基于计算语义的安全协议验证逻辑

唐朝京[1],鲁智勇[2],冯 超[1]
(1.国防科技大学电子科学与工程学院,湖南长沙 410073;2.中国洛阳电子装备试验中心,河南洛阳 471003)

摘 要: 提出了一个基于计算语义的安全协议验证逻辑,能准确描述安全协议中的各种计算行为和通信行为.设计了基于该逻辑的证明系统,能对密码学中常用加密算法的各类安全属性规范直接描述,具有密码学可靠性.发现了计算协议组合逻辑在加密算法安全性建模时存在的不可靠性,并提出了解决方法.通过对 Needham-Schroeder-Lowe 协议安全性的证明,验证了逻辑的证明能力.与大部分验证方法不同的是,本逻辑属于由密码学算法安全性到协议安全性的正向推理方法,兼具符号方法的易用性和计算方法的可靠性.

关键词: 加密算法; 安全性; 形式逻辑; 计算语义
中图分类号: TP309 **文献标识码:** A **文章编号:** 0372-2112 (2014)06-1179-07
电子学报 URL: http://www.ejournal.org.cn **DOI:** 10.3969/j.issn.0372-2112.2014.06.022

A Verification Logic for Security Protocols Based on Computational Semantics

TANG Chao-jing[1], LU Zhi-yong[2], FENG Chao[1]
(1. *College of Electronic Science and Engineering, National University of Defense Technology, Changsha, Hunan* 410073, *China*;
2. *LEETC, Luoyang, Henan* 471003, *China*)

Abstract: This paper proposes a logic for verifying security protocols in the computational model, which can describe the computation and communication actions precisely. We present a cryptographically sound proof system on the logic and can formalize the various security properties for encryption algorithms directly. During the research, several unsoundness of the formalization with the computational protocol compositional logic in prior work is discovered, and corresponding solutions are proposed. By proving the security properties for the Needham-Schroeder-Lowe protocol, the power of the logic is demonstrated. Being different from most of the current verification approaches, this logic reasons about the security of protocols from the security of cryptographic algorithms forwardly, and is both easy to use and cryptographically sound.

Key words: encryption algorithms; security properties; formal logics; computational semantics

1 引言

安全协议是一种特殊形式的分布式软件,与其他形式的软件相比,安全协议除了要保证正确性外,更需要保证安全性,即在任何输入的情况下,协议都满足安全规范.如何使用形式化方法对已有协议的安全性进行准确、高效的分析和验证成了国内外相关领域的热点研究问题.

经过了近30年的发展,安全协议形式化验证领域呈现两大类研究方法:基于符号模型的验证方法[1,2]和基于计算模型的验证方法[3].为了结合两种方法的优点,得到既简单又具有密码学可靠性的验证方法,文献[4]进行了相关研究;Blanchet B 提出了一个计算模型下的自动化证明工具 CryptoVerif[5];Kramer S 提出了密码协议逻辑(CPL)[6];Troina A 等人提出在安全协议验证中引入敌手的破解概率[7];Datta A 等人提出了计算语义下的协议组合逻辑(CPCL)[8];由于其公理系统的缺陷[9],CPCL 目前对公钥加密算法的推理支持不够,不能对基于公钥加密算法的认证协议进行证明.Durgin N 等人尝试使用 PCL 对公钥加密认证协议进行验证[10],但由于其公理系统不完备,证明结果存在缺陷.

从安全协议的构建要素看,协议的安全性依赖于两个方面:密码学算法的安全性和消息逻辑的安全性.计算方法之所以具有密码学可靠性,是因为其将协议的安

收稿日期:2013-03-04;修回日期:2013-07-16;责任编辑:梅志强
基金项目:国家自然科学基金(No.61301236,No.61303061)

全性归约到密码学算法的安全性,但属于反向推理的归约方法.

为解决加密算法安全协议验证,本文提出了一个密码学可靠的加密算法安全性形式逻辑,设计了描述协议行为的逻辑,构建了公理系统,应用于对协议安全性的推理中,并对 NSL 协议的认证性进行了证明,验证了该逻辑的证明能力.

2 协议演算

在定义协议演算的基础上,对密码学安全实验和安全协议进行统一描述.演算使用比特串表示协议消息,使用概率多项式时间函数表示协议中密码学算法和敌手计算能力.

2.1 演算语法

演算使用 η 表示安全参数,用于记录协议中使用密钥的长度,语法规则如下:

$M,N::=x[i]$;

$P,Q::=P\mid Q\mid !^{i\leqslant n}P\mid \mathrm{Nil}\mid \mathrm{receive}\ x[i]:T;P\mid \mathrm{send}\ M:T;P\mid \mathrm{new}\ m[i]:T;P\mid x[i]:=f(M_1:T_1,\cdots,M_m:T_m);P\mid \mathrm{if}\ m[i]\ \mathrm{then}\ P\ \mathrm{else}\ Q$.

演算使用 T 表示协议中使用的不同消息类型.对于每个安全参数 η,每个类型 T 对应集合 Bitstring$\cup\{\perp\}$的子集 $I\eta(T)$.其中 Bitstring 表示所有比特串的集合,$\perp$表示特殊符号.演算中预定义了一些类型,包括:bool 类型,其中 $I\eta(\mathrm{bool})=\{\mathrm{true},\mathrm{false}\}$.

协议中使用的函数 f 符号由 $f:T_1\times\cdots\times T_n\to T$ 声明,每个函数 f 对应的解释 $I_\eta(f)$为一个从 $I_\eta(T_1)\times\cdots\times I_\eta(T_n)$到 $I_\eta(T)$的函数.这里要求 $I_\eta(f)(x_1,\cdots,x_n)$在 $x_1,\cdots,x_n$ 的长度和 η 的多项式时间内可计算.为了方便表述,演算中预定义了一些函数,主要包括:中缀形式的 $M=N$ 用于等价测试,$M\neq N$ 用于不等价测试(两个函数都输入同类型参数 M 和 N,输出 bool 类型的结果);$(_,\cdots_):T_1\times\cdots\times T_n\to$bitstring,用于将不同类型的消息串接成一个比特串.

演算通过进程描述安全协议."|"为并行操作符号,$P\mid Q$ 表示进程 P 和进程 Q 并行执行(P 和 Q 称为并行子进程);$!^{i\leqslant n}P$ 表示 n 个 P 的进程副本并行执行,每个副本由 $P[i]$表示.若 $n=1$,$P[i]=P$.进程 $P[i]$中的变量 m 用$m[i]$表示,在上下文明确时,可以将$[i]$省略.为了表示方便,文中将 $P::=P_1$简写为$[P_1]P$.

并不是所有符合语法规范的进程都是可执行的,为了对正确执行语义的进程进行判定,需对有效进程进行定义.

定义 1(有效进程) P 为有效进程,当且仅当其中所有变量满足以下条件之一:(1)由接收语句绑定;(2)由生成语句绑定;(3)由赋值语句绑定;(4)自由变量;(5)相同变量的两次绑定在条件语句的不同分支中.

对于不满足有效进程定义的进程 P,需先对其变量重命名,再进行验证.本文以下章节考虑的进程均为有效进程.

2.2 演算语义

进程的执行语义定义在进程执行迹上.每个执行迹 $T=[S_1,\cdots,S_n,\cdots]$表示进程的一次运行,其中二元组 $S_i=\langle T,\sigma\rangle$表示执行迹的状态,$T$ 为目前为止该次执行中所有已执行语句的记录,σ 为由变量到比特串的映射.对于 $i\leqslant j$,称 S_j 为 S_i 的后续状态,记作 $S_i\leqslant S_j$.

顺序进程的执行迹的状态随着其中每条语句的执行不断更新,其中 $T::a$ 表示向 T 中扩充项 a;$\sigma\{m\mid\to n\}$表示向 σ 中增加由变量 m 到比特串 n 的映射.$([s;P']P[i],S)\rightsquigarrow([P']P[i],S')$表示在执行 $P[i]$的语句 s 后,进程执行迹的状态 S 将变为 S',$P[i]$将变为 P'.当$[\mathrm{send}\ M:T;P']P[i]$被执行时,$T$ 将被扩展$\{\mathrm{send}(M\sigma)\}P[i]$而 σ 保持不变;当$[\mathrm{receive}\ m:T;P']P[i]$被执行时,由于敌手对通信网络具有完全控制能力,进程 $P[i]$将进入阻塞状态.当某一时刻敌手向 $P[i]$输入类型为 T 的比特串 n 时,T 被扩展$\{\mathrm{receive}(n)\}P[i]$,$\sigma$ 将被扩展$\{m[i]\mid\to n\}$,$P[i]$将变为 P'.需要注意的是,敌手输入的消息 n 可能由敌手生成,也可能由敌手通过对其接收到的消息进行概率任意多项式时间计算得到.

对于并行进程而言,其执行迹为其各并行子进程执行迹的严格顺序合并迹的集合.

定义 2(严格顺序合并迹) 当执行迹 $T3=[S_{31},\cdots,S_{3n},\cdots]$满足以下条件时,则称其为 $T1=[S_{11},\cdots,S_{1n},\cdots]$和 $T2=[S_{21},\cdots,S_{2n},\cdots]$的严格顺序合并迹.规则如下:

对于任意的 i,k,存在 p,q,使得 $S_{3p}=S_{1i}$和 $S_{3q}=S_{2k}$成立;

对于任意的 j,存在 p,q,使得 $S_{3j}=S_{1p}$或 $S_{3j}=S_{2q}$成立;

对于任意的 $j\leqslant k$,存在 $p\leqslant q$,且 $S_{3p}=S_{1j}$和 $S_{3q}=S_{1k}$成立;

对于任意的 $j\leqslant k$,存在 $p\leqslant q$,且 $S_{3p}=S_{2j}$和 $S_{3q}=S_{2k}$成立.

2.3 加密算法

为了给正向推理奠定可靠的密码学基础,首先对使用协议演算准确描述安全协议中使用的加密算法及其安全性.加密算法主要包括两类:公钥加密算法和对称加密算法.

定义 3(公钥加密算法) 假设 T_r、T_{pk}、T_{sk}、T_m、T_c

和 T_r' 分别为公钥加密算法中随机密钥种子、公钥、私钥、消息明文、消息密文和随机加密种子对应的类型,公钥加密算法 AE = (akgen, aenc, adec)由如下算法定义:

(1)密钥生成算法 akgen: $T_r \rightarrow (T_{pk}, T_{sk})$,由两个子算法 pkgen: $T_r \rightarrow T_{pk}$ 和 skgen: $T_r \rightarrow T_{sk}$ 构成.在输入随机密钥种子 r 后,akgen 输出一个公私钥对 (p_k, s_k);

(2)加密算法 aenc: $T_m \times T_{pk} \times T_r' \rightarrow T_c$,对给定的消息明文 m、对称加密密钥 p_k 和随机加密种子 r',aenc 输出密文 c;

(3)解密算法 adec: $T_c \times T_{sk} \rightarrow T_m \cup \{\perp\}$,对于给定的消息密文 c 和解密密钥 s_k,如果 c 是关于 s_k 的有效密文,adec 输出对应的明文 m,否则输出 $\perp$,表示解密错误.

公钥加密算法的安全性通常表示为安全目标(GOAL)和攻击模型(ATK)的组合.安全目标主要有四种(缩写为大写英文字母):语义安全性(Semantic Security,SS),不可区分性(INDistinguishability,IND)和非延展性(Non-Malleability,NM).语义安全性和不可区分性则表示敌手不能获取密文对应明文任何新的信息;非延展性表示:给定一个挑战密文 y,敌手不能生成新的密文 y',其中 y' 的明文 x' 和 y 的明文 x 之间存在联系.攻击模型包括:选择明文攻击(Chosen Plaintext Attack,CPA),选择密文攻击(Chosen Ciphertext Attack 1,CCA1)和自适应选择密文攻击(Chosen Ciphertext Attack 2,CCA2).选择明文攻击下,敌手可以选择一些消息的明文,得到对应的密文;选择密文攻击下,敌手除了能进行选择明文攻击外,还能在得到挑战密文前选择并得到不同密文对应的明文;自适应选择密文攻击下,敌手除了能进行选择明文攻击外,还能得到除挑战密文外的其他密文对应的明文.将安全目标与攻击模型自由组合,得到公钥加密算法的九种安全属性:SS-CPA, SS-CCA1, SS-CCA2, IND-CPA, IND-CCA1, IND-CCA2, NM-CPA, NM-CCA1, NM-CCA2.文献[9]证明了语义安全性与不可区分性在任意攻击模型下是等价的,故只对语义安全性建模.

定义 4(对称加密算法) 假设 T_r, T_k, T_m, T_c 和 T_r' 分别为对称加密算法中随机密钥种子,加密密钥,消息明文,消息密文和随机加密种子对应的类型,对称加密算法 SE = (kgen, enc, dec)由以下算法定义:

(1)密钥生成算法 kgen: $T_r \rightarrow T_k$,根据随机密钥种子 r 计算对称加密密钥 k;

(2)加密算法 enc: $T_m \times T_k \times T_r' \rightarrow T_c$,根据消息明文 m,对称加密密钥 k 和随机加密种子 r',计算密文 c;

(3)解密算法 dec: $T_c \times T_k \rightarrow T_m \cup \{\perp\}$,根据消息密文 c 和解密密钥 k,判断 c 是否是关于 k 的有效密文,是否计算对应的明文 m,否则输出 $\perp$,表示解密错误.

对称加密算法的安全性与公钥加密算法的安全性类似.由于对称加密密钥一般不公开,非延展性变为完整性(INTegrity, INT).完整性主要包含明文完整性(PlainTeXT, PTXT)和密文完整性(CipherTeXT, CTXT).明文完整性表示敌手不能破坏密文内明文的完整性,密文完整性表示敌手不能破坏密文的完整性.通常情况下,完整性只与自适应选择密文攻击组合.对称加密算法公有八种安全属性:INT-PTXT, INT-CTXT, SS-CPA, SS-CCA1, SS-CCA2, IND-CPA, IND-CCA1, IND-CCA2.

验证加密算法 SE = (kgen, enc, dec)是否满足 GOAL 安全性的实验可以建模为进程:

pExp(SE, η, GOAL)::= new r; k:= kgen(r).
(! $j \leqslant n$1pEncrypt | ! $k \leqslant n$2pDecrypt | pChallenge | pTestGOAL).

pEncrypt::= receive m_i: T_m; new r': T_r'; c_i:= enc(m_i, k, r'); send c_i.

pDecrypt::= receive c_i; m_i:= dec(c_i, s_k); send m_i.

pChallenge::= new m: T_m; new r': T_r'; c:= enc(m, k, r'); send c.

pTestGOAL 的行为与需验证的具体目标相关:若 GOAL 为 SS,用 pTestSS 替代 pTest;若 GOAL 为 NM,用 pTestNM 替代 pTest.

pTestSS::= receive(m': T_m, R: $T_m \times T_m \rightarrow$ bool); new m'': T_m; if $R(m', m)$ = true then send true else send false.

pTestINT::= receive (c': T_c); m' = dec(c', k); if $m' \neq \perp$ then send true; else send false.

对称加密算法验证实验中敌手能力和安全属性的建模与公钥加密算法类似.

3 形式逻辑

从加密算法安全性验证实验的进程模型可以看出,给定一个具有 GOAL-ATK 安全性的公钥(或对称)加密算法 AE(或 SE),其安全性验证实验 pExp(AE, η, GOAL)的执行迹将具有以下特点:如果 pExp 的所有执行迹满足 ATK 攻击模型,那么敌手攻击成功的概率(pTestGOAL 输出 true)可忽略.从本质上看,pExp 和 ATK 模型都描述了对敌手资源获取能力的限制(密钥,明文,预言机等).如果任意进程 P 也满足这些限制,那么敌手攻击成功的概率必定也是可忽略的.

为了形式化描述和度量任意进程的执行是否满足上述的敌手限制能力,本节设计了一个形式逻辑,作为对任意进程的行为和执行状态进行断言.利用这些断言,能在安全协议环境下准确描述加密算法的安全性.

3.1 逻辑语法

进程通信和计算行为用行为公式描述,语法规则如下:

$a, a_1, a_2 ::= \mathrm{Recvd}(P, i, m) \mid \mathrm{Sent}(P, i, m) \mid \mathrm{Gen}(P, i, m) \mid \mathrm{Called}(P, i, f, m, r)$;

$\varphi, \theta ::= t_1 = t_2 \mid t_1 \neq t_2 \mid \mathrm{Ordered}(a_1, a_2) \mid \mathrm{NotLeaked}(P, m) \mid \mathrm{Contain}(m, n) \mid \mathrm{EncryptedIn}(c, m) \mid \mathrm{KeyIn}(c, k) \mid \mathrm{Fresh}(P, i, m) \mid \mathrm{FirstSent}(P, i, m, n) \mid s\mathrm{Ordered}(P, a_1, a_2) \mid \mathrm{Source1}(Q, c, m, k, r') \mid \mathrm{Source2}(Q, c, m, p_k, r') \mid \varphi \vee \theta \mid \varphi \wedge \theta \mid \neg \varphi \mid \varphi \rightarrow \theta \mid \exists x.\varphi$.

事件公式 a 表示进程中的语句 a 已被执行. $\mathrm{Recvd}(P, i, m)$ 表示进程 $P[i]$ 接收了消息 m; $\mathrm{Sent}(P, i, m)$ 表示进程 $P[i]$ 发送了消息 m; $\mathrm{Gen}(P, i, m)$ 表示进程 $P[i]$ 产生了消息 m; $\mathrm{Called}(P, i, f, m, r)$ 表示进程 $P[i]$ 以 m 为参数调用函数 f, 且 f 返回结果 r.

谓词公式 φ 对进程的执行状态进行断言. 公式 $\mathrm{Ordered}(a_1, a_2)$ 表示行为 a_1 在行为 a_2 之前发生; $\mathrm{NotLeaked}(P, i, m)$ 表示进程 $P[i]$ 未发送会泄露 m 信息的消息. 若对于所有的进程 P, $\mathrm{NotLeaked}(P, i, m)$ 皆成立, 则记为 $\mathrm{NotLeaked}(i, m)$. $\mathrm{EncryptedIn}(c, m)$ 表示 m 曾经被加密为密文 c; $\mathrm{KeyIn}(c, k)$ 表示 k 曾经被作为密钥生成密文 c; $\mathrm{Contain}(m, n)$ 表示 m 中可能包含 n 的信息; $\mathrm{Fresh}(P, i, m)$ 表示 m 是新鲜的, 即 $P[i]$ 未将生成的消息 m 以任何形式发送出去; $\mathrm{FirstSent}(P, i, m, n)$ 表示 $P[i]$ 第一次将 m 通过 n 发送出去; $\mathrm{Source1}(Q, c, m, k, r')$ 表示明文 m 和加密随机数 r' 仅以密文 $c = \mathrm{enc}(m, k, r')$ 的形式被 $Q[i]$ 发送出去; $\mathrm{Source2}(Q, c, m, p_k, r')$ 表示明文 m 和加密随机数 r' 仅以密文 $c = \mathrm{aenc}(m, p_k, r')$ 的形式被 $Q[i]$ 发送出去; 公式 $\varphi \vee \theta$, $\varphi \wedge \theta$, $\neg \varphi$, $\varphi \rightarrow \theta$ 和 $\exists x.\varphi$ 为一阶逻辑连接符.

3.2 逻辑语义

逻辑公式的语义建立在进程执行迹的各个状态上. 对于协议 Q 的状态 $S = \langle T, \sigma \rangle$, 语义关系 $Q, S \models \varphi$ 表示公式在协议 Q 的状态 S 下成立. 若对于协议 Q 的所有状态 S, 都有 $Q, S \models \varphi$ 成立, 则记作 $Q \models \varphi$. 各公式的语义如下所示:

$Q, \langle T, \sigma \rangle \models \varphi \vee \theta$, 若 $Q, \langle T, \sigma \rangle \models \varphi$ 或 $Q, \langle T, \sigma \rangle \models \theta$.

$Q, \langle T, \sigma \rangle \models \varphi \wedge \theta$, 若 $Q, \langle T, \sigma \rangle \models \varphi$ 且 $Q, \langle T, \sigma \rangle \models \theta$.

$Q, \langle T, \sigma \rangle \models \neg \varphi$, 若 $Q, \langle T, \sigma \rangle \not\models \theta$.

$Q, \langle T, \sigma \rangle \models \varphi \rightarrow \theta$, 若 $Q, \langle T, \sigma \rangle \models \neg \varphi \vee \theta$.

$Q, \langle T, \sigma \rangle \models \exists x.\varphi$, 若存在 d, $Q, \langle T, \sigma \rangle \models \varphi\{x \mid \rightarrow d\}$ 成立, $\varphi\{x \mid \rightarrow d\}$ 表示将 φ 中的 x 用 d 替换.

$Q, \langle T, \sigma \rangle \models \mathrm{Recvd}(P, i, m)$, 若 T 中存在 $\{\mathrm{receive}(m)\}_{P[i]}$ 项.

$Q, \langle T, \sigma \rangle \models \mathrm{Sent}(P, i, m)$, 若 T 中存在 $\{\mathrm{send}(m)\}_{P[i]}$ 项.

$Q, \langle T, \sigma \rangle \models \mathrm{Gen}(P, i, m)$, 若 T 中存在 $\{\mathrm{Gen}(m)\}_{P[i]}$ 项.

$Q, \langle T, \sigma \rangle \models \mathrm{Called}(P, i, f, m, r)$, 若 T 中存在 $\{\mathrm{Called}(f, m, r)\}_{P[i]}$ 项.

$Q, \langle T, \sigma \rangle \models \mathrm{Ordered}(a_1, a_2)$, 若 $Q, \langle T, \sigma \rangle \models a_2 \wedge a_1$ 不存在状态 S_1, S_2, $S_1 \leqslant S_2 \leqslant \langle T, \sigma \rangle$, 且 $Q, S_1 \models a_2$, $Q, S_2 \models a_1$.

$Q, \langle T, \sigma \rangle \models \mathrm{Called}(P, i, f, (n_1, \cdots n_{i-1}, n, n_{i+1}, \cdots, n_k), m)$ 成立.

$Q, \langle T, \sigma \rangle \models \mathrm{EncryptedIn}(c, m)$, 如果存在 $k, r, Q, \langle T, \sigma \rangle \models \mathrm{Called}(P, i, \mathrm{enc}, (m, k, r), c)$ 或 $Q, \langle T, \sigma \rangle \models \mathrm{Called}(P, i, \mathrm{aenc}, (m, k, r), c)$ 成立.

$Q, \langle T, \sigma \rangle \models s\mathrm{Ordered}(P, a_1, a_2)$, 若存在替换 θ_1, θ_2 和进程 P_1, P_2, P_3, $P ::= P_1; a_1\theta_1; P_2; a_2\theta_2; P_3$. 其中 θ_1, θ_2 用于对 a_1 和 a_2 中的自由变量重命名.

3.3 证明系统

证明系统包含一阶逻辑的公理系统(与一阶逻辑中公理相同, 公理 CON1 表示如果进程 $P[i]$ 执行了 $n := f(m_1, \cdots m_k)$, 那么 n 中可能包含 $m_1, \cdots, m_k$ 的信息; CON2 表示如果 $P[i]$ 执行了 $c := \mathrm{enc}(m, k, r)$ 或者 $c := \mathrm{aenc}(m, k, r)$, 那么公式 $\mathrm{EncryptedIn}(c, m)$, $\mathrm{KeyIn}(c, k)$ 和 $\mathrm{EncryptedIn}(c, r)$ 成立; CON3 表示密文中包含明文、密钥以及加密随机数的信息; GEN1 表示两个不同的进程不会产生相同的随机数; NLK1 表示如果进程 $P[i]$ 未发送或者仅将 m 作为密钥加密其他消息发送出去, $P[i]$ 并未泄露 m 的信息; FS1 表示如果 $P[i]$ 任何发送 m 的信息的动作都在 $\mathrm{Sent}(P, i, n)$ 之后, 那么如果 $P[i]$ 第一次发送 m 是在 n 中完成的; FS2 表示如果 $P[i]$ 第一次发送 m 在 n 中完成, 且存在进程 $Q[j]$ 执行了使用 m 的信息的动作 $a(Q, j, n')$, 那么 $a(Q, j, n')$ 一定发生在 $P[i]$ 发送 n 之后; SORD 表示语法上存在先后顺序的语句, 其对应的行为也存在先后顺序; ORD 表示行为动作具有保持性; PRE 表示在状态 S_1 和 S_2 有先后关系的情况下, 如果在状态 S_2 下 a_1, a_2 有顺序执行关系, 且在状态 S_1 下 a_2 成立, 那么在状态 S_1 下 a_1, a_2 仍然保持顺序执行关系; PART 表示顺序执行的行为一定都被执行过.

3.4 CPCL 的缺陷

文献[5]尝试使用以下 ENC 公理描述对称加密算法的 IND-CPA 安全性:

$\mathrm{PSource}(\underline{X}, b, m, k) \wedge \mathrm{Honest}(\underline{X}, Y) \wedge \mathrm{SharedKey}(\underline{X}, \underline{Y}, k) \wedge (\mathrm{Decrypts}(\underline{Y}, m, k) \wedge \mathrm{Contains}(m, m') \wedge \mathrm{Send}(\underline{Y}, m'') \rightarrow \neg\, \mathrm{Contain}(m'', m')) \wedge (\mathrm{Decrypts}(\underline{X}, m, k) \wedge \mathrm{Contains}(m, m') \wedge \mathrm{Send}(\underline{X}, m'') \rightarrow \neg\, \mathrm{Contain}(m'', m')) \wedge Z \neq X \wedge Z \neq \underline{Y} \rightarrow \mathrm{Indist}(Z, b)$.

其中 X,Y 表示协议主体,$\underline{X}$ 和 $\underline{Y}$ 分别表示 X 和 Y 执行的线程,Decrypts($\underline{Y}$,m,k)表示线程 $\underline{Y}$ 使用 k 对密文解密得到明文 m,其它谓词的语义与本文相关谓词类似.

ENC 公理的目标是表示:当 k 是线程 $\underline{X}$ 和 $\underline{Y}$ 间的共享密钥的情况时,如果线程 $\underline{X}$ 选择消息 m 和比特 b,并且这些线程都遵循 IND-CPA 规则,那么对于任意不等于 X 和 Y 的主体 Z,都不能区分 b 和其他的随机比特.

然而在对线程满足 IND-CPA 规则建模时,ENC 中包含错误,公式(Decrypts($\underline{Y}$,m,k) ∧ Contains(m,m') ∧ Send($\underline{Y}$,m'')→ ¬ Contain(m'',m'))实质上表示的是 $\underline{Y}$ 未将挑战明文 m 中的信息发送出去,而非 CPA 模型中规定的"敌手不能以任何形式调用解密预言机".本质上,ENC 公理描述的是对称加密算法的 IND-CCA2 安全性.

对 ENC 公理的修正比较简单,只需将两条描述攻击模型的蕴含公式变为(Decrypts($\underline{Y}$,n,k) ∧ Contains(n,m') ∧ Send($\underline{Y}$,m'')→ ¬ Contain(m'',m'))(Decrypts($\underline{X}$,n,k) ∧ Contains(n,m') ∧ Send($\underline{X}$,m'')→ ¬ Contain(m'',m')).

文献[10]中使用 PCL 对 NSL 协议的安全性进行了证明,证明过程依赖于公理 AR1[(m)]$_{\underline{X}}$ ∃ $\underline{Y}$. Sent($\underline{Y}$,m).该公理表示,如果某个线程接收到消息 m,必定存在线程 $\underline{Y}$,且 $\underline{Y}$ 发送了消息 m.该公理是不可靠的,因为对于公钥加密机制而言,由于公钥是公开的,敌手能生成并加密任何消息项.因此如果某个主体接收到由某公钥加密的密文,并不能断言某协议主体曾经发送过该密文.

3.5　加密算法的安全性

从加密算法安全性验证实验的过程可以看出,pExp 主要限制了敌手以下两个资源的访问能力:(1)相关密钥未泄露;(2)仅能通过挑战密文访问挑战明文和加密随机种子.结合敌手攻击模型,可以将加密算法的安全属性表示为逻辑蕴含公式"密钥未泄露 ∧ 仅能通过挑战密文访问挑战明文和加密随机种子 ∧ 协议遵循敌手攻击模型→ ¬(敌手攻击成功)".利用行为公式和谓词公式,可以将上述蕴含公式中的各项描述为:

(1)密钥未泄露:NotLeaked(k)

(2)仅能通过挑战密文访问挑战明文和加密随机种子:

Gen(Q,i,m) ∧ Gen(Q,i,r') ∧ Source1(Q,c,m,k,r')

Gen(Q,i,m) ∧ Gen(Q,i,r') ∧ Source2(Q,c,m,p_k,r')

(3)协议遵循敌手攻击模型:

CPA(c_2,dec,k) ≡ ¬ ∃ Q_1Q_2ijumc'. (ActionsInOrder(Called(Q_1,i,adec,(c',k),m),Sent(Q_1,i,u),Recvd(Q_2,j,c_2)) ∧ Contain(u,m))

选择明文攻击谓词 CPA(c_2,dec,k)表示进程所有执行迹满足如下条件:在 $Q_2[j]$接收到攻击密文 c_2 之前,不存在进程 $Q_1[i]$对密文 c'解密,并将明文的部分信息发送出去.

CCA1(c,c_2,adec,k) ≡ ¬ ∃ $Q_1Q_2Q_3c'$ m' n' n'' ijk. (FirstSent(Q_2,j,c,n'') ∧ m' = dec(c',k) ∧ FirstSent(Q_1,i,m',n') ∧ ActionsInOrder(Sent(Q_2,j,n''),Called(Q_1,i,dec,(c',k),m'),Sent(Q_1,i,n'),Recvd(Q_3,k,c_2))).

选择密文攻击谓词 CCA1(c,c_2,dec,k)表示进程所有执行迹满足如下条件:在 $Q_2[j]$发送挑战密文 c 后、$Q_3[k]$收到攻击密文 c_2前,不存在进程 $Q_1[i]$对密文 c'进行解密,并将解密的部分消息发送出去.

CCA2(c,c_2,dec,k) ≡ ¬ ∃ Q_1Q_2ijuu. (ActionsInOrder(Called(Q_1,i,adec,(c,k),m),Sent(Q_1,i,u),Recvd(Q_2,j,c_2)) ∧ Contain(u,m)).

自适应选择密文攻击谓词 CCA2(c,c_2,dec,k)表示进程所有的执行迹满足如下条件:在 $Q_2[j]$收到攻击密文 c_2之前,不存在进程 $Q_1[i]$对挑战密文 c 解密并将明文的部分信息发送出去.

INT(m,c,c_2,k) ≡ ¬ ∃ Q_1 Q_2 $ijuu$. (ActionsInOrder(Called(Q_1,i,aenc,(m,k,r),c),Sent(Q_1,i,u),Recvd(Q_2,j,c_2)) ∧ Contain(u,m)).

完整性 INT(m,c,c_2,k)表示进程所有的执行迹均满足如下条件:在 $Q_2[j]$接收到攻击密文 c_2前,不存在进程 $Q_1[i]$对挑战明文 m 加密,并将加密结果 c 并发送出去.

(4)敌手攻击成功

对语义安全性攻击成功,表示敌手生成明文 m',且与挑战明文 m 之间满足二元函数 R:$T_1 \times T_2$→bool,使用公式 PRelated(m',m) ≡ ∃ R. R(m',m) = true 表示.

对非延展性攻击成功,表示敌手生成明文 c',且与挑战密文 c 的明文之间满足二元函数 R:$T_1 \times T_2$→bool,使用公式 CRelated(c',c,s_k) ≡ ∃ $mm'R$. m' = adec(c',s_k) ∧ $m' \neq \perp$ ∧ m = adec(c,s_k) ∧ $m \neq \perp$ ∧ R(m',m) = true 表示.

语义安全性与非延展性要求敌手输出关系 R,而在安全协议中没有进程以函数为输入参数.为了能在安全协议中应用上述两个性质,这里规定 $R \in$ Fun,其中 Fun 表示当前进程中所有函数的集合.

4　测试与验证

本节证明了 Needham-Schroeder-Lowe(NSL)公钥协议的安全性,对逻辑的表达和证明能力进行了测试和验证.

NSL 协议共包含三条消息,四个步骤:

步骤 1 $A\rightarrow B:\{A,N_a\}_{pkB}$

步骤 2 $B\rightarrow A:\{N_a,N_b,B\}_{pkA}$

步骤 3 $A\rightarrow B:\{N_b\}_{pkB}$

在步骤 1 中主体 A 生成新鲜数 N_a,并将(A,N_a)用 B 的公钥pkB 加密发送出去;步骤 2 中主体 B 将接收到的消息解密,生成新鲜数 N_b,并将(N_a,N_b,B)用 A 的公钥 p_{kA}加密发送出去;步骤 3 中主体 A 将接收到的消息解密,若明文第一项是为 N_a,则将 N_b 使用p_{kB}加密发送出去,并完成对响应方的认证; 最后一步中主体 B 将接收到的消息解密,若解密结果等于 N_b,则完成对初始化方的认证.

NSL 协议通过协议演算可以建模为:

$Q_{\mathrm{NSL}}::=Q_{\mathrm{Init}};(!^{i\leqslant n}Q_A|!^{i\leqslant n}Q_B)$.

$Q_{\mathrm{Init}}::=$ new $r_a,r_b,r_c:T_r;p_{kA}:=\mathrm{pkgen}(r_a);s_{kA}:=\mathrm{skgen}(r_a);p_{kB}:=\mathrm{pkgen}(r_b);s_{kB}:=\mathrm{skgen}(r_b);p_{kC}:=\mathrm{pkgen}(r_c);s_{kC}:=\mathrm{skgen}(r_c)$;send $(p_{kA},p_{kB},p_{kC},s_{kC})$.

$Q_A::=$ receive $X:T_h$;new $N_a:T$;new $r_1':T_r'$;if $X=A$ then $p_{kX}:=p_{kA}$ else if $X=B$ then $p_{kX}:=p_{kB}$ else $p_{kX}:=p_{kC}$;$c_1:=\mathrm{aenc}((A,N_a),p_{kX},r_1')$;send c_1;receive $c_2:T_c$;$(=N_a,N_x,=X):=\mathrm{adec}(c_2,s_{kA})$;new $r_2':T_r'$;$c_3:=\mathrm{aenc}((N_x),p_{kX},r_1')$;accept$A(A,X,N_a,N_x)$;send c_3.

$Q_B::=$ receive $c_1':T_c$;$(N_y,Y:T_h):=\mathrm{adec}(c_1',s_{kB})$;new $N_b:T_n$;new $r_3':T_r'$;if $Y=A$ then $p_{kY}:=p_{kA}$ else if $Y=B$ then $p_{kY}:=p_{kB}$ else $p_{kY}:=p_{kC}$;$c_2':=\mathrm{aenc}((N_y,N_b,B),p_{kY},r_3')$;send c_2';receive c_3';$(=N_b):=\mathrm{adec}(c_3',s_{kB})$;accept$B(Y,B,N_y,N_b)$.

Q_{NSL}首先执行初始化子进程 Q_{Init}. Q_{Init}首先生成密钥种子 r_a, r_b, r_c和公私钥对(p_{kA},s_{kA}),(p_{kB},s_{kB}),(p_{kC},s_{kC}),将 p_{kA} p_{kB} p_{kC}以及 s_{kC}发送出去,然后并行执行进程 Q_A和 Q_B.

Q_A对主体 A 执行的 NSL 协议发起方程序进行建模. Q_A首先从网络中接收协议响应方 ID X (模拟敌手能初始化任何协议会话),从 T_n中生成随机新鲜数 N_a,查询并使用 X 的公钥p_{kX}对(A,N_a)进行加密,并将密文发送出去;接着 Q_A进入阻塞状态,等待网络上输入的消息.当从网络收到消息 c_2后,首先使用 A 的私钥s_{kA}对其解密,如果解密成功,且解密得到的明文的第一部分和第三部分分别等于 N_a和 X,则将明文的第二部分保存到变量 N_x中,接着使用 p_{kX}对 N_x进行加密,并调用函数 accept$A(A,X,N_a,N_x)$ (记此时 Q_{NSL}的状态为 S_A),用于断言 A 已经以 NSL 发起方的身份完成协议,且协议会话使用的新鲜数为 N_a和N_x.

Q_B对主体 B 执行 NSL 协议响应方程序进行建模. Q_B首先从网络中接收密文 c_1',使用 B 的私钥 s_{kB}对其解密,且将解密明文解析为新鲜数 N_y和协议发起方 ID Y. 接着 Q_B从 T_n中生成新的随机新鲜数 N_b,查询和使用 Y 的公钥 p_{kY}对(N_y,N_b,B)加密,将得到的密文 c_2'发送出去,然后进入阻塞状态,等待从网络输入的消息;当从网络收到消息 c_3'后,使用 s_{kB}对其解密,若解密得到的明文等于 N_b, Q_B调用 accept$A(A,B,N_y,N_b)$,用于断言 B 已经以 NSL 响应方的身份完成协议,且协议认证会话使用的新鲜数为 N_y和N_b.

NSL 协议的安全目标为双向认证性,采用匹配会话对其建模,即:如果发起方 A 认为其当前会话的响应方为B,且协议中使用的新鲜数为 N_a, N_b,那么 A 和 B 确认对方身份前发送每条消息都被对方按顺序收到;反之亦然.用逻辑公式表示为 $Q_{\mathrm{NSL}}|=\Gamma_1\wedge\Gamma_2$,其中 Γ_1和 Γ_2分别表示 A 和B 完成对对方的认证.

$\Gamma_1=\mathrm{Called}(Q_A,\mathrm{accept}A,i,(A,B,m,n),\perp)\rightarrow\exists j,c_1c_2r_1'r_2'r_ar_b.\mathrm{Ordered}(\mathrm{Sent}(Q_A,i,c_1),\mathrm{Recvd}(Q_B,j,c_1),\mathrm{Sent}(Q_B,j,c_2),\mathrm{Recvd}(Q_A,i,c_2))\wedge c_1=\mathrm{aenc}((A,m),\mathrm{pkgen}(r_b),r_1')\wedge c_2=\mathrm{aenc}((m,n,B),\mathrm{pkgen}(r_a),r_2')\Gamma_2=\mathrm{Called}(Q_B,\mathrm{accept}B,j,(A,B,m,n),\perp)\rightarrow\exists i,c_1c_2c_3r_1'r_2'r_3'r_ar_b.\mathrm{Ordered}(\mathrm{Sent}(Q_A,i,c_1),\mathrm{Recvd}(Q_B,j,c_1),\mathrm{Sent}(Q_B,j,c_2),\mathrm{Recvd}(Q_A,i,c_2),\mathrm{Sent}(Q_A,i,c_3),\mathrm{Recvd}(Q_B,j,c_3))\wedge c_1=\mathrm{aenc}((A,m),\mathrm{pkgen}(r_b),r_1')\wedge c_2=\mathrm{aenc}((m,n,B),\mathrm{pkgen}(r_a),r_2')\wedge c_3=\mathrm{aenc}((n),\mathrm{pkgen}(r_b),r_3')$.

公式 Γ_1的证明主要包含七个步骤.前三个步骤主要证明:如果 Called$(Q_A,\mathrm{accept}A,i,(A,B,m,n),\perp)$成立,那么进程 Q_A必定已经随机生成 m,计算并发送 $c_1=\{A,m\}^{r1'}_{pkB}$,接收并验证 c_2的明文为(m,n,B).由于 Q_A仅在$\{A,m\}^{r1'}_{pkB}$中发送 m,而接收到的协议第二条消息 c_2的明文为(m,n,B),意味着 c_1破坏了的非延展性,由推论 A-NM-CCA2-2 得到 Q_B必定接收并解密 c_1,且发送 c_2'.中间三个步骤主要证明:同样 Q_B仅 c_2'在中发送随机生成的随机数n,并且在 Q_A刚接收到 c_2而未对其解密时不存在其他任何进程对 c_2'解密,根据推理 A-NM-CCA2 – 1 得出 c_2'必定与 c_2 相等.最后一个步骤得出蕴含公式结论.

5 结论

本文提出了一个计算可靠的加密算法形式逻辑.该逻辑弥补了现有文献中各种逻辑方法的不足,能对常用加密算法的安全属性进行准确描述,并能应用于对 NSL 协议安全性的证明中.与现有形式化方法相比,本逻辑既具有计算模型下的可靠性,又具有符号模型下的易用性.

经过扩展,本逻辑将很容易有以下描述和推理能力:(1)对签名、散列、消息校验等密码学原语安全性的建模;(2)对不可否认性,机密性等安全属性的推理;(3)若逻辑系统建立在概率逻辑上,将具有定量推理能力.

参考文献

[1] 董玲,陈克非,来学嘉.密码协议分析的信任多集方法[J].软件学报,2009,20(11):3060 – 3076.
DONG L, CHEN KF, LAI XJ. Belief multiset formalism for cryptographic protocol analysis[J]. Journal of Software. 2009, 20(11):3060 – 3076. (in Chinese)

[2] 张畅,王亚弟,韩继红,等.一种改进的密码协议形式化模型[J].软件学报.2007,18(7):1746 – 1755.
ZHANG C, WANG YD, HAN JH, et al. An improved formal model of cryptographic protocol[J]. Journal of Software. 2007, 18(7):1746 – 1755. (in Chinese)

[3] Warinschi B. A computational analysis of the needham-schroeder-(lowe) protocol[J]. Computer Security, 2005, 13(3):565 – 591.

[4] Abadi M, Rogaway P. Reconciling two views of cryptography (The computational soundness of formal encryption) [A]. TCS'00: Proceedings of the International Conference IFIP on Theoretical Computer Science, Exploring New Frontiers of Theoretical Informatics[C]. London, UK: Springer-Verlag, 2000. 3 – 22.

[5] Blanchet B. A Computationally sound mechanized prover for security protocols[J]. IEEE Transactions on Dependable and Secure Computing, 2008, 5(4):193 – 207.

[6] Kramer S. Logical concepts in cryptography[J]. ACM SIGACT News, 2007, 38(4):65 – 66.

[7] Troina A, Aldini A, Gorrieri R. Towards a formal treatment of secrecy against computational adversaries[A]. 2nd IST/FET International Workshop on Global Computing[C]. Rovereto, Italy: Springer, 2005. 77 – 92.

[8] Datta A, Derek A, Mitchell J C , et al. Probabilistic polynomial-time semantics for a protocol security logic[A]. 32nd International Colloquium on Automata, Languages and Programming [C]. Lisbon, Portugal: Springer, 2005. 16 – 29.

[9] Cremers C. On the protocol composition logic PCL[A]. 2008 ACM Symposium on Information, Computer and Communications Security[C]. New York, USA: ACM, 2008. 66 – 76.

[10] Durgin N, Mitchell J, Pavlovic D. A compositional logic for proving security properties of protocols[J]. Journal of Computer Security, 2003, 11(4):677 – 721.

作者简介

唐朝京　男,1962 年生于江苏,国防科技大学教授,博士生导师,电子科学与工程学院院长,国家 863 – 711 专题专家,中国通信学会理事,湖南省通信学会副理事长,主要研究方向为通信网信息安全与对抗.

鲁智勇　男,1969 年生于河南,博士,高级工程师,主要研究方向为电子靶场武器系统对抗、网络安全与评估.

冯　超　男,1983 生于江苏盱眙,国防科技大学电子科学与工程学院博士,主要研究方向为通信网络安全.