电子商务安全协议及其非单调动态逻辑验证*

陈庆锋[1,2] 白硕[1] 王驹[3] 张师超[2] 隋立颖[1]

[1](国家智能计算机研究开发中心 北京 100080)

[2](广西师范大学数学与计算机科学系 桂林 541004)

[3](中国科学院软件研究所 北京 100080)

E-mail: bai@ncic.ac.cn

摘要 该文介绍了SET(secure electronic transactions)的付费业务流程,对NDL(non-monotomic dynamic logic)的逻辑框架进行了扩展,即针对SET协议,增加了新的公理,重新给出积累规则的定义.在此基础上,用对SET中的几个重要的范例进行的逻辑验证,说明了NDL在验证电子商务协议上的重要性,并初步提出了积累规则中需要进一步研究的问题.

关键词 信息安全,逻辑验证,电子商务,注册,付费.

中图法分类号 TP309

随着Internet国际互联网的日趋流行,它已经渗透到社会上的各行各业,在金融、教学、科研上起着重要作用.同时,由于互联网上传播信息量的增加,如何保证网上信息传播的安全成了人们日益关注的一个重要问题.在这个过程当中,因为电子商务与人们日常生活的关系最密切,它的安全性问题就成了最敏感的领域.实际上,电子商务对传统金融业的冲击已经是不可否认的事实,任何组织和个人只要在互联网上建立一个WWW站点地址,就可以通过网络进行购物和付费,可以说,Internet改变了人们传统的生活方式.正因为电子商务的重要性,为了提供一种安全、有效的购物方式,世界上的几大公司,如IBM,Microsoft,Netscape,RSA,SAIC,Terisa和Versign共同制定了SET(secure electronic transactions)电子商务安全协议[1,2].

以往对安全协议的验证有很多手段,一种是用常规的方法,对协议进行实际攻击,另一种是从表面上进行直观检测.但是,前者必须在系统已经建立之后才能检测出它的错误;后者由于检测手段不够严密,难免有疏漏之处.形式化逻辑方法的出现,使得我们可以在安全协议付诸实施之前,用一种严密可靠的方法对它的安全性质进行验证,正是由于它的严密性,使它成为信息安全领域中的一个重要研究方向.

在众多的验证逻辑,如BAN[3],GNY[4],AUTLOG[5]中,要么没有监听者,要么过于复杂,要么过于简单,并不能从根本上解决问题.Kailar[6]提出的方法比较明确地针对安全协议中的"责任性",他所建立的框架保证任何一方一旦做了某个动作,用逻辑的方法就能够证明,使他对所做过的动作无法抵赖.

而在文献[7]中提出的NDL(non-monotomic dynamic logic)验证逻辑,引入了通信安全问题上的动态性(dynamic)和非单调性(non-monotonic)两种概念,使它更加符合SET协议中电子商务的特征,也为我们对SET协议进行逻辑验证打下了坚实的基础.

本文的第1节是SET协议简介.第2节对NDL进行了扩展.第3节是对SET中的几个范例的验证.第4节对全文进行总结,并提出了积累规则的符号表示中可能存在的一些问题.

---

* 本文研究得到国家863高科技项目基金(No.863-306-ZD-10-02)资助.作者陈庆锋,1971年生,助理工程师,主要研究领域为信息安全,电子商务.白硕,1956年生,博士,研究员,博士生导师,主要研究领域为人工智能,计算机语言学,Internet/Intranet应用软件.王驹,1950年生,博士,研究员,主要研究领域为数理逻辑,计算机理论.张师超,1962年生,博士,教授,主要研究领域为人工智能,数据库技术.隋立颖,女,1973年生,硕士,主要研究领域为Internet/Intranet应用软件,计算机理论.

本文通讯联系人:白硕,北京 100080.国家智能计算机研究开发中心

本文1998-04-03收到原稿,1999-03-22收到修改稿

1 SET的付费业务流程简介

SET的付费业务处理[1,2]主要由持卡人注册、商家注册、购买请求付费认证、付费获得这几个部分组成.

1.1 持卡人注册

持卡人C向商家发送SET消息之前,必须向证书授权当局CA(certificate authorities)注册,同时为了能向CA发送SET消息,持卡人必须知道CA的数据交换公钥,它由CA的数据交换公钥证书提供.当持卡人要求CA的数据交换公钥证书时,处理过程开始.具体步骤如下:

·持卡人软件

(1) 持卡人向证书授权当局CA发出初始化请求.

·证书授权当局

(2) 证书授权当局收到初始化请求.

(3) 证书授权当局产生初始化回答,并产生初始化回答的消息文摘,然后用证书授权当局CA的签名私钥加密该消息文摘,得到初始化回答消息的数字签名.

(4) 证书授权当局向持卡人发送初始化回答、初始化回答消息的数字签名、CA的签名公钥证书和CA的数据交换公钥证书.

·持卡人软件

(5) 持卡人收到初始化回答,并通过PKI(public key infrastructure)信任树,即沿着信任链直到树根,逐级验证CA证书的合法性.

(6) 持卡人通过比较用CA的签名公钥解开CA签名得到的结果和收到的初始化回答消息产生的消息文摘是否一致来验证CA签名的合法性.

(7) 持卡人输入他的帐号.

(8) 持卡人软件产生注册表请求.

(9) 持卡人软件用一个随机产生的单钥 k_1 加密注册表请求消息,并用CA的数据交换公钥加密 k_1 与持卡人的帐号,形成数字信封.

(10) 持卡人软件将所产生的消息发送给CA.

·证书授权当局

(11) CA用他的数据交换私钥解出持卡人的单钥 k_1 及帐号,然后用 k_1 解出注册表请求.

(12) CA用帐号的前6~11位来识别持卡人的金融机构,并选定合适的注册表.CA产生注册表的消息文摘,然后用CA的签名私钥加密,得到注册表的数字签名.

(13) CA向持卡人发送注册表和他的签名公钥证书.

·持卡人软件

(14) 持卡人软件收到注册表,然后通过PKI信任树验证CA签名证书的合法性.

(15) 持卡人软件通过比较用CA签名公钥解开CA签名所得到的结果和自己产生的所收到的注册表消息的消息文摘是否一致,来验证CA签名的合法性.

(16) 持卡人软件产生一对签名密钥.

(17) 持卡人完成注册表(向注册表填信息,例如持卡人名称、届满日期、帐单地址及其他一些被金融机构用来辨别是否有效的持卡人的信息).

(18) 持卡人软件产生包含输入注册表信息的证书请求.

(19) 持卡人软件将证书请求、持卡人的签名公钥和一个新产生的单钥 k_2 一起形成一个新的消息,然后用持卡人的签名私钥对它签名.

(20) 持卡人软件用一个随机产生的单钥 k_3 加密第(19)步产生的结果,并用CA的数据交换公钥加密 k_3 与持卡人的帐户信息,形成数字信封.

(21) 持卡人软件将所产生的消息发送给 CA.

· 证书授权当局

(22) CA 用他的数据交换私钥解出 k_3 与持卡人帐号,然后用 k_3 解出证书请求.

(23) CA 通过比较用持卡人的签名公钥解开持卡人签名得到的结果和自己产生的所收到消息的消息文摘是否一致,来验证持卡人签名的合法性.

(24) CA 用持卡人的帐户信息和从注册表得到的信息验证证书请求的合法性.

(25) 基于第(24)步的验证,CA 产生持卡人签名公钥证书,并用 CA 的签名私钥对它签名.

(26) CA 产生证书回答,并用 CA 的签名私钥对它作数字签名.

(27) CA 用从持卡人请求得到 k_2 的加密第(26)步时所产生的消息.

(28) CA 将加密的证书回答、持卡人的签名公钥证书和 CA 的签名公钥一起发送给持卡人.

· 持卡人软件

(29) 持卡人软件通过 PKI 树验证 CA 的签名公钥证书和持卡人的签名公钥证书的合法性.

(30) 持卡人软件用第(19)步存储的 k_2 解出回答.

(31) 持卡人软件通过比较用 CA 的签名公钥解开 CA 签名得到的结果和回答消息新产生的消息文摘是否一致来验证 CA 签名的合法性.

(32) 持卡人软件为将来的电子交易存储签名公钥证书和从回答得到的相关信息.

1.2 商家注册

为了能从持卡人收到 SET 付费指令或能通过付费网关处理 SET 业务,商家 M 必须先向证书授权当局 CA 注册.同时,为了向 CA 发送 SET 消息,商家必须知道 CA 的数据交换公钥,它由 CA 的数据交换公钥证书提供.此外,商家还需要从商家金融机构得到注册表.

当商家软件要求 CA 的数据交换公钥证书和适合自己的注册表时,注册过程开始.具体步骤如下:

· 商家软件

(1) 商家软件向 CA 发出初始化请求.

· 证书授权当局

(2) CA 收到初始化请求.

(3) CA 选定合适的注册表,产生注册表的消息文摘,然后用 CA 的签名私钥加密,产生数字签名.

(4) CA 向商家发送注册表、CA 的签名公钥证书和数据交换公钥证书.

· 商家软件

(5) 商家软件收到注册表,并通过 PKI 验证 CA 的证书的合法性.

(6) 商家软件通过比较用 CA 的签名公钥解开 CA 签名得到的结果和收到的注册表消息新产生的消息文摘是否一致来验证 CA 签名的合法性.

(7) 商家软件产生两对密钥,一对为其签名密钥,另一对为其数据交换密钥.

(8) 商家完成注册表(例如姓名、地址、身份证号等).

(9) 商家软件产生证书请求.

(10) 商家软件将证书请求、商家产生的签名公钥和数据交换公钥一起形成一个消息,然后用商家的签名私钥加密该消息的消息文摘得到数字签名.

(11) 商家软件用一个随机产生的单钥 k_1 加密第(10)步产生的消息.然后用 CA 的数据交换公钥加密 k_1 和商家的帐户数据形成数字信封.

(12) 商家软件向 CA 发送加密的证书请求消息.

· 证书授权当局

(13) CA 用自己的数据交换私钥解出 k_1 和商家的帐户数据,然后用 k_1 解出证书请求消息.

(14) CA 通过比较用商家的签名公钥解开商家签名得到的结果和证书请求消息新产生的消息文摘是否一致,来验证商家签名的合法性.

(15) CA 用商家的帐户信息和注册表上的信息来确认证书请求的合法性.

(16) 基于上述验证,CA 用自己的签名私钥对 CA 产生的商家证书签名.

(17) CA 产生证书回答,并用自己的签名私钥对它作数字签名.

(18) CA 向商家发送回答.

· 商家软件

(19) 商家软件通过 PKI 树验证收到的证书的合法性.

(20) 商家软件通过比较用 CA 的签名公钥解开 CA 签名得到的结果和回答消息新产生的消息文摘是否一致,来验证 CA 签名的合法性.

(21) 商家软件为将来的电子交易存储证书和从回答得到的消息.

1.3 购买请求

当持卡人已完成浏览、选择和定货后,SET 协议才起作用. 为了向商家发送 SET 消息,持卡人必须知道付费网关 P 的数据交换密钥. 当持卡人软件要求付费网关的数据交换公钥证书时,SET 的订购处理开始. 持卡人发出的消息表明,在交易中将使用哪一种付费卡品牌. 具体步骤如下:

· 持卡人软件

(1) 持卡人完成浏览和选购.

(2) 持卡人软件向商家发送初始化请求.

· 商家软件

(3) 商家软件收到初始化请求.

(4) 商家软件产生回答,并用商家的签名私钥加密回答的消息文摘,产生回答的数字签名.

(5) 商家软件向持卡人发送回答,商家的签名公钥证书和付费网关 P 的数据交换公钥证书.

· 持卡人软件

(6) 持卡人软件收到初始化回答,并通过 PKI 树验证商家和付费网关证书的合法性.

(7) 持卡人软件通过比较用商家签名公钥解开商家签名得到的结果和回答消息新产生的消息文摘是否一致,来验证商家签名的合法性.

(8) 持卡人软件用从浏览和选购阶段得到的信息产生订购信息 OI(order information).

(9) 持卡人软件完成付费指令 PI(payment instruction).

(10) 持卡人软件用自己的签名私钥产生 OI 和 PI 的双重签名.

(11) 持卡人软件用一个随机产生的单钥 k_1 加密 PI 的双重签名,然后将 k_1 与持卡人的帐户信息一起,用付费网关的数据交换公钥加密,形成数字信封.

(12) 持卡人软件向商家发送 OI、加密后的 PI、双重签名和持卡人的签名证书.

· 商家软件

(13) 商家软件通过 PKI 树验证持卡人证书的合法性.

(14) 商家软件用持卡人签名公钥解开 OI 的双重签名,通过比较得到的结果和 OI 及 PI 的消息文摘连接后得到的消息新产生的消息文摘是否一致,来验证 OI 上双重签名的合法性.

(15) 商家 M 处理请求(包括将付费指令 PI 提交付费网关 P 认证).

(16) 商家软件产生包含有商家签名证书的购买回答,然后用商家的签名私钥对它作数字签名.

(17) 商家软件向持卡人传送购买回答和他自己的签名公钥证书.

(18) 若业务被认证,商家履行持卡人的业务要求(例如,发货).

· 持卡人软件

(19) 持卡人软件通过 PKI 树验证商家签名的合法性.

(20) 持卡人软件通过比较用商家的签名公钥解开商家签名得到的结果和购买回答消息新产生的消息文摘是否一致,来验证商家签名的合法性.

(21) 持卡人软件存储购买回答.

1.4 付费认证

当处理持卡人的定货时,商家将认证该项业务.具体步骤如下:

·商家软件

(1) 商家软件产生认证请求.

(2) 商家软件用自己的签名私钥加密认证请求的消息文摘,产生认证请求的数字签名.

(3) 商家软件用一个随机产生的单钥 k_2 加密认证请求和其数字签名.然后用付费网关的数据交换公钥加密 k_2,形成数字信封.

(4) 商家软件将加密的认证请求及数字签名、从持卡人的购买请求得到的加密的 PI、持卡人的签名公钥证书和商家的证书传送给付费网关.

·付费网关

(5) 付费网关用 PKI 树验证商家证书的合法性.

(6) 付费网关用自己的数据交换私钥解出 k_2,并用 k_2 解出认证请求.

(7) 付费网关通过比较用商家签名公钥解开商家签名得到的结果和认证请求消息新产生的消息文摘是否一致,来验证商家签名的合法性.

(8) 付费网关用 PKI 树验证持卡人的签名公钥证书的合法性.

(9) 付费网关用自己的数据交换私钥解出 k_1 和持卡人的帐户信息.然后用 k_1 解出 PI.

(10) 付费网关利用持卡人的签名公钥验证持卡人在 PI 上的双重签名的合法性.

(11) 付费网关确认商家的认证请求与持卡人的付费指令 PI 之间的一致性.

(12) 付费网关通过金融网把认证请求传给持卡人的金融机构.

(13) 付费网关产生认证回答,然后用付费网关的签名私钥加密认证回答的消息文摘,得到认证回答消息的数字签名.

(14) 付费网关用一个新的随机产生的单钥 k_3 加密认证回答.然后用商家的数据交换公钥加密 k_3 和持卡人的帐户信息,形成数字信封.

(15) 付费网关产生获得令牌(CapToken),用付费网关的签名私钥加密获得令牌的消息文摘,得到数字签名.

(16) 付费网关用一个新的随机产生的单钥 k_4 加密获得令牌.然后用付费网关的数据交换公钥加密 k_4 跟持卡人的帐户信息,形成数字信封.

(17) 付费网关向商家传送认证回答.

·商家软件

(18) 商家软件用 PKI 树验证付费网关签名公钥证书的合法性.

(19) 商家软件用商家的数据交换私钥解出 k_3,然后用 k_3 解出认证回答.

(20) 商家软件通过比较用付费网关的签名公钥解开付费网关签名得到结果和认证回答消息新产生消息文摘是否一致,来验证付费网关签名的合法性.

(21) 商家软件为以后的获得处理而存储加密的获得令牌和信封.

(22) 商家完成购买请求处理.

1.5 付费获得

当完成持卡人的定货处理后,商家将请求付费.具体步骤如下:

·商家软件

(1) 商家软件产生获得请求.

(2) 商家软件在获得请求中加入商家的证书,并用商家的签名私钥加密获得请求的消息文摘,产生数字签名.

(3) 商家软件用一个随机产生的单钥 k_5 加密获得请求,然后用付费网关的数据交换公钥加密 k_5,形成数字信封.

(4) 商家软件向付费网关传送加密的获得请求和以前从认证回答存储的加密的获得令牌及商家的证书.

·付费网关

(5) 付费网关用 PKI 树验证商家证书的合法性.

(6) 付费网关用自己的数据交换私钥解出 k_5,然后用 k_5 解出获得请求.

(7) 付费网关通过比较用商家签名公钥解开商家签名得到的结果和获得请求消息新产生的消息文摘是否一致,来验证商家签名的合法性.

(8) 付费网关用自己的数据交换私钥解出 k_4,然后用 k_4 解出获得令牌.

(9) 付费网关确认商家的获得请求和获得令牌之间的一致性.

(10) 付费网关通过一个金融网向持卡人的金融机构发送获得请求.

(11) 付费网关产生获得回答信息,包括付费网关的签名证书,并用付费网关的签名私钥加密获得回答消息的消息文摘,产生数字签名.

(12) 付费网关用一个新的随机产生的单钥 k_6 加密获得回答,然后用商家的数据交换公钥加密 k_6.

(13) 付费网关向商家发送加密的获得回答.

·商家软件

(14) 商家软件用 PKI 树验证付费网关证书的合法性.

(15) 商家软件用自己的数据交换私钥解出 k_6,然后用 k_6 解出获得回答.

(16) 商家软件通过比较用付费网关的签名公钥解开付费网关签名得到的结果和获得回答消息新产生的消息文摘是否一致,来验证付费网关签名的合法性.

(17) 为了与请求者收到的付费保持一致,商家软件存储获得回答.

上面给出了 SET 协议的 5 个付费流程的介绍,为了对 SET 进行逻辑验证,必须扩展 NDL 的公理系统和积累规则.下面,我们将扩充 NDL 的公理及重新定义积累规则.

2 NDL 逻辑系统的扩展

2.1 扩展的动作

在文献[7]的 NDL 框架的基础上,结合 SET 安全协议的具体内容,我们在 NDL 系统里扩充一个基本动作:验证(verify)和合法性(legal).动作 Verify 的作用是,当参与方收到其他参与方发来的证书时,要通过树状链结构 PKI 对证书的合法性逐级进行验证,为了能够更好地表示这一认证过程,在扩展的公理中引入了这个动作,其中 x 为动作执行者,Cert 为要验证的证书,$\langle CA, X_2, \ldots, X_{n-1}, root\rangle$ 为 PKI 树的各级证书授权当局.动作 $Legal(CA, CerReq)$ 表示 CA 验证注册表请求合法性这个过程,CA 是动作的执行者.

$Verify(x, Cert, \langle CA_1, CA_2, \ldots, CARoot\rangle)$ x 沿 PKI 逐级验证由 CA_1 发出的证书 Cert 的合法性.此动作仅在 $Know(x, Spb(CARoot))$ 和 $Know(x, Cert)$ 同时成立时才可以做.

$Legal(CA, CertReq)$ CA 验证注册表请求的合法性.

2.2 扩展的谓词

在文献[1]的 NDL 框架的基础上,增加一个 IsVerified 谓词.

$IsVerified(x, CA, Cert)$ 当 x 作完动作 Verify 后,若证明证书 Cert 是合法的,则 $IsVerified(x, Cert, CA)$ 的值为真,否则为假.其真假值依赖于动作 Verify 的成功与否,即依赖于证书 Cert 的合法与否(其值不能由我们决定,是开放的).CA 是发放 Cert 的证书授权机构.

2.3 扩展的公理

(1) 加密公理

1-1 $Know(x, m) \wedge Know(x, k) \rightarrow Know(x, E(m, k))$

1-2 $Know(x,m) \wedge Know(x,Kpb(y)) \rightarrow Know(x,S(m,Kpb(y)))$

(2) 密钥分配公理

2-1 $Know(x,Kpv(x))$

2-2 $Know(x,Spv(x))$

2-3 $Know(x,Spb(x))$

2-4 $Know(x,Kpb(x))$

2-5 $Know(x,Spb(CARoot))$

2-6 $Know(x,Kpb(CARoot))$

(3) 解密公理

3-1 $Know(x,k) \wedge Know(x,E(m,k) \rightarrow Know(x,m)$

3-2 $Know(x,Kpv(y)) \wedge Know(x,S(m,Kpb(y))) \rightarrow Know(x,m)$

(4) 签名公理

4-1 $Know(x,m) \rightarrow Know(x,H(m))$

4-2 $Know(x,m) \wedge Know(x,Spv(y)) \rightarrow Know(x,S(H(m),Spv(y)))$

(5) 认证公理

5-1 $Know(x,m) \wedge Know(x,S(H(m),Spv(y))) \wedge Know(x,Spb(y)) \rightarrow Auth(x,y,m)$

5-2 $Know(x,m) \wedge Auth(x,y,H(m)) \rightarrow Auth(x,y,m)$

(6) 分合公理

6-1 $Know(x,\langle m_1,\ldots,m_n\rangle) \leftrightarrow Know(x,m_1) \wedge \ldots \wedge Know(x,m_n)$

6-2 $Auth(x,y,\langle m_1,\ldots,m_n\rangle) \rightarrow Auth(x,y,m_1) \wedge \ldots \wedge Auth(x,y,m_n)$

(7) PKI 公理

7-1 $IsVerified(x,CA,CertS(y)) \rightarrow Auth(x,CA,\langle y,Spb(y)\rangle)$

7-2 $IsVerified(x,CA,CertK(y)) \rightarrow Auth(x,CA,\langle y,Kpb(y)\rangle)$

在这里,CA 指的是发放 $CertS$ 或 $CertK$ 给 y 的那个 CA.

以上公理是在 NDL 框架的基础上结合 SET 安全协议的具体内容对 NDL 公理系统的扩充,在稍后的范例验证中我们将会看到它的应用.

2.4 规则的扩展

2.4.1 新增的规则

在文献[7]中提出的 NDL 框架的基础上,结合 SET 安全协议的具体内容,我们还需对 NDL 规则系统进行如下的扩充,新增加一条验证规则:

(R-7) PKI 验证规则

$$\vdash_{Verify(x,Cert,\langle CA_1,\ldots,CARoot\rangle)} IsVerified(x,CA,Cert)$$

正如我们前面所说的,$IsVerified(x,CA,Cert)$成立与否是依赖于动作 Verify 的结果的,而只有当($Know\ x$, $Cert$)和 $Know(x,Spb(CARoot))$同时成立时,才有可能做动作 Verify,且此规则仅适用于 Verify 验证成功的情况.

2.4.2 积累规则的扩展

在文献[7]中提到的积累规则只适用于在 SET 业务处理过程中,不更换密钥并且对消息有"记忆"功能的情况.但是,在实际业务中,密钥是允许修改的,而且对某些消息可能没有"记忆"功能,为了能够满足在电子商务中可能发生的各种情况,在本文中,我们扩展了积累规则.

在电子商务中,SET 协议最可能发生改变的是密钥 k,$Spb(x)$,$Spv(x)$,$Kpb(x)$,$Kpv(x)$.对于$\langle Spv(x),Spb(x)\rangle$与$\langle Kpv(x),Kpb(x)\rangle$,它们分别是同时产生的签名密钥对和数据交换密钥对,因此,更改其中的任何一个,另一个也会发生改变.除了密钥可能会发生改变外,某些消息也会发生改变,总的来说大致可以分为以下 3 种情况:

(1) k,$\langle Spb(x),Spv(x)\rangle$,$\langle Kpb(x),Kpv(x)\rangle$中有改变,但有"记忆"功能.

(2) k,$\langle Spb(x),Spv(x)\rangle$,$\langle Kpb(x),Kpv(x)\rangle$都不改变,但没有"记忆"功能.

(3) k,$\langle Spb(x),Spv(x)\rangle$,$\langle Kpb(x),Kpv(x)\rangle$中有改变,且没有"记忆"功能.

(1)又可分为两种情况.一种是x随机产生密钥k,$\langle Spb(x),Spv(x)\rangle$或$\langle Kpb(x),Kpv(x)\rangle$,但相应地让$y$知道$k$,$Spb(x)$或$Kpb(x)$.由于有"记忆"功能,其他消息仍然存在,因此,文献[7]中的积累规则仍然起作用.另一种是在执行过程中,x自己单方面更改了密钥,但没有让y知道.这时候,y对原来密钥的知识已经"作废",积累规则不再起作用.根据上面两种情况,可以相应地得到以下两条规则.

(R-3-1) 积累规则 1

$$\frac{P\vdash_a Q, Know(y,newkey)}{P\vdash_{a\cdot Generate(x,newkey)} Q}$$

在整个动作发生的过程中,x改变密钥,且让对方知道.那么已证明成立的结论经过动作$Generate(x,newkey)$后仍然成立.其中$newkey$为k,$Spb(x)$,$Kpb(x)$中的任何一个.注意,$Spv(x)$和$Kpv(x)$不能让y知道.

(R-3-2) 积累规则 2

$$\frac{P\vdash_a Q, \neg Know(y,newkey)}{P\not\vdash_{a\cdot Generate(x,newkey)} Q}$$

x改变了密钥,但没让y知道,那么原来成立的结论Q,现在可以非单调地说,经过动作$Generate(x,newkey)$后,Q不成立.

(2)是密钥没有改变,但是实体在整个协议执行过程中对某些消息没有"记忆"功能.

(R-3-3) 积累规则 3

$$\frac{P\vdash_a Q, Equal(m,m')}{P\vdash_{a\cdot Generate(x,m)} Q}$$

m'为几个消息的集合,即$\langle m_1,m_2,\ldots,m_n\rangle$.$x$随机产生新消息$m$,且$m$与原来产生的消息$m'$一致,没有发生变化,则经过动作$Generate(x,m)$后,$Q$仍然成立.

(R-3-4) 积累规则 4

$$\frac{P\vdash_a Q, \neg Equal(m,m')}{P\not\vdash_{a\cdot Generate(x,m)} Q}$$

x产生新消息m,但是m与原来产生的消息m'不一致,已经发生了变化.那么可以非单调地说,经过动作$Generate(x,m)$后,Q不成立.

(3)是密钥改变,且实体在协议执行过程中对消息没有记忆功能.它可以分为以下 4 种情况:

(R-3-5) 积累规则 5

$$\frac{P\vdash_a Q, Know(y,newkey), Equal(m,m')}{P\vdash_{a\cdot Generate(x,newkey)\cdot Generate(x,m)} Q}$$

在协议执行过程中,x改变密钥,且让y知道发生改变的密钥.同时,x新产生的消息m与原来的消息m'保持一致,则经过动作$Generate(x,newkey)\cdot Generate(x,m)$后,$Q$仍然成立.

(R-3-6) 积累规则 6

$$\frac{P\vdash_a Q, Know(y,newkey), \neg Equal(m,m')}{P\not\vdash_{a\cdot Generate(x,newkey)\cdot Generate(x,m)} Q}$$

在协议执行过程中,x改变密钥,且让y知道发生改变的密钥.但是,x新产生的消息m与原来的消息m'不一致,则可以非单调地说,经过动作$Generate(x,newkey)\cdot Generate(x,m)$后,$Q$不成立.

(R-3-7) 积累规则 7

$$\frac{P\vdash_a Q, \neg Know(y,newkey), Equal(m,m')}{P\not\vdash_{a\cdot Generate(x,newkey)\cdot Generate(x,m)} Q}$$

在协议执行过程中,x改变密钥,但是不让y知道.x新产生的消息m与原来的消息m'保持一致,则可以非单调地说,经过动作$Generate(x,newkey)\cdot Generate(x,m)$后,$Q$不成立.

(R-3-8) 积累规则 8

$$\frac{P \vdash_{\alpha} Q, \neg Know(y, newkey), \neg Equal(m, m')}{P \mid\sim_{\alpha \cdot Generate(x,newkey) \cdot Generate(x,m)} Q}$$

在协议执行过程中,x 改变密钥,但不让 y 知道.且 x 新产生的消息 m 与原来的消息 m' 不一致,则可以非单调地说,经过动作 $Generate(x,newkey) \cdot Generate(x,m)$ 后,Q 不成立.

扩展后的积累规则,综合考虑了电子商务中可能出现的各种意外情况.但必须指出,对于规则中的一些符号的描述仍然存在一些问题,像密钥的表示符号,只要对象 x 被指定后,它似乎不再发生改变,实际上它随着时间的变化也在不断地改变.因此,如果要使规则的定义更加准确,必须对现有的符号系统进行改进,对此我们将另文讨论.

3 SET 协议验证示例

3.1 SET 协议的片段的验证示例

下面,我们将用"持卡人注册"和"商家注册"阶段中的两个例子来说明扩展后的 NDL 逻辑框架在 SET 协议验证上的应用.

例 1:已知

$P=\{Know(C,Spb(CARoot)), Know(C,Kpb(CARoot))\}$,

$\alpha=Generate(CA,InitRes) \cdot Send(CA,C,Sign(CA,InitRes)) \cdot$
$Send(CA,C,CertS(CA)) \cdot Send(CA,C,CertK(CA)) \cdot$
$Verify(C,CertS(CA),\langle X_2,\ldots,X_{n-1},root\rangle) \cdot$
$Verify(C,CertK(CA),\langle X_2,\ldots,X_{n-1},root\rangle)$,

$Q=\{Auth(C,X_2,\langle CA,Spb(CA)\rangle); Auth(C,X_2,\langle CA,Kpb(CA)\rangle)\}$.

求证:$P \vdash_{\alpha} Q$

证明:

(1)	$Know(C,Spb(CARoot))$	[前提]
(2)	$Know(C,Kpb(CARoot))$	[前提]
(3) $Generate(CA,InitRes)$		[动作]
(4)	$Know(CA,InitRes)$	(3)[R-2]
(5)	$Know(CA,Spv(CA))$	[2-2]
(6)	$Know(CA,S(H(InitRes),Spv(CA)))$	(4)(5)[4-2]
(7)	$Know(CA,\langle InitRes,S(H(InitRes),Spv(CA))\rangle)$	(4)(6)[6-1]
(8)	$Know(CA,Sign(CA,InitRes))$	(7)[定义]
(9) $Send(CA,C,Sign(CA,InitRes))$		[动作]
(10) $Send(CA,C,CertS(CA))$		[动作]
(11) $Send(CA,C,CertK(CA))$		[动作]
(12)	$Know(C,Sign(CA,InitRes))$	(9)[R-1]
(13)	$Know(C,CertS(CA))$	(10)[R-1]
(14)	$Know(C,CertK(CA))$	(11)[R-1]
(15) $Verify(C,CertS(CA),\langle X_2,\ldots,X_{n-1},CARoot\rangle)$		(1)(13)[动作]
	/* 如果检验失败,C 没有在 PKI 树中找到 $root$,则持卡人停止注册. */	
(16)	$IsVerified(C,X_2,CertS(CA))$	(15)[R-6]
(17)	$Auth(C,X_2,\langle CA,Spb(CA)\rangle)$	(16)[7-1]
(18) $Verify(C,CertS(CA),\langle X_2,\ldots,X_{n-1},CARoot\rangle)$		(1)(14)[动作]
	/* 如果检验失败,C 没有在 PKI 树中找到 $root$,则持卡人停止注册. */	

(19) $IsVerified(C,X_2,CertK(CA))$ (18)[R-6]

(20) $Auth(C,X_2,\langle CA,Kpb(CA)\rangle)$ (19)[7-1]

/* $CA,X_2,\ldots,X_{n-1},CARoot$ 为 PKI 树的各级证书授权当局. */

式(18)和(20)即所要证明的结果. □

例 2:已知

$P=\{Know(M,Acct(M)),Know(M,Sign(M,\langle CertReq,Spb(M),Kpb(M)\rangle)),Know(CA,Spb(M)),Know(M,Kpb(CA))\}$,

$\alpha=Generate(M,k_1)\circ Send(M,CA,E(Sign(M,\langle CertReq,Spb(M),Kpb(M)\rangle),k_1))\circ Send(M,CA,S(\langle Acct(M),k_1\rangle,Kpb(CA)))\circ Legal(CA,CerReq)$,

$Q=\{Auth(CA,M,\langle CertReq,Spb(M),Kpb(M)\rangle)\}$.

求证:$P\vdash_{\alpha}Q$

证明:

(1) $Know(M,Acct(M))$ [前提]

(2) $Know(M,Sign(M,\langle CertReq,Spb(M),Kpb(M)\rangle))$ [前提]

(3) $Know(CA,Spb(M))$ [前提]

(4) $Generate(M,k_1)$ [动作]

(5) $Know(M,k_1)$ (4)[R-2]

(6) $Know(M,E(Sign(M,\langle CertReq,Spb(M),Kpb(M)\rangle),k_1))$ (2)(5)[1-1]

(7) $Know(M,\langle Acct(M),k_1\rangle)$ (1)(5)[6-1]

(8) $Know(M,Kpb(CA))$ [前提]

(9) $Know(M,S(\langle Acct(M),k_1\rangle,Kpb(CA)))$ (7)(8)[1-2]

(10) $Send(M,CA,E(Sign(M,\langle CertReq,Spb(M),Kpb(M)\rangle),k_1))$ [动作]

(11) $Send(M,CA,S(\langle Acct(M),k_1\rangle,Kpb(CA)))$ [动作]

(12) $Know(CA,E(Sign(M,\langle CertReq,Spb(M),Kpb(M)\rangle),k_1))$ (10)[R-1]

(13) $Know(CA,S(\langle Acct(M),k_1\rangle,Kpb(CA)))$ (11)[R-1]

(14) $Know(CA,Kpv(CA))$ [2-1]

(15) $Know(CA,\langle Acct(M),k_1\rangle)$ (13)(14)[3-2]

(16) $Know(CA,Acct(M))$ (15)[6-1]

(17) $Know(CA,k_1)$ (15)[6-1]

(18) $Know(CA,Sign(M,\langle CertReq,Spb(M),Kpb(M)\rangle))$ (12)(17)[3-1]

(19) $Auth(CA,M,\langle CertReq,Spb(M),Kpb(M)\rangle)$ (3)(16)[定理 2]

(20) $Know(CA,CertReq)$ (18)[定义]

(21) $Legal(CA,CerReq)$ [动作]

/* CA 用已知的商家的信息检验 $CertReq$ 中的注册表的合法性. */

(19)式即为所要证明的结果. □

第 1 个例子属于"持卡人注册"阶段,第 2 个例子属于"商家注册"阶段.在第 1 个范例证明中,都使用了动作 $Verify(x,Cert,\langle CA,X_2,\ldots,X_{n-1},root\rangle)$,因为接收方在验证发送方传给自己的证书时,要通过树状链结构 PKI 对证书的合法性逐级进行验证,为了能够更好地表示这一认证过程,在扩展的公理中引入了这个动作,其中 x 为动作执行者,Cert 为要验证的证书,$\langle CA,X_2,\ldots,X_{n-1},root\rangle$ 为 PKI 树的各级证书授权当局.动作 $Legal(CA,CerReq)$ 表示 CA 验证注册表请求合法性这个过程,CA 是动作的执行者.从上面 3 个范例的验证可以看出,不引入这些符号,要表示这两个动作是非常困难的事,而用简洁的符号来表示这些复杂且难以描述的动作,就使人很容易理解.

3.2 验证逻辑的 Prolog 程序实现

由于逻辑推导所涉及的东西大多是符号和规则,此推理过程若要由人来完成,则无论在人力上,还是在时间上均是一个极大的浪费.但逻辑推理具有易在机器上验证的特点,因此我们编制了基于 NDL 逻辑框架的Prolog "安全协议验证系统",其规则系统即为 NDL 的公理和规则,用户可以根据增加的需要输入前提,来达到验证安全协议是否存在漏洞的目的.

4 结束语

上面介绍了 SET 协议的付费业务流程,对 NDL 的公理系统和积累规则进行了扩展,并用 3 个例子的验证来说明 NDL 在验证安全协议上的应用.可以看出,扩展后的 NDL 逻辑框架能对相当大一部分安全协议的安全性进行验证,对于每种安全协议,只要在现有的逻辑系统的基础上稍加扩充,就可以用来对很多问题进行证明.

但是应该看到,对于如何用符号准确地表示密钥和消息的变化,需要我们做进一步的研究.本文主要介绍 SET 协议的内容和 NDL 逻辑框架的扩展.只是用 SET 中几个简单的例子说明了 NDL 的应用,并没有给出 SET 付费业务流程的完整的验证过程,对此也需要另文进行研究.

在上面 3 个例子的证明过程中,我们已经发现了 SET 协议中可能存在的一些问题,在对 SET 进行全面的验证之后,我们将会指出 SET 中的一些漏洞,并探讨它的解决方法.

参考文献

1 SET Secure Electronic Transaction Specification. Book 1: Business Description Version 1.0. May 31, 1997

2 SET Secure Electronic Transaction Specification. Book 2: Programmer's Guide Version 1.0. May 31, 1997

3 Burrows M, Abadi M, Needham R. A logic of authentication. ACM Transactions on Computer System, 1990,8(1):18～36

4 Abadi M, Tuttle M. A semantics for a logic of authentication. In: Proceedings of the 10th Annual ACM Symposium on Principles of Distributed Computing. Montreal: ACM Press, 1991. 201～216

5 Kessler V, Wedel C. AUTLOG—an advanced logic of authentication. In: Wernered B ed. Proceedings of the 7th IEEE Computer Security Foundations Workshop. Los Alamitors, CA: IEEE Computer Society Press, 1994. 90～99

6 Kailar R. Accountability in electronic commerce protocols. Proceedings of the IEEE Transactions on Software Engineering, 1996,22(5):313～328

7 Bai Shuo, Sui Li-ying, Chen Qing-feng *et al*. Authentication logic for secure protocols. Journal of Software, 2000,11(2):213～221
(白硕,隋立颖,陈庆锋等.安全协议的验证逻辑.软件学报,2000,11(2):213～221)

The Secure Electronic Transactions Protocol and Its Logical Verification with Non-Monotomic Dynamic Logic

CHEN Qing-feng[1,2]　BAI Shuo[1]　WANG Ju[3]　ZHANG Shi-chao[2]　Sui Li-ying[1]

[1](*National Research Center for Intelligent Computing Systems　Beijing*　100080)

[2](*Department of Mathemetics Computer Science　Guangxi Normal University　Guilin*　541004)

[3](*Institute of Software　The Chinese Academy of Sciences　Beijing*　100080)

Abstract　This paper introduces the payment process of SET (secure electronic transactions) protocol, and extends the logical framework of NDL to fit the purpose of logical verification of SET. This means to add some new axioms about SET protocols and redefine some given inference rules. Based on these, the logical verification for key fragments of SET protocol is given to show the importance of NDL and its extensions in E-commerce. Topics for further research on the "Rule of Accumulation" are also proposed.

Key words　Information security, logical verification, electronic commerce, registration, payment.