6). checksig(confirm. 2(x_6), Pk(SK_A)) then F_B(x_6)))))

P_S: = (γID_S)(γTID_S)(γDate)(γ$Nonce_S$)(C_{BS}(x_1). ($\underline{C_{SB}}$ < x_2 σ_2 > |C_{BS}(x_3). (C_{SA} < x_4 σ_4 > |C_{AS}(x_5). C_{SB} < x_6 σ_6 >. F_S(x_5))))

P_A: = (γSK_A)! (C_{SA}(x_4). if Slip. 2(Decrypt(ARequest. 3(x_4), SK_A)) = H(Slip. 1(Decrypt(ARequest. 3(x_4), SK_A)), Clear. 1(AResponse. 1(x_4)), Clear. 2(ARequest. 1(x_4)), Clear. 3(ARequest. 1(x_4)), Clear. 4(ARequest. 1(x_4)), H(Slip. 3(Decrypt(ARequest. 3(x_4), SK_A)), Slip. 4(Decrypt(ARequest. 3(x_4), SK_A))), ARequest. 1(x_4)) ∧checktime(datetime, Slip(Decrypt(ARequest. 3(x_4)))))

Then ((γARespcode)($\underline{C_{AS}}$ < x_5 σ_5 >. F_A(x_4))))

在这些进程中,函数 F_B(x), F_S(x), F_A(x) 各自表示购买方、销售方和转账方在协议结束后所做的事情。

3.2 认证性的验证

在这一节中,我们首先考虑 1KP 协议的认证性。当一个协议开始运行的时候,如果转账方得到允许,开始将购买方账户上的钱转向销售方,那么这个允许一定是由购买方授权的。这就是我们所要考虑协议的认证性。因此,我们有如下定理:

定理 1(转账方的认证性) 假设 P→$^{\eta}$P′,如果在 η中存在

$\underline{C_{AS}}$ < AResponse(Respcode, {Respcode, H(Common)}$_{PK_A}$) >

则在此项之前,η中必定存在

$\underline{C_{BS}}$ < {Slip}$_{Pk(SK_A)}$ > and $\underline{C_{SA}}$ < ARequest(Clear, H($Salt_B$, Desc), {Slip}$_{Pk(SK_A)}$) >

证明:我们可以用对应性断言(Correspondence Assertion)[8] 来解释这个定理,即一旦转账方开始转账,则必定是由购买方授权这次转账。

如果在 η中存在 $\underline{C_{AS}}$ < AResponse(Respcode, {Respcode, H(Common)}$_{PK_A}$) >,则说明转账方成功地验证了购买方和销售方的身份。它通过 Ban_B 来验证购买方的身份,通过匹配 H(Common) 来验证销售方的身份,因此我们必定有 $\underline{C_{BS}}$ < {Slip}$_{Pk(SK_A)}$ > 和 $\underline{C_{SA}}$ < ARequest(Clear, H($Salt_B$, Desc), {Slip}$_{Pk(SK_A)}$) >。

同样,对于销售方和购买方的认证性,我们可以类似地构造并且证明定理,此处不再赘述。

3.3 1KP 协议匿名性的讨论

在网络交易中,为了使购买方安全购物转账,需要对销售方和网络上其他偷听者保持匿名性。然而 iKP 协议族并不完全提供购买方的匿名性,只是对于其账户提供匿名服务。在下面的定理中,我们可以证明购买方对于销售方和网络偷听者可以保持其账户的匿名性。

定理 2(购买方的匿名性) 对于购买方和网络偷听者,有 $P_B(Ban_B) \approx_s P_B(Ban'_B)$,则说明购买方在 Ban_B 具有匿名性。

证明:我们首先定义下面的框架。

定义 $\Phi_1(P_B(Ban_B))$ 为

{x_1 = ($Salt_B$, H(R_B, Ban_B))(x_3 = {Authprice, H(Common), Ban_B, R_B, Expiration}$_{Pk(SK_A)}$}

定义 $\Phi_2(P_B(Ban'_B))$ 为

{x_1 = ($Salt_B$, H(R_B, Ban_B))(x_3 = {Authprice, H(Common), Ban'_B, R_B, Expiration}$_{Pk(SK_A)}$}

显然对于销售方和网络偷听者,我们有 $\Phi_1(P_B(Ban_B)) \approx_s \Phi_2(P_B(Ban'_B))$,因此有 $P_B(Ban_B) \approx_s P_B(Ban'_B)$,表示购买方的账户不会泄漏给销售方和网络偷听者。但是对于转账方,由于 Slip. 3(Decrypt(x_3, K_A)) = Ban_B 在 $\Phi_1(P_B(Ban_B))$ 成立,但是在 $\Phi_2(P_B(Ban'_B))$ 中不成立,所以我们可推出 $P_B(Ban_B) \approx_s P_B(Ban'_B)$ 不成立,表示购买方对于转账方不保持匿名性。

4 结束语

本文采用了应用 pi 演算来验证 iKP 协议的认证性和匿名性。由此我们可以看出:作为 pi 演算的一个扩展,应用 pi 为电子支付协议的描述和论证提供了很好的支持;同样,这一个工作也可以应用在描述和验证其他的电子支付协议的性质中。

在接下来的工作中,我们计划继续使用 pi 演算及其扩展来验证电子支付协议的其他性质,如非否认性、公平性、原子性等,并且开发一个自动验证工具来完成这些性质的证明。

参考文献:

[1] Milner, J Parrow, *et al*. A Calculus of Mobile Processes, Parts I and II[J]. Information and Computation, 1992, 1-40, 41-77.

[2] M Abadi, C Fournet. Mobile Values, New Names and Secure Communication[C]. The 28th ACM Symposium on Principles of Programming Languages(POPL '01), 2001. 104-115.

[3] M Abadi, B Blanchet. Computer-assisted Verification of a Protocol for Certified Email[C]. Static Anlaysis, the 10th International Symposium(SAS '03), LNCS 2694, Springer-Verlag, 2003. 316-335.

[4] M Abadi, B Blanchet, C Fournet. Just Fast Keying in the pi Calculus [C]. The 13th European Symposium on Programming, LNCS 2986, Springer-Verlag, 2004. 340-354.

[5] M Abadi, C Fournet. Hiding Names: Private Authentication in the Applied pi Calculus[C]. The Proceedings of the International Symposium on Software Security(ISSS '02), LNCS 2609, Springer-Verlag, 2003. 317-338.

[6] M Bellare, *et al*. *i*KP: A Family of Secure Electronic Payment Protocols[C]. USENIX, 1995.

[7] M Bellare, J Garay, R Hauser, *et al*. Design, Implementation, and Deployment of the *i*KP Secure Electronic Payment System[J]. IEEE Journal of Selected Areas in Communications, 2000, 18: 611-627.

[8] T Y C Woo, S S Lam. A Semantic Model for Authentication Protocols [C]. The 14th IEEE Symposium on Research in Security and Privacy, IEEE Computer Society Press, 1993. 178-194.

作者简介:

顾永跟(1968-),男,在读工学博士,主要研究方向为安全协议、形式化方法;李国强(1979-),男,在读工学硕士,主要研究方向为进程演算、网络安全;王国均(1946-),男,浙江湖州人,教授,研究方向为算法设计与分析、图像处理。