软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2019,30(12):3730−3749 [doi: 10.13328/j.cnki.jos.005603]

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

一种网络间可信连接协议*

赖英旭, 刘 岩, 刘 静

(北京工业大学 信息学部 计算机学院,北京 100124)

通讯作者: 赖英旭, E-mail: laiyingxu@bjut.edu.cn

摘 要: 为了解决深化“互联网+先进制造业”进程中网络可信互连问题,引入了可信连接架构(trusted connect architecture,简称 TCA)技术.基于 TCA 技术思想,针对网络间可信认证需求,设计了一种支持网络间互连的可信连接协议(TCA-SNI).引入了网络间双向认证过程,给出了 TCA-SNI 协议的交互过程;使用扩展的 SVO 逻辑系统对协议进行逻辑推理,证明该协议是安全可靠的;使用 Dolev-Yao 攻击者模型对协议进行攻击测试,实验结果表明,协议的安全目标均已达成,证明该协议可以抵御真实网络中的攻击.

关键词: 可信连接架构;可信计算;可信连接协议;SVO;AVISPA

中图法分类号: TP393

中文引用格式: 赖英旭,刘岩,刘静.一种网络间可信连接协议.软件学报,2019,30(12):3730−3749. http://www.jos.org.cn/1000-9825/5603.htm

英文引用格式: Lai YX, Liu Y, Liu J. Trusted connection protocol between networks. Ruan Jian Xue Bao/Journal of Software, 2019,30(12):3730−3749 (in Chinese). http://www.jos.org.cn/1000-9825/5603.htm

Trusted Connection Protocol Between Networks

LAI Ying-Xu, LIU Yan, LIU Jing

(College of Computer Science, Faculty of Information Technology, Beijing University of Technology, Beijing 100124, China)

Abstract: Trusted connect architecture (TCA) technology was introduced to solve the problem of trusted connect between networks in “Pushing Forward the Internet plus Advanced Manufacturing” plan. Based on the idea of TCA technology, this study proposed a trusted connection protocol (TCA-SNI) for trusted authentication and evaluation between networks. The two-way authentication process is introduced and the interaction of TCA-SNI is given. The extended SVO logic system is used to infer the protocol logicalness, which proves that the protocol is safe and reliable. The protocol is detected using the Dolev-Yao model. Experimental results show that the proposed protocol has achieved the security goal, and can withstand attacks in the real network.

Key words: trusted connect architecture (TCA); trust computing; trusted connection protocol; SVO; AVISPA

随着网络技术和“互联网+”应用的快速发展,国民经济和社会发展越来越依赖基础网络和信息系统[1].国务院关于深化“互联网+先进制造业”发展工业互联网的指导意见中明确指出:要完善生态体系,整合企业资源,鼓励企业跨领域、跨产业链紧密协作,协同发展.在此过程中,网络的可信互连是企业间协同生产的基础,是保障企业间共享资源、紧密协作的重点.网络本身具有脆弱性,为保护网络中信息系统的安全,由防火墙、入侵检测、病毒防范系统等组成的传统信息安全系统得到广泛运用.但这些传统的信息安全系统以抵挡外部入侵为主要手段,很难从源头上保障信息系统的安全,治标不治本[2,3].

可信计算组织(trusted computing group,简称 TCG)提出了可信网络连接(trusted network connection,简称

* 基金项目: 青海省自然科学基金(2017-ZJ-912); 北京市自然科学基金(4162006)

Foundation item: Natural Science Foundation of Qinghai Province (2017-ZJ-912); Beijing Municipal Natural Science Foundation (4162006)

收稿时间: 2017-12-03; 修改时间: 2018-03-16; 采用时间: 2018-05-16

TNC)规范[4,5],在可信计算基础上,依据 TPM 可信度量和报告机制构建分层的可信网络框架.TNC 对终端进行用户认证,包括用户身份的检查以及平台认证和平台完整性检查.TNC 采用了 VPN 和 IEEE 802.1x 进行认证并建立通道,将终端的可信状态延续到网络中,极大提高了网络安全性.然而,TNC 只能针对终端进行了单向可信认证,还存在无法抵御伪装及重放攻击等问题[6−11].

针对 TNC 的不足,我国制定了《信息安全技术 可信计算规范 可信连接架构》标准[12],该标准采用了一种三元三层、对等、集中管理的可信连接架构(trusted connect architecture,简称 TCA)技术.为了集中管理访问请求者和访问控制者,引入了策略管理器作为可信第三方,实现了对访问请求者和访问控制者的双向可信认证和评估.然而,面对深化“互联网+先进制造业”进程中网络互连的安全需求,TCA 技术仅提供了终端接入网络的可信连接方案,无法适用于网络间的可信认证.

本文对 TCA 进行扩展,提出了一种支持网络间互连的可信连接协议 TCA-SNI(TCA-support network interconnection).通过引入网络间双向四元认证过程,TCA-SNI 具备了网络相互评估的机制,增强了 TCA 在网络间的可信认证能力.

1 可信连接架构

TCA 是我国自主创新的一种三元三层的可信连接架构,在终端连接到网络时进行双向身份鉴别和平台鉴别,实现终端到网络的可信连接,其架构如图 1 所示.

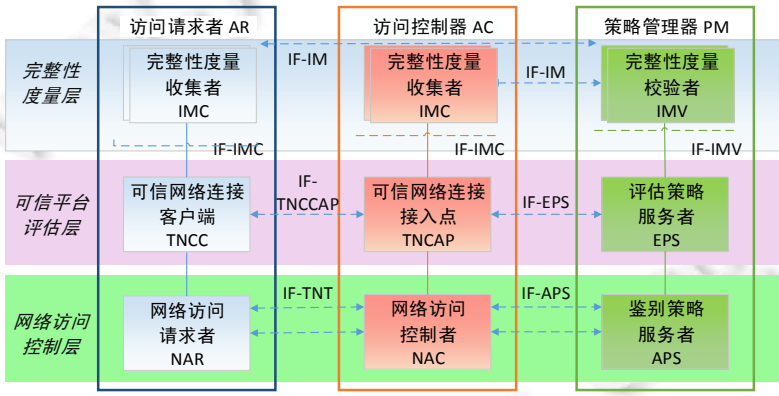

Fig.1 Trusted connect architecture

图 1 可信连接架构

TCA 采用的是三元架构,存在 3 个实体[12]:访问请求者(access requestor,简称 AR)、访问控制器(access controller,简称 AC)和策略管理器(policy manager,简称 PM).自上至下分为 3 个抽象层:完整性度量层(integrity measurement layer)、可信平台评估层(trusted platform evaluation layer)和网络访问层(network access layer).通过身份鉴别、平台鉴别进行访问控制,身份鉴别可完成身份合法性验证,平台鉴别可进行平台完整性评估,从而确保网络两端设备的可信.TCA 支持配置多种平台完整性评估策略,用户可针对需要评估的组件设置灵活或多层次的评估策略,根据最终的度量结果,给予不同的访问权限.

TCA 的具体步骤为[12]:

(1) 在建立可信连接前,TNCC(可信连接客户端)与 TNCAP(可信连接接入点)分别加载它们上层的各个 IMC(完整性度量收集者),而 EPS(评估策略服务者)加载它上端的各个 IMV(完整性度量校验者);

(2) NAR(网络访问请求者)向 NAC(网络访问控制者)发起网络访问请求;

(3) NAC 收到访问请求后,与 NAR 和 APS(鉴别策略服务者)执行用户身份鉴别协议,来实现 AR(访问请求者)和 AC(访问控制者)之间的双向用户身份鉴别.APS 在鉴别过程中充当可信第三方;

(4) 发起平台鉴别过程,NAR 向 TNCC 发送平台鉴别请求,NAC 向 TNCC 发送平台鉴别请求;

(5) 当 TNCAP 收到平台鉴别请求时,启动平台鉴别过程,与 TNCC 和 EPS 执行平台鉴别协议;

(6) 在平台鉴别过程中,TNCC 通过 IF-IMC 与其上端的各个 IMC 进行信息交互,TNCAP 通过 IF-IMC 与其上端的各个 IMC 进行信息交互;

(7) EPS 负责验证 AR 和 AC 的 PKI 证书,并通过 IF-IMV 调用它上端的各个 IMV 来校验评估 AR 和 AC 的平台完整性度量值.EPS 依据平台完整性评估策略生成 AR 和 AC 的平台完整性评估结果,最后将 PKI 证书验证结果和平台完整性评估结果发送给 TNCC 和 TNCAP;

(8) 当 AR 和 AC 的平台鉴别完成时,TNCC 和 TNCAP 分别依据 EPS 生成 AR 和 AC 的 PKI 证书验证结果和平台完整性评估结果生成访问策略,并分别发送给 NAR 和 NAC;

(9) NAR 依据它所生成的访问策略或从 TNCC 接收到的访问策略执行访问控制,NAC 依据它所生成的访问策略或从 TNCC 接收到的访问策略执行访问控制,从而实现可信连接.

由上述步骤可以看出,可信连接架构与传统的方法完全不同,可信连接架构为网络外的用户或终端规定了安全策略,只有符合策略的终端才可以接入到网络中,将可能导致恶意攻击的终端隔离到网络外,极大地减少了安全隐患.

2 基于 TCA 的网络间可信连接协议

本文基于TCA技术的思想,设计了一种支持网络间互连的可信连接协议TCA-SNI[13],协议模型如图2所示.

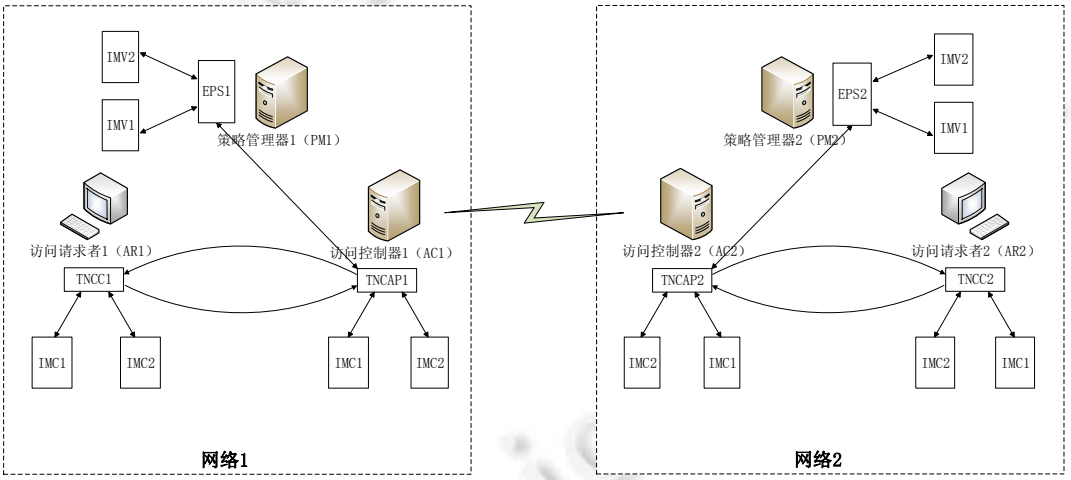

Fig.2 Trusted connect protocol topology between networks

图 2 网络间可信连接协议拓扑

图 2 中,每个网络中的策略管理器是进行可信认证的核心设备,受到高等级保护,仅能与本网中的访问控制器进行通信.访问请求者在发起接入网络请求时,处于不可信区域,仅能与访问控制器通信,在可信认证后,才能成功接入网络,与网络内除策略管理器外的其他设备通信.

2.1 设计思想

2.1.1 网络间双向四元可信认证

网络 1 向网络 2 发送连接请求时,网络 2 的策略管理器为可信第三方,根据网络 2 的安全策略评估网络 1 的访问控制器.当网络 2 向网络 1 发送连接请求时,网络 1 的策略管理器为可信第三方,根据网络 1 的安全策略评估网络 2 的访问控制器.此双向四元结构能够在双方网络安全策略不同的情况下进行可信连接,可使两个网络的终端在可信的环境下进行基础通信.若需要执行高级命令,则需继续进行深度可信认证[13].四元三层可信连

接架构如图 3 所示.

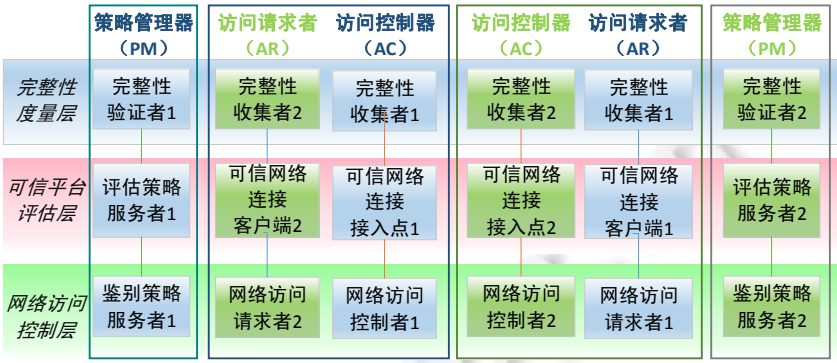

Fig.3 Trusted connect architecture between networks

图 3 网络间可信连接架构

2.1.2 多级可信认证

网络双方在网间可信连接过程中,不仅对身份信息及完整性信息进行可信评估,而且策略管理器还对对方网络的安全策略进行可信评估.双方网络的策略管理器协同工作,使网络双方构成可信通信环境.此时,网络内的终端可以与对方网络进行基础通信,完成基本操作.

由于网络双方安全策略的不同,网络 1 中的可信终端可能不符合网络 2 的安全策略,一旦该终端发送高级控制指令,网络 2 将会面临安全威胁,造成严重后果.而网络 1 中,只有部分终端需要对网络 2 发起高级控制指令,因此不需要对网络 1 中所有终端进行深度认证.对于需要向网络 2 发送高级命令的终端,应进行终端深度认证.从访问控制器 1 到终端逐级进行可信认证,将完成深度可信认证的终端列入网络 2 的白名单,保证安全的同时节省资源,提高系统运行效率[13].

2.2 协议工作流程

TCA-SNI 协议有 3 个阶段[13]:终端可信入网、网间可信连接、终端深度认证.以图 2 中网络 1 与网络 2 可信连接为例,说明此协议的具体流程.其中,对于身份认证,其包括用户身份和平台身份的认证.用户身份的验证可采用静态口令、动态口令、USB KEY、生物特征等多因素身份认证方式.平台身份认证可根据平台自身的特有标识进行验证,如序列号、制造商 ID、产品型号、MAC 地址等.在本协议中,可将用户身份和平台身份进行绑定,共同发送到策略管理器进行认证,也可以定制化地接入现有的身份认证系统.

2.2.1 终端可信入网

本阶段是终端可信接入局域网过程[13].使用 TPCM 引导启动的终端 *AR*1 向 *AC*1 发送接入网络请求,*AC*1 接到 *AR*1 的请求后,与 *PM*1 共同完成与 *AR*1 的双向对等可信认证过程.此过程中,*AR*1 在发起接入请求时,处于不可信状态,只能与 *AC*1 通信.*PM*1 受到网络高安全级别保护,只能与 *AC*1 通信,因此 *AR*1 与 *PM*1 的通信均由 *AC*1 转发.具体步骤如下.

(1) *AR*1 向 *AC*1 发送包含随机数 $Rand_{AR1PM1}$ 的连接请求.

$$AR1 \rightarrow AC1:\{Rand_{AR1PM1}\} \tag{1}$$

(2) *AC*1 收到 *AR*1 连接请求,生成随机数 $Rand_{AC1PM1}$,转发至 *PM*1.

$$AC1 \rightarrow PM1:\{Rand_{AR1PM1},Rand_{AC1PM1}\} \tag{2}$$

(3) *PM*1 收到请求,保存随机数.生成两个随机数,返回公钥证书.

$$PM1 \rightarrow AC1:\{Cert_{PM1},Rand_{PM1AR1},Rand_{PM1AC1}\} \tag{3}$$

其中,$Cert_{PM1}$ 是 *PM*1 的公钥证书,$Rand_{PM1AR1}$ 是用来生成 *PM*1 与 *AR*1 之间通信密钥的随机数,$Rand_{PM1AC1}$ 是用来生成 *PM*1 与 *AC*1 之间通信密钥的随机数.

(4) $AC1$ 收到 $PM1$ 的数据包后,提取 $PM1$ 的公钥证书并进行验证:若证书验证失败,则断开连接;若证书验证成功,提取随机数,转发给 $AR1$.

$$AC1\rightarrow AR1:\{Cert_{PM1},Rand_{PM1AR1}\} \tag{4}$$

(5) $AR1$ 收到数据包后,提取 $PM1$ 的公钥证书并进行验证:若证书验证失败,则断开连接;若证书验证成功,提取证书中 $PM1$ 的公钥 Pk_{PM1}.生成随机数 Sct_{AR1PM1},使用 $PM1$ 的公钥 Pk_{PM1} 加密后发给 $AC1$.

$$AR1\rightarrow AC1:\{Sct_{AR1PM1},ID_{AR1}\}Pk_{PM1} \tag{5}$$

(6) $AC1$ 收到 $AR1$ 数据包后,生成随机数 Sct_{AC1PM1},使用 $PM1$ 的公钥 Pk_{PM1} 加密后发送 $PM1$.

$$AC1\rightarrow PM1:\{\{Sct_{AR1PM1},ID_{AR1}\}Pk_{PM1},Sct_{AC1PM1},ID_{AC1}\}Pk_{PM1} \tag{6}$$

(7) $PM1$ 收到数据包后,使用私钥解密数据包,提取出 $AC1$ 的随机数 Sct_{AC1PM1},与 $Rand_{AC1PM1}$,$Rand_{PM1AC1}$ 共同通过密钥生成算法生成与 $AC1$ 的通信密钥 K_{PM1AC1}.继续解密数据包,提取 $AR1$ 的随机数 Sct_{AR1PM1},与 $Rand_{AR1PM1}$,$Rand_{PM1AR1}$ 共同通过密钥生成算法生成与 $AR1$ 的通信密钥 K_{PM1AR1}.分别对 $PM1$-$AR1$,$PM1$-$AC1$ 的随机数进行 hash 计算,结果记为 H_{PM1AR1},H_{PM1AC1},使用 $PM1$ 私钥进行签名后返回.

$$PM1\rightarrow AC1:\{\{H_{PM1AR1},ID_{AR1}\}Pk_{PM1}^{-1},\{H_{PM1AR1},ID_{AR1}\},\{H_{PM1AC1},ID_{AC1}\}Pk_{PM1}^{-1},\{H_{PM1AC1},ID_{AC1}\}\} \tag{7}$$

(8) $AC1$ 收到数据包后,获取 $AC1$ 相关信息.使用 $PM1$ 公钥验证签名,并与本地已有随机数 Sct_{AC1PM1},$Rand_{AC1PM1}$,$Rand_{PM1AC1}$ 的 hash 值进行校验,将其余部分转发至 $AR1$:若校验成功,共同通过密钥生成算法生成与 $PM1$ 的通信密钥 K_{PM1AC1};若校验失败,断开连接.

$$AC1\rightarrow AR1:\{\{H_{PM1AR1},ID_{AR1}\}Pk_{PM1}^{-1},\{H_{PM1AR1},ID_{AR1}\}\} \tag{8}$$

(9) $AR1$ 收到数据包后,使用 $PM1$ 公钥验证签名,并与本地已有随机数 Sct_{AR1PM1},$Rand_{AR1PM1}$,$Rand_{PM1AR1}$ 的 hash 值进行校验:若校验成功,通过密钥生成算法生成与 $PM1$ 的通信密钥 K_{PM1AR1},将身份信息 U_{AR1} 用通信密钥 K_{PM1AR1} 加密,发送给 $AC1$;若校验失败,断开连接.

$$AR1\rightarrow AC1:\{U_{AR1},ID_{AR1}\}K_{PM1AR1} \tag{9}$$

(10) $AC1$ 收到数据包后,将身份信息 U_{AC1} 使用通信密钥 K_{PM1AC1} 加密后,发给 $PM1$.

$$AC1\rightarrow PM1:\{\{U_{AR1},ID_{AR1}\}K_{PM1AR1},U_{AC1},ID_{AC1}\}K_{PM1AC1} \tag{10}$$

(11) $PM1$ 收到数据包后,使用通信密钥 K_{PM1AC1} 解密,获取 $AC1$ 的身份信息 U_{AC1}.使用通信密钥 K_{PM1AR1} 解密,获取 $AR1$ 的身份信息 U_{AR1}.对 $AC1$,$AR1$ 的身份信息进行验证:若 U_{AR1} 或/和 U_{AC1} 验证成功,将成功标识 RU_{AC1} 或/和 RU_{AR1} 加密后返回;若 U_{AR1} 或/和 U_{AC1} 验证失败,则不返回.

$$PM1\rightarrow AC1:\{\{RU_{AC1},ID_{AR1}\}K_{PM1AR1},RU_{AR1},ID_{AC1}\}K_{PM1AC1} \tag{11}$$

其中,RU_{AR1} 是 $PM1$ 对 $AR1$ 的身份信息 U_{AR1} 验证成功的标识,返回给 $AC1$;RU_{AC1} 是 $PM1$ 对 $AC1$ 的身份信息 U_{AC1} 验证成功的标识,返回给 $AR1$.

(12) $AC1$ 收到数据包后,使用通信密钥 K_{PM1AC1} 解密数据包,获取 RU_{AR1},并转发数据包其余内容至 $AR1$.若获取 RU_{AR1} 失败,禁止 $AR1$ 连入网络.

$$AC1\rightarrow AR1:\{RU_{AC1},ID_{AR1}\}K_{PM1AR1} \tag{12}$$

(13) $AR1$ 收到数据包后,使用通信密钥 K_{PM1AR1} 解密数据包,获取 RU_{AC1}:若获取 RU_{AC1} 失败,拒绝连入该网络,主动断开连接;若成功获取 RU_{AC1},将平台完整性信息 I_{AR1} 用通信密钥 K_{PM1AR1} 加密后,发送至 $AC1$.

$$AR1\rightarrow AC1:\{\{I_{AR1},ID_{AR1}\}K_{PM1AR1} \tag{13}$$

(14) $AC1$ 收到数据包后,将平台完整性信息 I_{AC1} 使用通信密钥 K_{PM1AC1} 加密后,发送至 $PM1$.

$$AC1\rightarrow PM1:\{\{I_{AR1},ID_{AR1}\}K_{PM1AR1},I_{AC1},ID_{AC1}\}K_{PM1AC1} \tag{14}$$

(15) $PM1$ 收到数据包后,使用通信密钥 K_{PM1AC1} 解密,获取 $AC1$ 的平台完整性信息 I_{AC1}.使用通信密钥 K_{PM1AR1} 解密,获取 $AR1$ 的平台完整性信息 I_{AR1}.对 $AC1$,$AR1$ 的平台完整性信息进行度量:若 I_{AR1} 或/和 I_{AC1} 度量成功,将访问决策 RI_{AC1} 或/和 RI_{AR1} 加密后返回;若 I_{AR1} 或/和 I_{AC1} 度量失败,则不返回.

$$PM1\rightarrow AC1:\{\{RI_{AC1},ID_{AR1}\}K_{PM1AR1},RI_{AR1},ID_{AC1}\}K_{PM1AC1} \tag{15}$$

其中,RI_{AR1} 是 $PM1$ 根据 $AR1$ 的平台完整性信息 I_{AR1} 度量成功的标识,返回给 $AC1$;RI_{AC1} 是 $PM1$ 对 $AC1$ 的平台

完整性信息 I_{AC1} 度量成功的标识,返回给 *AR*1.

(16) *AC*1 收到数据包后,使用通信密钥 K_{PM1AC1} 解密数据包,获取 RI_{AR1},并转发其余内容至 *AR*1.若获取 RI_{AR1} 失败,禁止 *AR*1 连入网络.

$$AC1\rightarrow AR1:\{RI_{AC1},ID_{AR1}\}K_{PM1AR1} \tag{16}$$

(17) *AR*1 收到数据包后,使用通信密钥 K_{PM1AR1} 解密数据包,获取 RI_{AC1}:若获取 RI_{AC1} 失败,拒绝连入该网络,主动断开连接;若成功获取 RI_{AC1},表示已可信连入该网络,可进行正常数据通信.

本阶段可信认证,可保证任意终端接入局域网时双方的身份和平台可信,保障局域网整体可信.此阶段是进行网间可信连接的必要条件.

2.2.2 网间可信连接

本阶段是网络 1 与网络 2 的可信连接过程[13].*AC*2 收到 *AC*1 发起网络连接请求后,与 *PM*1,*PM*2 共同完成四元对等的可信认证过程.此过程中,网络 1 与网络 2 互不信任,网络间仅有 *AC*1 和 *AC*2 可互相通信,完成认证过程.具体步骤如下.

(1) *AC*1 向 *PM*1 请求公钥证书,并发送随机数 $Rand_{AC1PM1}$.

$$AC1\rightarrow PM1:\{Rand_{AC1PM1}\} \tag{17}$$

(2) *PM*1 收到 *AC*1 的数据包,获取随机数 $Rand_{AC1PM1}$,生成随机数 $Rand_{PM1AC1}$,$Rand_{PM1AC2}$,同 *PM*1 公钥证书 $Cert_{PM1}$ 一同返回.

$$PM1\rightarrow AC1:\{Cert_{PM1},Rand_{PM1AC1},Rand_{PM1AC2}\} \tag{18}$$

(3) *AC*1 收到 *PM*1 的数据包后,获取并验证 *PM*1 公钥证书 $Cert_{PM1}$:若证书验证失败,则断开连接;若证书验证成功,生成随机数 $Rand_{AC1PM2}$,发送至 *AC*2.

$$AC1\rightarrow AC2:\{Rand_{AC1PM2},Cert_{PM1},Rand_{PM1AC2}\} \tag{19}$$

(4) *AC*2 收到 *AC*1 的数据包后,获取并验证 *PM*1 公钥证书 $Cert_{PM1}$:若证书验证失败,则断开连接;若证书验证成功,提取 *PM*1 公钥 Pk_{PM1},获取随机数 $Rand_{PM1AC2}$.生成随机数 $Rand_{AC2PM2}$,发送至 *PM*2.

$$AC2\rightarrow PM2:\{Rand_{AC1PM2},Rand_{AC2PM2}\} \tag{20}$$

(5) *PM*2 收到数据包后,获取随机数 $Rand_{AC1PM2}$,$Rand_{AC2PM2}$,生成随机数 $Rand_{PM2AC2}$,$Rand_{PM2AC1}$,同 *PM*2 的公钥证书 $Cert_{PM2}$ 返回至 *AC*2.

$$PM2\rightarrow AC2:\{Rand_{PM2AC2},Cert_{PM2},Rand_{PM2AC1}\} \tag{21}$$

(6) *AC*2 收到数据包后,获取并验证 *PM*2 公钥证书 $Cert_{PM2}$:若证书验证失败,则断开连接;若证书验证成功,提取 *PM*2 公钥 Pk_{PM2},获取随机数 $Rand_{PM2AC2}$.生成随机数 $Rand_{AC2PM1}$,Sct_{AC2PM1},使用 *PM*1 公钥 Pk_{PM1} 加密 Sct_{AC2PM1} 后,发送给 *AC*1.

$$AC2\rightarrow AC1:\{\{Sct_{AC2PM1},ID_{AC2}\}Pk_{PM1},Rand_{AC2PM1},Cert_{PM2},Rand_{PM2AC1}\} \tag{22}$$

(7) *AC*1 收到数据包后,获取并验证 *PM*2 公钥证书 $Cert_{PM2}$:若证书验证失败,则断开连接;若证书验证成功,提取 *PM*2 公钥 Pk_{PM2},获取数据包中随机数 $Rand_{PM2AC1}$.生成随机数 Sct_{AC1PM1},使用 *PM*1 公钥 Pk_{PM1} 加密后,发送至 *PM*1.

$$AC1\rightarrow PM1:\{\{Sct_{AC2PM1},ID_{AC2}\}Pk_{PM1},Rand_{AC2PM1},\{Sct_{AC1PM1},ID_{AC1}\}Pk_{PM1}\} \tag{23}$$

(8) *PM*1 收到数据包后,使用私钥解密数据包,提取随机数 Sct_{AC2PM1},与 $Rand_{AC2PM1}$,$Rand_{PM1AC2}$ 共同通过密钥生成算法生成与 *AC*2 的通信密钥 K_{PM1AC2};提取随机数 Sct_{AC1PM1},与 $Rand_{AC1PM1}$,$Rand_{PM1AC1}$ 共同通过密钥生成算法生成与 *AC*1 的通信密钥 K_{PM1AC1}.分别对 *PM*1-*AC*1,*PM*1-*AC*2 的随机数进行 hash 计算,结果记为 H_{PM1AC1},H_{PM1AC2},使用 *PM*1 私钥进行签名后,返回至 *AC*1.

$$PM1\rightarrow AC1:\{\{H_{PM1AC2},ID_{AC2}\}Pk_{PM1}^{-1},\{H_{PM1AC2},ID_{AC2}\},\{H_{PM1AC1},ID_{AC1}\}Pk_{PM1}^{-1},\{H_{PM1AC1},ID_{AC1}\}\} \tag{24}$$

(9) *AC*1 收到数据包后,获取 *AC*1 相关信息.使用 *PM*1 公钥验证签名,并与本地已有随机数 Sct_{AC1PM1},$Rand_{PM1AC1}$,$Rand_{AC1PM1}$ 的 hash 值进行校验:若校验成功,共同通过密钥生成算法生成与 *PM*1 的通信密钥 K_{PM1AC1},生成随机数 Sct_{AC1PM2},使用 *PM*2 公钥 Pk_{PM2} 加密后,发送至 *AC*2;若校验失败,断开连接.

$$AC1 \rightarrow AC2:\{\{H_{PM1AC2}, ID_{AC2}\}Pk_{PM1}^{-1}, \{H_{PM1AC2}, ID_{AC2}\}, \{Sct_{AC1PM2}, ID_{AC1}\}Pk_{PM2}\} \tag{25}$$

(10) $AC2$ 收到数据包后,获取 $AC2$ 相关信息.使用 $PM1$ 公钥验证签名,并与本地已有随机数 Sct_{AC2PM1}, $Rand_{PM1AC2}$,$Rand_{AC2PM1}$ 的 hash 值进行校验:若校验成功,共同通过密钥生成算法生成与 $PM1$ 的通信密钥 K_{PM1AC2},生成随机数 Sct_{AC2PM2},使用 $PM2$ 公钥 Pk_{PM2} 加密后,发送至 $PM2$;若校验失败,断开连接.

$$AC2 \rightarrow PM2:\{\{Sct_{AC1PM2}, ID_{AC1}\}Pk_{PM2}, \{Sct_{AC2PM2}, ID_{AC2}\}Pk_{PM2}\} \tag{26}$$

(11) $PM2$ 收到数据包后,使用私钥解密数据包,提取随机数 Sct_{AC1PM2},与 $Rand_{AC1PM2}$,$Rand_{PM2AC1}$ 共同通过密钥生成算法生成与 $AC1$ 的通信密钥 K_{PM2AC1};提取随机数 Sct_{AC2PM2},与 $Rand_{AC2PM2}$,$Rand_{PM2AC2}$ 共同通过密钥生成算法生成与 $AC2$ 的通信密钥 K_{PM2AC2}.分别对 $PM2$-$AC1$,$PM2$-$AC2$ 的随机数进行 hash 计算,结果记为 H_{PM2AC1}, H_{PM2AC2},使用 $PM2$ 私钥进行签名后,返回至 $AC2$.

$$PM2 \rightarrow AC2:\{\{H_{PM2AC1}, ID_{AC1}\}Pk_{PM2}^{-1}, \{H_{PM2AC1}, ID_{AC1}\}, \{H_{PM2AC2}, ID_{AC2}\}Pk_{PM2}^{-1}, \{H_{PM2AC2}, ID_{AC2}\}\} \tag{27}$$

(12) $AC2$ 收到数据包后,获取 $AC2$ 相关信息.使用 $PM2$ 公钥验证签名,并与本地已有随机数 Sct_{AC2PM2}, $Rand_{AC2PM2}$,$Rand_{PM2AC2}$ 的 hash 值进行校验:若校验成功,共同通过密钥生成算法生成与 $PM2$ 的通信密钥 K_{PM2AC2},将身份信息 U_{AC2} 使用通信密钥 K_{PM1AC2} 加密后,发送至 $AC1$;若校验失败,断开连接.

$$AC2 \rightarrow AC1:\{\{H_{PM2AC1}, ID_{AC1}\}Pk_{PM2}^{-1}, \{H_{PM2AC1}, ID_{AC1}\}, \{U_{AC2}, ID_{AC2}\}K_{PM1AC2}\} \tag{28}$$

(13) $AC1$ 收到数据包后,获取 $AC1$ 相关信息.使用 $PM2$ 公钥验证签名并与本地已有随机数 Sct_{AC1PM2}, $Rand_{AC1PM2}$,$Rand_{PM2AC1}$ 的 hash 值进行校验:若校验成功,共同通过密钥生成算法生成与 $PM2$ 的通信密钥 K_{PM2AC1},将身份信息 U_{AC1} 使用通信密钥 K_{PM1AC2} 加密后,发送至 $PM1$;若校验失败,断开连接.

$$AC1 \rightarrow PM1:\{\{U_{AC1}, ID_{AC1}\}K_{PM1AC1}, \{U_{AC2}, ID_{AC2}\}K_{PM1AC2}\} \tag{29}$$

(14) $PM1$ 收到数据包后,使用通信密钥 K_{PM1AC1},K_{PM1AC2} 解密,获取 $AC1$ 的身份信息 U_{AC1} 和 $AC2$ 的身份信息 U_{AC2}.对 $AC1$,$AC2$ 的身份信息进行验证:若 U_{AC1} 或/和 U_{AC2} 验证成功,将成功标识 RU_{PM1AC1} 或/和 RU_{PM1AC2} 加密后返回;若 U_{AC1} 或/和 U_{AC2} 验证失败,则不返回.

$$PM1 \rightarrow AC1:\{\{RU_{PM1AC2}, ID_{AC1}\}K_{PM1AC1}, \{RU_{PM1AC1}, ID_{AC2}\}K_{PM1AC2}\} \tag{30}$$

其中,RU_{PM1AC1} 是 $PM1$ 对 $AC1$ 的身份信息 U_{AC1} 验证成功的标识,加密后返回给 $AC2$;RU_{PM1AC2} 是 $PM1$ 对 $AC2$ 的身份信息 U_{AC2} 验证成功的标识,加密后返回给 $AC1$.

(15) $AC1$ 收到数据包后,使用通信密钥 K_{PM1AC1} 解密数据包,获取身份验证成功标识 RU_{PM1AC2}.将 $AC1$ 的身份信息 U_{AC1} 使用通信密钥 K_{PM2AC1} 加密后,与数据包其余内容一同发送至 $AC2$.若获取 RU_{PM1AC2} 失败,禁止 $AC2$ 连入网络.

$$AC1 \rightarrow AC2:\{\{U_{AC1}, ID_{AC1}\}K_{PM2AC1}, \{RU_{PM1AC1}, ID_{AC2}\}K_{PM1AC2}\} \tag{31}$$

(16) $AC2$ 收到数据包后,使用通信密钥 K_{PM1AC2} 解密数据包,获取身份验证成功标识 RU_{PM1AC1}.将 $AC2$ 的身份信息 U_{AC2} 使用通信密钥 K_{PM2AC2} 加密后,与数据包其余内容一同发送至 $PM2$.若获取 RU_{PM1AC1} 失败,断开与 $AC1$ 的网络连接.

$$AC2 \rightarrow PM2:\{\{U_{AC1}, ID_{AC1}\}K_{PM2AC1}, \{U_{AC2}, ID_{AC2}\}K_{PM2AC2}\} \tag{32}$$

(17) $PM2$ 收到数据包后,使用通信密钥 K_{PM2AC1},K_{PM2AC2} 解密,获取 $AC1$ 的身份信息 U_{AC1} 和 $AC2$ 的身份信息 U_{AC2}.对 $AC1$,$AC2$ 的身份信息进行验证:若 U_{AC1} 或/和 U_{AC2} 验证成功,将成功标识 RU_{PM2AC1} 或/和 RU_{PM2AC2} 加密后返回;若 U_{AC1} 或/和 U_{AC2} 验证失败,则不返回.

$$PM2 \rightarrow AC2:\{\{RU_{PM2AC2}, ID_{AC1}\}K_{PM2AC1}, \{RU_{PM2AC1}, ID_{AC2}\}K_{PM2AC2}\} \tag{33}$$

其中,RU_{PM2AC1} 是 $PM2$ 对 $AC1$ 的身份信息 U_{AC1} 验证成功的标识,加密后返回给 $AC2$;RU_{PM2AC2} 是 $PM2$ 对 $AC2$ 的身份信息 U_{AC2} 验证成功的标识,加密后返回给 $AC1$.

(18) $AC2$ 收到数据包后,使用通信密钥 K_{PM2AC2} 解密数据包,获取身份验证成功标识 RU_{PM2AC1}.将 $AC2$ 的平台完整性信息 I_{AC2} 使用通信密钥 K_{PM1AC2} 加密后,与数据包其余内容一同发送至 $AC1$.若获取 RU_{PM2AC1} 失败,禁止 $AC1$ 连入网络.

$$AC2 \rightarrow AC1:\{\{RU_{AC2}, ID_{AC1}\}K_{PM2AC1}, \{I_{AC2}, ID_{AC2}\}K_{PM1AC2}\} \tag{34}$$

(19) $AC1$ 收到数据包后,使用通信密钥 K_{PM2AC1} 解密数据包,获取身份验证成功标识 RU_{PM2AC2}.将 $AC1$ 的平台完整性信息 I_{AC1} 使用通信密钥 K_{PM1AC1} 加密后,与数据包其余内容一同发送至 $PM1$.若获取 RU_{PM2AC2} 失败,断开与 $AC2$ 的网络连接.

$$AC1\rightarrow PM1:\{\{I_{AC1},ID_{AC1}\}K_{PM1AC1},\{I_{AC2},ID_{AC2}\}K_{PM1AC2}\} \tag{35}$$

(20) $PM1$ 收到数据包后,使用通信密钥 K_{PM1AC1},K_{PM1AC2} 解密,获取 $AC1$ 的平台完整性信息 I_{AC1} 和 $AC2$ 的平台完整性信息 I_{AC2},以 $PM1$ 的可信评估策略进行度量:若 I_{AC1} 或/和 I_{AC2} 度量成功,将成功标识 RI_{PM1AC1} 或/和 RI_{PM1AC2} 分别加密后返回;若 I_{AC1} 或/和 I_{AC2} 度量失败,则不返回.

$$PM1\rightarrow AC1:\{\{RI_{PM1AC2},ID_{AC1},TES_{PM1}\}K_{PM1AC1},\{RI_{PM1AC1},ID_{AC2}\}K_{PM1AC2}\} \tag{36}$$

其中,RI_{PM1AC1} 是 $PM1$ 对 $AC1$ 的平台完整性 I_{AC1} 度量成功标识,加密后返回给 $AC2$;RI_{PM1AC2} 是 $PM1$ 对 $AC2$ 的平台完整性 I_{AC2} 度量成功标识,加密后返回至 $AC1$.TES_{PM1} 是 $PM1$ 的可信评估策略,最终返回至 $PM2$,进行安全策略互评.

(21) $AC1$ 收到数据包后,使用通信密钥 K_{PM1AC1} 解密,获取平台完整性度量成功标识 RI_{PM1AC2} 和 $PM1$ 的可信评估策略 TES_{PM1}.将 TES_{PM1} 和 $AC1$ 的平台完整性信息 I_{AC1} 使用通信密钥 K_{PM2AC1} 加密后,与数据包其余内容一同发送至 $AC2$.若获取 RI_{PM1AC2} 失败,禁止 $AC2$ 连入网络.

$$AC1\rightarrow AC2:\{\{I_{AC1},ID_{AC1},TES_{PM1}\}K_{PM2AC1},\{RI_{PM1AC1},ID_{AC2}\}K_{PM1AC2}\} \tag{37}$$

(22) $AC2$ 收到数据包后,使用通信密钥 K_{PM1AC2} 解密,获取平台完整性度量成功标识 RI_{PM1AC1}.将 $AC2$ 的平台完整性信息 I_{AC2} 使用通信密钥 K_{PM2AC2} 加密后,与数据包其余内容一同发送至 $PM2$.若获取 RI_{PM1AC1} 失败,断开与 $AC1$ 的网络连接.

$$AC2\rightarrow PM2:\{\{I_{AC1},ID_{AC1},TES_{PM1}\}K_{PM2AC1},\{I_{AC2},ID_{AC2}\}K_{PM2AC2}\} \tag{38}$$

(23) $PM2$ 收到数据包后,使用通信密钥 K_{PM2AC1},K_{PM2AC2} 解密,获取 $AC1$ 的平台完整性信息 I_{AC1},$PM1$ 的可信评估策略 TES_{PM1} 和 $AC2$ 的平台完整性信息 I_{AC2}.对 TES_{PM1} 进行评估,若评估结果为可信,生成评估成功标识 $RTES_{PM2PM1}$.对 $AC1$,$AC2$ 的平台完整性信息以 $PM2$ 的可信评估策略进行度量:若 I_{AC1} 或/和 I_{AC2} 度量成功,生成度量成功标识 RI_{PM2AC1} 或/和 RI_{PM2AC2},将 $PM2$ 的可信评估策略 TES_{PM2} 与评估和度量结果标识一同加密后返回;若评估或度量失败,则不返回结果标识.

$$PM2\rightarrow AC2:\{\{RI_{PM2AC1},RTES_{PM2PM1},ID_{AC2},TES_{PM2}\}K_{PM2AC2},\{RI_{PM2AC2},ID_{AC1}\}K_{PM2AC1}\} \tag{39}$$

其中,RI_{PM2AC1} 是 $PM2$ 对 $AC1$ 的平台完整性 I_{AC1} 度量成功标识,加密后返回给 $AC2$;RI_{PM2AC2} 是 $PM2$ 对 $AC2$ 的平台完整性 I_{AC2} 度量成功标识,加密后返回至 $AC1$.TES_{PM2} 是 $PM2$ 的可信评估策略,最终返回至 $PM1$,进行安全策略互评.

(24) $AC2$ 收到数据包后,使用通信密钥 K_{PM2AC2} 解密,获取平台完整性度量成功标识 RI_{PM2AC1}、策略评估成功标识 $RTES_{PM2PM1}$ 和 $PM2$ 的可信评估策略 TES_{PM2}.将 TES_{PM2} 使用通信密钥 K_{PM1AC2} 加密,与数据包其余内容一同发送至 $AC1$:若获取 $RTES_{PM2PM1}$,RI_{PM2AC1} 失败,禁止 $AC1$ 连入网络;若获取成功,则允许网络 1 进行正常的数据通信.

$$AC2\rightarrow AC1:\{\{TES_{PM2}\}K_{PM1AC2},\{RI_{PM2AC2},ID_{AC1}\}K_{PM2AC1}\} \tag{40}$$

(25) $AC1$ 收到数据包后,使用通信密钥 K_{PM2AC1} 解密,获取平台完整性度量成功标识 RI_{PM2AC2},将数据包其余内容发送至 $PM1$.若获取 RI_{PM2AC2} 失败,断开与 $AC2$ 的网络连接.

$$AC1\rightarrow PM1:\{\{TES_{PM2}\}K_{PM1AC2}\} \tag{41}$$

(26) $PM1$ 收到数据包后,使用通信密钥 K_{PM1AC2} 解密,获取 $PM2$ 的可信评估策略 TES_{PM2},并对 $PM2$ 的可信评估策略进行评估:若评估结果为可信,加密返回评估成功标识 $RTES_{PM1PM2}$;否则,不返回成功标识.

$$PM1\rightarrow AC1:\{\{RTES_{PM1PM2}\}K_{PM1AC1}\} \tag{42}$$

(27) $AC1$ 收到数据包后,使用通信密钥 K_{PM1AC1} 解密,获取策略评估成功标识 $RTES_{PM1PM2}$:若获取失败,拒绝 $AC2$ 连入网络;若获取成功,则允许网络 2 进行正常数据通信.

本阶段可信认证,可以使网络 1 与网络 2 进行正常通信,完成初级命令操作.若部分终端需要进行高级操作,

则需进行终端深度认证.

2.2.3 终端深度认证

本阶段是网络 1 中部分终端向网络 2 发送高级操作命令前进行的深度可信认证过程[13].当网络 1 中的终端 *AR*1 向网络 2 发送高级操作命令时,需要进行深度可信认证.该过程由 *PM*2,*AC*2 共同参与,完成对 *AR*1 的三元对等认证过程.此过程中,*PM*2 受网络 2 的高等级保护,仅能与 *AC*2 通信,*AR*1 与 *PM*2 的通信由 *AC*2 进行转发.具体步骤如下.

(1) *AR*1 向 *AC*2 发送包含随机数 $Rand_{AR1PM2}$ 的认证请求.

$$AR1\rightarrow AC2:\{Rand_{AR1PM2}\} \tag{43}$$

(2) *AC*2 收到 *AR*1 认证请求后,生成随机数 $Rand_{AC2PM2}$ 并转发至 *PM*2.

$$AC2\rightarrow PM2:\{Rand_{AR1PM2},Rand_{AC2PM2}\} \tag{44}$$

(3) *PM*2 收到数据包后,获取随机数.生成两个随机数,并返回公钥证书至 *AC*2.

$$PM2\rightarrow AC2:\{Cert_{PM2},Rand_{PM2AR1},Rand_{PM2AC2}\} \tag{45}$$

其中,$Cert_{PM2}$ 是 *PM*2 的公钥证书,$Rand_{PM2AR1}$ 是用来生成 *PM*2 与 *AR*1 之间通信密钥的随机数,$Rand_{PM2AC2}$ 是用来生成 *PM*2 与 *AC*2 之间通信密钥的随机数.

(4) *AC*2 收到 *PM*2 的数据包后,获取 *PM*2 的公钥证书并进行验证:若证书验证失败,则断开连接;若证书验证成功,提取随机数,转发给 *AR*1.

$$AC2\rightarrow AR1:\{Cert_{PM2},Rand_{PM2AR1}\} \tag{46}$$

(5) *AR*1 收到数据包后,提取 *PM*2 的公钥证书并进行验证:若证书验证失败,则断开连接;若证书验证成功,提取证书中 *PM*2 的公钥 Pk_{PM2},生成随机数 Sct_{AR1PM2},使用 *PM*2 的公钥 Pk_{PM2} 加密后,发送至 *AC*2.

$$AR1\rightarrow AC2:\{Sct_{AR1PM2},ID_{AR1}\}Pk_{PM2} \tag{47}$$

(6) *AC*2 收到数据包后,生成随机数 Sct_{AC2PM2},使用 *PM*2 的公钥 Pk_{PM2} 加密后,发送至 *PM*2.

$$AC2\rightarrow PM2:\{\{Sct_{AR1PM2},ID_{AR1}\}Pk_{PM2},Sct_{AC2PM2},ID_{AC2}\}Pk_{PM2} \tag{48}$$

(7) *PM*2 收到数据包后,使用私钥解密,提取 *AC*2 的随机数 Sct_{AC2PM2},与 $Rand_{AC2PM2}$,$Rand_{PM2AC2}$ 共同通过密钥生成算法生成与 *AC*2 的通信密钥 K_{PM2AC2}.继续使用私钥解密数据包,提取 *AR*1 的随机数 Sct_{AR1PM2},与 $Rand_{AR1PM2}$,$Rand_{PM2AR1}$ 共同通过密钥生成算法生成与 *AR*1 的通信密钥 K_{PM2AR1}.分别对 *PM*2-*AR*1,*PM*2-*AC*2 的随机数进行 hash 计算,结果记为 H_{PM2AR1},H_{PM2AC2},使用 *PM*2 私钥进行签名后返回至 *AC*2.

$$PM2\rightarrow AC2:\{\{H_{PM2AR1},ID_{AR1}\}Pk_{PM2}^{-1},\{H_{PM2AR1},ID_{AR1}\},\{H_{PM2AC2},ID_{AC2}\}Pk_{PM2}^{-1},\{H_{PM2AC2},ID_{AC2}\}\} \tag{49}$$

(8) *AC*2 收到数据包后,获取 *AC*2 相关信息.使用 *PM*2 公钥验证签名,并与本地已有随机数 Sct_{AC2PM2},$Rand_{AC2PM2}$,$Rand_{PM2AC2}$ 的 hash 值进行校验,将数据包其余部分转发至 *AR*1:若校验成功,通过密钥生成算法生成与 *PM*2 的通信密钥 K_{PM2AC2};若校验失败,断开连接.

$$AC2\rightarrow AR1:\{\{H_{PM2AR1},ID_{AR1}\}Pk_{PM2}^{-1},\{H_{PM2AR1},ID_{AR1}\}\} \tag{50}$$

(9) *AR*1 收到数据包后,使用 *PM*2 公钥验证签名,并与本地已有的 3 个随机数 Sct_{AR1PM2},$Rand_{AR1PM2}$,$Rand_{PM2AR1}$ 的 hash 值进行校验:若校验成功,通过密钥生成算法生成与 *PM*2 的通信密钥 K_{PM2AR1},将平台完整性信息 I_{AR1} 使用通信密钥 K_{PM2AR1} 加密后,发送至 *AC*2;若校验失败,断开连接.

$$AR1\rightarrow AC2:\{I_{AR1},ID_{AR1}\}K_{PM2AR1} \tag{51}$$

(10) *AC*2 收到数据包后,使用通信密钥 K_{PM2AC2} 加密平台完整性信息 I_{AC2},发送至 *PM*2.

$$AC2\rightarrow PM2:\{\{I_{AR1},ID_{AR1}\}K_{PM2AR1},I_{AC2},ID_{AC2}\}K_{PM2AC2} \tag{52}$$

(11) *PM*2 收到数据包后,使用通信密钥 K_{PM2AC2} 解密,获取 *AC*2 的平台完整性信息 I_{AC2}.使用通信密钥 K_{PM2AR1} 解密,获取 *AR*1 的平台完整性信息 I_{AR1}.并对 *AC*2,*AR*1 的平台完整性信息进行度量:若 I_{AR1} 或/和 I_{AC2} 度量成功,将成功标识 RI_{AR1} 或/和 RI_{AC2} 加密后返回至 *AC*2;若 I_{AR1} 或/和 I_{AC2} 度量失败,则不返回.

$$PM2\rightarrow AC2:\{\{RI_{AC2},ID_{AR1}\}K_{PM2AR1},RI_{AR1},ID_{AC2}\}K_{PM2AC2} \tag{53}$$

其中,RI_{AR1} 是 *PM*2 根据 *AR*1 的平台完整性信息 I_{AR1} 度量成功的标识,返回给 *AC*2;RI_{AC2} 是 *PM*2 对 *AC*2 的平台

完整性信息 I_{AC2} 度量成功的标识,返回给 *AR*1.

(12) *AC*2 收到数据包后,使用通信密钥 K_{PM2AC2} 解密,获取 RI_{AR1},并转发数据包其余内容至 *AR*1:若获取失败,禁止接收 *AR*1 发送的高级操作命令;若获取成功,则允许接收 *AR*1 发送的高级操作命令.

$$AC2\rightarrow AR1:\{RI_{AC2},ID_{AR1}\}K_{PM2AR1} \tag{54}$$

(13) *AR*1 收到数据包,使用通信密钥 K_{PM2AR1} 解密,获取 RI_{AC2}:若获取失败,则未被允许发送高级操作命令;若获取成功,则可进行高级操作命令的通信.

本阶段认证,根据网络 2 的可信评估策略对要进行高级命令操作的终端进行可信评估,若评估结果可信,则说明该终端可信,可进行高级命令操作.

3 安全性推理及分析

3.1 SVO逻辑

SVO 逻辑[14−16]是 BAN 逻辑的一种扩展,是由 Syverson 和 Orschot 在 BAN 逻辑[17]、GNY 逻辑[18]、AT 逻辑[19]、VO 逻辑[19]等逻辑系统的基础上提出的,具有以上逻辑的优点,同时又具有十分简洁的推理规则和公理,是 BAN 类逻辑中较为成熟的逻辑系统.

3.2 SVO逻辑分析协议

3.2.1 基本语义

SVO 逻辑与 BAN 类逻辑相似,使用符号|≡,◁,|~,|≈,|⇒,∋,#,≡分别表示相信(believed)、接收到(reveived)、发送过(said)、新发送过(says)、管辖(controls)、拥有(has)、新鲜(fresh)与等价(equivalent)[20].

(1) *P*,*Q*:表示具体的通信主体;

(2) K_{ab},K_{as}:表示具体的通信主体的共享密钥;

(3) K_a,K_b:表示具体的通信主体的公开密钥;

(4) K_a^{-1},K_b^{-1}:表示具体的通信主体的私密密钥;

(5) *X*,*Y*:表示一般意义上的语句;

(6) *P*|≡*X*:P 相信 *X*,*P* 认为 *X* 为真;

(7) *P*|◁*X*:*P* 曾收到包含 *X* 的消息,*P* 能读出并重复 *X*;

(8) *P*|~*X*:*P* 曾发送包含 *X* 的消息;

(9) #(*X*):表示 *X* 是新鲜的;

(10) $P\xleftrightarrow{K}Q$:*K* 是 *P* 和 *Q* 的共享密钥;

(11) *PK*(*P*,*K*):*K* 是 *P* 的公钥;

(12) $\{X^P\}_K$:消息 *X* 被密钥 *K* 加密后的结果,*P* 是发送者(常省略);

(13) $[X]_K$:密钥 *K* 对消息 *X* 签名后的结果,即$[X]_K=(X,\{H(X)\}_K)$;

(14) *SV*(*X*,*K*,*Y*):应用密钥 *K* 可以验证 *X* 是 *Y* 的签名;

3.2.2 SVO 逻辑基本公理

仅列举本文分析 TCA-SNI 协议安全性时涉及到的基本公理[21,22].

(1) 信任公理

A1:$(P|\equiv\alpha\wedge P|\equiv\psi)\equiv(P|\equiv\alpha\wedge\psi)$

主体相信所有来自他的信念的信念.

(2) 接收公理

A2:$P\triangleleft(X_1,\ldots,X_n)\supset P\triangleleft X_i$;

A3: $(P\triangleleft\{X\}_K\wedge P\ni\tilde{K})\supset P\triangleleft X$;

其中, $\tilde{K}$ 是 K 的解密密钥.

(3) 消息拥有公理

A4:$P \triangleleft X \supset P \ni X$;

(4) “好的”共享密码对称性公理

A5: $P \xleftrightarrow{K} Q \equiv Q \xleftrightarrow{K} P$.

3.2.3 扩展 SVO 逻辑

为了使 SVO 逻辑能够分析 TCA-SNI 协议,我们对 SVO 逻辑基本语义进行扩展[23−26].

- 语义(15)

$f_k(\cdot)$表示共享密钥生成算法.

$f_k(X_1,X_2,X_3)$表示使用参数(X_1,X_2,X_3)生成共享密钥.

- 语义(16)

$H(\cdot)$表示 TPCM 中提供的 *Hash*$(\cdot)$函数.

$H(X_1,X_2,X_3)$表示将 X_1,X_2,X_3 用 TPCM 提供的 *Hash*$(\cdot)$函数运算的结果.

- 公理 A6

$P \ni X \wedge Q \ni Y \wedge P \mid\equiv (PK(Q,K)) \wedge P \mid\equiv (Q \mid\sim [Y]_{K^{-1}}) \wedge P \mid\equiv (P \triangleleft \#([Y]_{K^{-1}})) \wedge SV([Y]_{K^{-1}},K,X) \supset X = Y$.

此条公理说明,若 $[Y]_{K^{-1}}$ 是 X 签名,则 X 和 Y 是完全相同的.其中,K^{-1} 是 Q 的签名密钥.

3.2.4 协议理想化描述

将 TCA-SNI 协议流程使用 SVO 逻辑符号进行理想化描述[13].

- 第 1 阶段理想化描述.

(1-5) $AR1 \to AC1: \{Sct_{AR1PM1}, ID_{AR1}\}_{K_{PM1}}$;

(1-6) $AC1 \to PM1: \{Sct_{AR1PM1}, ID_{AR1}\}_{K_{PM1}}, \{Sct_{AC1PM1}, ID_{AC1}\}_{K_{PM1}}$;

(1-7) $PM1 \to AC1: [H_{PM1AR1}, ID_{AR1}]_{Pk_{PM1}^{-1}}, [H_{PM1AC1}, ID_{AC1}]_{Pk_{PM1}^{-1}}$;

(1-8) $AC1 \to AR1: [H_{PM1AR1}, ID_{AR1}]_{Pk_{PM1}^{-1}}$;

(1-9) $AR1 \to AC1: \{U_{AR1}, ID_{AR1}\}_{K_{PM1AR1}}$;

(1-10) $AC1 \to PM1: \{U_{AR1}, ID_{AR1}\}_{K_{PM1AR1}}, \{U_{AC1}, ID_{AC1}\}_{K_{PM1AC1}}$;

(1-11) $PM1 \to AC1: \{RU_{AR1}, ID_{AC1}\}_{K_{PM1AR1}}, \{RU_{AC1}, ID_{AR1}\}_{K_{PM1AC1}}$;

(1-12) $AC1 \to AR1: \{RU_{AC1}, ID_{AR1}\}_{K_{PM1AR1}}$;

(1-13) $AR1 \to AC1: \{I_{AR1}, ID_{AR1}\}_{K_{PM1AR1}}$;

(1-14) $AC1 \to PM1: \{I_{AR1}, ID_{AR1}\}_{K_{PM1AR1}}, \{I_{AC1}, ID_{AC1}\}_{K_{PM1AC1}}$;

(1-15) $PM1 \to AC1: \{RI_{AC1}, ID_{AR1}\}_{K_{PM1AR1}}, \{RI_{AR1}, ID_{AC1}\}_{K_{PM1AC1}}$;

(1-16) $AC1 \to AR1: \{RI_{AC1}, ID_{AR1}\}_{K_{PM1AR1}}$.

步骤 1~步骤 4 省略,因为明文传输可以被伪造,对 TCA-SNI 协议安全性没有影响,所以不进行理想化描述.

- 第 2 阶段理想化描述.

(2-6) $AC2 \to AC1: \{Sct_{AC2PM1}, ID_{AC2}\}_{K_{PM1}}$;

(2-7) $AC1 \to PM1: \{Sct_{AC2PM1}, ID_{AC2}\}_{K_{PM1}}, \{Sct_{AC1PM1}, ID_{AC1}\}_{K_{PM1}}$;

(2-8) $PM1 \to AC1: [H_{PM1AC2}, ID_{AC2}]_{Pk_{PM1}^{-1}}, [H_{PM1AC1}, ID_{AC1}]_{Pk_{PM1}^{-1}}$;

(2-9) $AC1 \to AC2: [H_{PM1AC2}, ID_{AC2}]_{Pk_{PM1}^{-1}}, \{Sct_{AC1PM2}, ID_{AC1}\}_{K_{PM2}}$;

(2-10) $AC2 \to PM2: \{Sct_{AC1PM2}, ID_{AC1}\}_{K_{PM2}}, \{Sct_{AC2PM2}, ID_{AC2}\}_{K_{PM2}}$;

(2-11) $PM2 \to AC2: [H_{PM2AC1}, ID_{AC1}]_{Pk_{PM2}^{-1}}, [H_{PM2AC2}, ID_{AC2}]_{Pk_{PM2}^{-1}}$;

(2-12) $AC2 \to AC1: [H_{PM2AC1}, ID_{AC1}]_{Pk_{PM2}^{-1}}, \{U_{AC2}, ID_{AC2}\}_{K_{PM1AC2}}$;

(2-13) $AC1 \to PM1:\{U_{AC1},ID_{AC1}\}_{K_{PM1AC1}},\{U_{AC2},ID_{AC2}\}_{K_{PM1AC2}}$;

(2-14) $PM1 \to AC1:\{RU_{PM1AC2},ID_{AC1}\}_{K_{PM1AC1}},\{RU_{PM1AC1},ID_{AC2}\}_{K_{PM1AC2}}$;

(2-15) $AC1 \to AC2:\{U_{AC1},ID_{AC1}\}_{K_{PM2AC1}},\{RU_{PM1AC1},ID_{AC2}\}_{K_{PM1AC2}}$;

(2-16) $AC2 \to PM2:\{U_{AC1},ID_{AC1}\}_{K_{PM2AC1}},\{U_{AC2},ID_{AC2}\}_{K_{PM2AC2}}$;

(2-17) $PM2 \to AC2:\{RU_{PM2AC2},ID_{AC1}\}_{K_{PM2AC1}},\{RU_{PM2AC1},ID_{AC2}\}_{K_{PM2AC2}}$;

(2-18) $AC2 \to AC1:\{RU_{PM2AC2},ID_{AC1}\}_{K_{PM2AC1}},\{I_{AC2},ID_{AC2}\}_{K_{PM1AC2}}$;

(2-19) $AC1 \to PM1:\{I_{AC1},ID_{AC1}\}_{K_{PM1AC1}},\{I_{AC2},ID_{AC2}\}_{K_{PM1AC2}}$;

(2-20) $PM1 \to AC1:\{RI_{PM1AC2},ID_{AC1},TES_{PM1}\}_{K_{PM1AC1}},\{RI_{PM1AC1},ID_{AC2}\}_{K_{PM1AC2}}$;

(2-21) $AC1 \to AC2:\{I_{AC1},ID_{AC1},TES_{PM1}\}_{K_{PM2AC1}},\{RI_{PM1AC1},ID_{AC2}\}_{K_{PM1AC2}}$;

(2-22) $AC2 \to PM2:\{I_{AC1},ID_{AC1},TES_{PM1}\}_{K_{PM2AC1}},\{I_{AC2},ID_{AC2}\}_{K_{PM2AC2}}$;

(2-23) $PM2 \to AC2:\{RI_{PM2AC2},ID_{AC1}\}_{K_{PM2AC1}},\{RI_{PM2AC1},RTES_{PM2PM1},ID_{AC2},TES_{PM2}\}_{K_{PM2AC2}}$;

(2-24) $AC2 \to AC1:\{TES_{PM2}\}_{K_{PM1AC2}},\{RI_{PM2AC2},ID_{AC1}\}_{K_{PM2AC1}}$;

(2-25) $AC1 \to PM1:\{TES_{PM2}\}_{K_{PM1AC2}}$;

(2-26) $PM1 \to AC1:\{RTES_{PM1PM2}\}_{K_{PM1AC1}}$.

步骤 1~步骤 5 省略,因为明文传输可以被伪造,对 TCA-SNI 协议安全性没有影响,所以不进行理想化描述.

- 第 3 阶段理想化描述.

(3-5) $AR1 \to AC2:\{Sct_{AR1PM2},ID_{AR1}\}_{K_{PM2}}$;

(3-6) $AC2 \to PM2:\{Sct_{AR1PM2},ID_{AR1}\}_{K_{PM2}},\{Sct_{AC1PM2},ID_{AC2}\}_{K_{PM2}}$;

(3-7) $PM2 \to AC2:[H_{PM2AR1},ID_{AR1}]_{Pk_{PM2}^{-1}},[H_{PM2AC2},ID_{AC2}]_{Pk_{PM2}^{-1}}$;

(3-8) $AC2 \to AR1:[H_{PM2AR1},ID_{AR1}]_{Pk_{PM2}^{-1}}$;

(3-9) $AR1 \to AC2:\{I_{AR1},ID_{AR1}\}_{K_{PM2AR1}}$;

(3-10) $AC2 \to PM2:\{I_{AR1},ID_{AR1}\}_{K_{PM2AR1}},\{I_{AC2},ID_{AC2}\}_{K_{PM2AC2}}$;

(3-11) $PM2 \to AC2:\{RI_{AC2},ID_{AR1}\}_{K_{PM2AR1}},\{RI_{AR1},ID_{AC2}\}_{K_{PM2AC2}}$;

(3-12) $AC2 \to AR1:\{RI_{AC2},ID_{AR1}\}_{K_{PM2AR1}}$.

步骤 1~步骤 4 省略,因为明文传输可以被伪造,对 TCA-SNI 协议安全性没有影响,所以不进行理想化描述.

3.2.5 初始假设

使用 SVO 逻辑对 TCA-SNI 协议进行推理分析,需要定义协议中每个参与者在初始时刻具有的知识和信仰,即初始假设[13].初始假设是推理证明的基础和初始状态,根据 TCA-SNI 协议流程,设定如下假设.

(1) $AR1|\equiv(PK(PM1,K_{PM1})\wedge PK(PM2,K_{PM2})),AC1|\equiv(PK(PM1,K_{PM1})\wedge PK(PM2,K_{PM2})),$
$AC2|\equiv(PK(PM1,K_{PM1})\wedge PK(PM2,K_{PM2})),PM1|\equiv PK(PM1,K_{PM1}),PM1|\equiv PK(PM2,K_{PM2}))$;

(2) $AR1|\equiv(AR1\ni Sct_{AR1PM1}\wedge AR1\ni Sct_{AR1PM2}),AC1|\equiv(AC1\ni Sct_{AC1PM1}\wedge AC1\ni Sct_{AC1PM2}),$
$AC2|\equiv(AC2\ni Sct_{AC2PM1}\wedge AC2\ni Sct_{AC2PM2})$;

(3) $AR1|\sim Rand_{AR1PM1},AR1\lhd Rand_{PM1AR1}$;

(4) $AC1|\sim Rand_{AC1PM1},AC1\lhd Rand_{PM1AC1}$;

(5) $PM1\lhd Rand_{AR1PM1},PM1|\sim Rand_{PM1AR1},PM1\lhd Rand_{AC1PM1},PM1|\sim Rand_{PM1AC1}$;

(6) $AR1\ni f_k(\cdot),AC1\ni f_k(\cdot),AC2\ni f_k(\cdot),PM1\ni f_k(\cdot),PM2\ni f_k(\cdot)$;

(7) $AR1\ni H(\cdot),AC1\ni H(\cdot),AC2\ni H(\cdot),PM1\ni H(\cdot),PM2\ni H(\cdot)$;

(8) $AC1|\sim Rand_{AC1PM1},AC1\lhd Rand_{PM1AC1}$;

(9) $AC1|\sim Rand_{AC1PM2},AC1\lhd Rand_{PM2AC1}$;

(10) $AC2|\sim Rand_{AC2PM1},AC2\lhd Rand_{PM2AC2}$;

(11) $AC2|\sim Rand_{AC2PM2}, AC2\lhd Rand_{PM2AC2}$;

(12) $PM1\lhd Rand_{AC1PM1}, PM1|\sim Rand_{PM1AC1}, PM1\lhd Rand_{AC2PM1}, PM1|\sim Rand_{PM1AC2}$;

(13) $PM2\lhd Rand_{AC1PM2}, PM2|\sim Rand_{PM2AC1}, PM2\lhd Rand_{AC2PM2}, PM2|\sim Rand_{PM2AC2}$;

(14) $AR1|\sim Rand_{AR1PM2}, AR1\lhd Rand_{PM2AR1}$;

(15) $AC2|\sim Rand_{AC2PM2}, AC2\lhd Rand_{PM2AC2}$;

(16) $PM2\lhd Rand_{AR1PM2}, PM2|\sim Rand_{PM2AR1}, PM2\lhd Rand_{AC2PM2}, PM2|\sim Rand_{PM2AC2}$.

3.2.6 协议目标

由于 TCA-SNI 协议存在可信第三方 *PM*,因此对于 TCA-SNI 协议要达到的目标,主要是协议主体 *AR* 和 *AC* 获得 *PM* 的可信认证结果,必要条件是各方相信彼此通信是有着良好的通信密钥来保障通信安全[13].TCA-SNI 协议目标分阶段用 SVO 逻辑语言表达如下.

- 第 1 阶段

(1) $AR1|\equiv AR1\xleftrightarrow{K_{PM1AR1}}PM1$;

(2) $AC1|\equiv AC1\xleftrightarrow{K_{PM1AC1}}PM1$;

(3) $AR1|\equiv(PM1|\equiv U_{AC1}\wedge PM1|\equiv I_{AC1})$;

(4) $AC1|\equiv(PM1|\equiv U_{AR1}\wedge PM1|\equiv I_{AR1})$.

- 第 2 阶段

(5) $AC1|\equiv AC1\xleftrightarrow{K_{PM1AC1}}PM1$;

(6) $AC1|\equiv AC1\xleftrightarrow{K_{PM2AC1}}PM2$;

(7) $AC2|\equiv AC2\xleftrightarrow{K_{PM1AC2}}PM1$;

(8) $AC2|\equiv AC2\xleftrightarrow{K_{PM2AC2}}PM2$;

(9) $AC1|\equiv(PM1|\equiv(U_{AC2}\wedge I_{AC2}\wedge TES_{PM2}))$;

(10) $AC1|\equiv(PM2|\equiv(U_{AC2}\wedge I_{AC2}))$;

(11) $AC2|\equiv(PM1|\equiv(U_{AC1}\wedge I_{AC1}))$;

(12) $AC2|\equiv(PM2|\equiv(U_{AC1}\wedge I_{AC1}\wedge TES_{PM1}))$.

- 第 3 阶段

(13) $AR1|\equiv AR1\xleftrightarrow{K_{PM2AR1}}PM2$;

(14) $AC2|\equiv AC2\xleftrightarrow{K_{PM2AC2}}PM2$;

(15) $AR1|\equiv(PM2|\equiv I_{AC2})$;

(16) $AC2|\equiv(PM2|\equiv I_{AR1})$.

3.2.7 协议分析

依据初始假设,使用 SVO 逻辑系统进行逻辑推理,验证 TCA-SNI 协议的安全性[13].

- 第 1 阶段验证

由步骤(1-5)、步骤(1-6)、初始假设(1)、公理 A3 可得:

R1:$PM1|\equiv(PM1\lhd Sct_{AR1PM1})\wedge(PM1\lhd Sct_{AC1PM1})$.

由 R1、初始假设(5)、初始假设(7)可得:

R2:$PM1\ni H_{PM1AR1}, PM1\ni H_{PM1AC1}$.

由 R1、初始假设(5)、初始假设(6)、公理 A4 可得:

R3: $PM1|\equiv(PM1\ni PM1\xleftrightarrow{K_{PM1AR1}}AR1), PM1|\equiv(PM1\ni PM1\xleftrightarrow{K_{PM1AC1}}AC1)$.

由步骤(1-7)、步骤(1-8)、初始假设(2)~初始假设(4)、初始假设(6)、初始假设(7)可得:

R4: $AR1\lhd[H_{PM1AR1}, ID_{AR1}]_{Pk_{PM1}^{-1}}, AC1\lhd[H_{PM1AC1}, ID_{AC1}]_{Pk_{PM1}^{-1}}$;

R5:$AC1\ni H(Rand_{AC1PM1}, Rand_{PM1AC1}, Sct_{AC1PM1}, ID_{AC1}), AR1\ni H(Rand_{AR1PM1}, Rand_{PM1AR1}, Sct_{AR1PM1}, ID_{AR1})$;

R6: $AC1 \ni PM1 \xleftrightarrow{K_{PM1AC1}} AC1, AR1 \ni PM1 \xleftrightarrow{K_{PM1AR1}} AR1$.

由 R4、R5、初始假设(1)、初始假设(6)、公理 A2、公理 A3、扩展公理 A6 可得:

R7: $AC1 \mid\equiv PM1 \lhd (Rand_{AC1PM1}, Rand_{PM1AC1}, Sct_{AC1PM1}), AC1 \mid\equiv (PM1 \xleftrightarrow{K_{PM1AC1}} AC1)$;

R8: $AR1 \mid\equiv PM1 \lhd (Rand_{AR1PM1}, Rand_{PM1AR1}, Sct_{AR1PM1}), AR1 \mid\equiv (PM1 \xleftrightarrow{K_{PM1AR1}} AR1)$.

由步骤(1-9)、步骤(1-10)、规则 R3、公理 A3 可得:

R9:$PM1 \mid\equiv (PM1 \lhd U_{AR1}) \wedge (PM1 \lhd U_{AC1}) \wedge (AR1 \mid\sim U_{AR1}) \wedge (AC1 \mid\sim U_{AC1})$.

由步骤(1-11)、步骤(1-12)、规则 R6、公理 A3 可得:

R10:$AC1 \lhd RU_{AR1}, AR1 \lhd RU_{AC1}, AC1 \mid\equiv (PM1 \mid\equiv U_{AR1}), AR1 \mid\equiv (PM1 \mid\equiv U_{AC1})$.

由步骤(1-13)、步骤(1-14)、规则 R3、公理 A3 可得:

R11:$PM1 \mid\equiv (PM1 \lhd I_{AR1}) \wedge (PM1 \lhd I_{AC1}) \wedge (AR1 \mid\sim I_{AR1}) \wedge (AC1 \mid\sim I_{AC1})$.

由步骤(1-15)、步骤(1-16)、规则 R6、公理 A3 可得:

R12:$AC1 \lhd RI_{AR1}, AR1 \lhd RI_{AC1}, AC1 \mid\equiv (PM1 \mid\equiv I_{AR1}), AR1 \mid\equiv (PM1 \mid\equiv I_{AC1})$.

由规则 R7、规则 R8、规则 R10、规则 R12、公理 A1、公理 A5 可证得协议目标(1)~协议目标(4).

- 第 2 阶段验证

由步骤(2-6)、步骤(2-7)、初始假设(1)、公理 A3 可得:

R13:$PM1 \mid\equiv (PM1 \lhd Sct_{AC1PM1}) \wedge (PM1 \lhd Sct_{AC2PM1})$.

由规则 R13、初始假设(7)、初始假设(12)可得:

R14:$PM1 \ni H_{PM1AC1}, PM1 \ni H_{PM1AC2}$.

由规则 R13、初始假设(6)、初始假设(12)、公理 A4 可得:

R15: $PM1 \mid\equiv (PM1 \ni PM1 \xleftrightarrow{K_{PM1AC1}} AC1), PM1 \mid\equiv (PM1 \ni PM1 \xleftrightarrow{K_{PM1AC2}} AC2)$.

由步骤(2-8)、步骤(2-9)、初始假设(2)、初始假设(6)~初始假设(8)、初始假设(10)可得:

R16: $AC1 \lhd [H_{PM1AC1}, ID_{AC1}]_{Pk_{PM1}^{-1}}, AC2 \lhd [H_{PM1AC2}, ID_{AC2}]_{Pk_{PM1}^{-1}}$;

R17:$AC1 \ni H(Rand_{AC1PM1}, Rand_{PM1AC1}, Sct_{AC1PM1}, ID_{AC1}), AC2 \ni H(Rand_{AC2PM1}, Rand_{PM1AC2}, Sct_{AC2PM1}, ID_{AC2})$;

R18: $AC1 \ni (PM1 \xleftrightarrow{K_{PM1AC1}} AC1), AC2 \ni (PM1 \xleftrightarrow{K_{PM1AC2}} AC2)$.

由规则 R16、规则 R17、初始假设(1)、初始假设(6)、公理 A2、公理 A3、扩展公理 A6 可得:

R19: $AC1 \mid\equiv PM1 \lhd (Rand_{AC1PM1}, Rand_{PM1AC1}, Sct_{AC1PM1}), AC1 \mid\equiv (PM1 \xleftrightarrow{K_{PM1AC1}} AC1)$;

R20: $AC2 \mid\equiv PM1 \lhd (Rand_{AC2PM1}, Rand_{PM1AC2}, Sct_{AC2PM1}), AC2 \mid\equiv (PM1 \xleftrightarrow{K_{PM1AC2}} AC2)$.

由步骤(2-9)、步骤(2-10)、初始假设(1)、公理 A3 可得:

R21:$PM2 \mid\equiv (PM2 \lhd Sct_{AC1PM2}) \wedge (PM2 \lhd Sct_{AC2PM2})$.

由规则 R21、初始假设(7)、初始假设(13)可得:

R22:$PM2 \ni H_{PM2AC1}, PM2 \ni H_{PM2AC2}$.

由规则 R21、初始假设(6)、初始假设(13)、公理 A4 可得:

R23: $PM2 \mid\equiv (PM2 \ni PM2 \xleftrightarrow{K_{PM2AC1}} AC1), PM2 \mid\equiv (PM2 \ni PM2 \xleftrightarrow{K_{PM2AC2}} AC2)$.

由步骤(2-11)、步骤(2-12)、初始假设(2)、初始假设(6)、初始假设(7)、初始假设(9)、初始假设(11)可得:

R24: $AC1 \lhd [H_{PM2AC1}, ID_{AC1}]_{Pk_{PM2}^{-1}}, AC2 \lhd [H_{PM2AC2}, ID_{AC2}]_{Pk_{PM2}^{-1}}$;

R25:$AC1 \ni H(Rand_{AC1PM2}, Rand_{PM2AC1}, Sct_{AC1PM2}, ID_{AC1}), AC2 \ni H(Rand_{AC2PM2}, Rand_{PM2AC2}, Sct_{AC2PM2}, ID_{AC2})$;

R26: $AC1 \ni (PM2 \xleftrightarrow{K_{PM2AC1}} AC1), AC2 \ni (PM2 \xleftrightarrow{K_{PM2AC2}} AC2)$.

由规则 R24、规则 R25、初始假设(1)、初始假设(6)、公理 A2、公理 A3、扩展公理 A6 可得:

R27: $AC1 \mid\equiv PM2 \lhd (Rand_{AC1PM2}, Rand_{PM2AC1}, Sct_{AC1PM2}), AC1 \mid\equiv (PM2 \xleftrightarrow{K_{PM2AC1}} AC1)$;

R28: $AC2 \mid\equiv PM2 \lhd (Rand_{AC2PM2}, Rand_{PM2AC2}, Sct_{AC2PM2}), AC2 \mid\equiv (PM2 \xleftrightarrow{K_{PM2AC2}} AC2)$.

由步骤(2-12)、步骤(2-13)、规则 R15、公理 A3 可得:

R29:$PM1|\equiv(PM1\lhd U_{AC1})\wedge(PM1\lhd U_{AC2})\wedge(AC1|\sim U_{AC1})\wedge(AC2|\sim U_{AC2})$.

由步骤(2-14)、步骤(2-15)、规则 R19、规则 R20、公理 A3 可得:

R30:$AC1\lhd RU_{PM1AC1},AC2\lhd RU_{PM1AC1},AC1|\equiv(PM1|\equiv U_{AC2}),AC2|\equiv(PM1|\equiv U_{AC1})$.

由步骤(2-15)、步骤(2-16)、规则 R23、公理 A3 可得:

R31:$PM2|\equiv(PM2\lhd U_{AC1})\wedge(PM2\lhd U_{AC2})\wedge(AC1|\sim U_{AC1})\wedge(AC2|\sim U_{AC2})$.

由步骤(2-17)、步骤(2-18)、规则 R27、规则 R28、公理 A3 可得:

R32:$AC1\lhd RU_{PM2AC2},AC2\lhd RU_{PM2AC1},AC1|\equiv(PM2|\equiv U_{AC2}),AC2|\equiv(PM2|\equiv U_{AC1})$.

由步骤(2-18)、步骤(2-19)、规则 R15、公理 A3 可得:

R33:$PM1|\equiv(PM1\lhd I_{AC1})\wedge(PM1\lhd I_{AC2})\wedge(AC1|\sim I_{AC1})\wedge(AC2|\sim I_{AC2})$.

由步骤(2-20)、步骤(2-21)、规则 R19、规则 R20、公理 A3 可得:

R34:$AC1\lhd RI_{AC2},AC2\lhd RI_{AC1},AC1|\equiv(PM1|\equiv I_{AC2}),AC2|\equiv(PM1|\equiv I_{AC1})$.

由步骤(2-20)~步骤(2-22)、规则 R23、公理 A3 可得:

R35:$PM2|\equiv(PM2\lhd I_{AC1})\wedge(PM2\lhd I_{AC2})\wedge(PM2\lhd TES_{PM1})\wedge(AC1|\sim I_{AC1})\wedge(AC2|\sim I_{AC2})\wedge(PM1|\sim TES_{PM1})$.

由步骤(2-23)、步骤(2-24)、规则 R27、规则 R28、公理 A3 可得:

R36:$AC1\lhd RI_{PM2AC2},AC2\lhd RI_{PM2AC1},AC2\lhd RTES_{PM2PM1},AC1|\equiv(PM2|\equiv I_{AC2})$,
$AC2|\equiv(PM2|\equiv I_{AC1}),AC2|\equiv(PM2|\equiv TES_{PM1})$.

由步骤(2-23)~步骤(2-25)、规则 R15、公理 A3 可得:

R37:$PM1|\equiv(PM1\lhd TES_{PM2})\wedge(PM2|\sim TES_{PM2})$.

由步骤(2-26)、规则 R19、公理 A3 可得:

R38:$AC1\lhd RTES_{PM1PM2},AC1|\equiv(PM1|\equiv TES_{PM2})$.

由规则 R19、规则 R20、规则 R27、规则 R28、公理 A1、公理 A5 可证得协议目标(5)~协议目标(8);由规则 R30、规则 R32、规则 R36、规则 R38、公理 A1、公理 A5 可证得协议目标(9)~协议目标(12).

- 第 3 阶段验证

由步骤(3-5)、步骤(3-6)、初始假设(1)、公理 A3 可得:

R39:$PM2|\equiv(PM2\lhd Sct_{AR1PM2})\wedge(PM2\lhd Sct_{AC2PM2})$.

由规则 R39、初始假设(7)、初始假设(16)得:

R40:$PM2\ni H_{PM2AR1},PM2\ni H_{PM2AC2}$.

由规则 R39、初始假设(6)、初始假设(16)、公理 A4 可得:

R41: $PM2|\equiv(PM2\ni PM2\xleftrightarrow{K_{PM2AR1}}AR1),PM2|\equiv(PM2\ni PM2\xleftrightarrow{K_{PM2AC2}}AC2)$.

由步骤(3-7)、步骤(3-8)、初始假设(2)、初始假设(6)、初始假设(7)、初始假设(14)、初始假设(15)可得:

R42: $AR1\lhd[H_{PM2AR1},ID_{AR1}]_{Pk_{PM2}^{-1}},AC2\lhd[H_{PM2AC2},ID_{AC2}]_{Pk_{PM2}^{-1}}$;

R43:$AC2\ni H(Rand_{AC2PM2},Rand_{PM2AC2},Sct_{AC2PM2},ID_{AC2}),AC1\ni H(Rand_{AR1PM2},Rand_{PM2AR1},Sct_{AR1PM2},ID_{AR1})$;

R44: $AC2\ni PM2\xleftrightarrow{K_{PM2AC2}}AC2,AR1\ni PM2\xleftrightarrow{K_{PM2AR1}}AR1$.

由规则 R42、规则 R43、初始假设(1)、初始假设(6)、公理 A2、公理 A3、扩展公理 A6 可得:

R45: $AR1|\equiv PM2\lhd(Rand_{AR1PM2},Rand_{PM2AR1},Sct_{AR1PM2}),AR1|\equiv(PM2\xleftrightarrow{K_{PM2AR1}}AR1)$;

R46: $AC2|\equiv PM2\lhd(Rand_{AC2PM2},Rand_{PM2AC2},Sct_{AC2PM2}),AC2|\equiv(PM2\xleftrightarrow{K_{PM2AC2}}AC2)$.

由步骤(3-9)、步骤(3-10)、规则 R41、公理 A3 可得:

R47:$PM2|\equiv(PM2\lhd I_{AR1})\wedge(PM2\lhd I_{AC2})\wedge(AR1|\sim I_{AR1})\wedge(AC2|\sim I_{AC2})$.

由步骤(3-11)、步骤(3-12)、规则 R44、公理 A3 可得:

R48:$AC2\lhd RI_{AR1},AR1\lhd RI_{AC2},AC2|\equiv(PM2|\equiv I_{AR1}),AR1|\equiv(PM2|\equiv I_{AC2})$.

由规则 R45、规则 R46、规则 R48、公理 A1、公理 A3 可证得协议目标(13)~协议目标(16).

至此,协议目标以全部证出.从上述 SVO 逻辑分析可以得出,TCA-SNI 协议达到了网络间可信认证的目标,具有较高的安全性.

4 安全性测试及分析

4.1 AVISPA工具

我们选择 Dolev-Yao 攻击者模型[27−30]测试 TCA-SNI 协议.在 Dolev-Yao 模型中,入侵者完全控制通信信道,协议中消息的传递必须经过入侵者,入侵者可以非常容易地窃听、分析、拦截和修改任何消息,可以合法地参与协议的运行,并向任何人发送伪造的消息.

AVISPA(automated validation of internet security protocols and applications)[31−34]是一套著名的建立和分析安全协议模型工具,融合了 4 种不同的分析工具:动态模型检验期(OFMC)、基于逻辑约束的攻击搜索器(CL-AtSe)、基于 SAT 的模型检验器(SATMC)、基于自动逼近的树自动机安全协议分析(TA4SP).AVISPA 架构如图 4 所示.

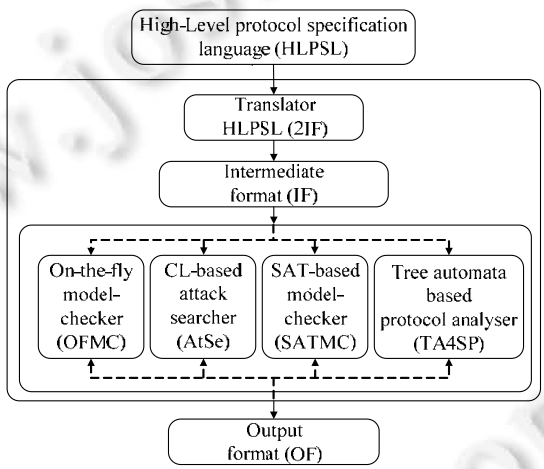

Fig.4 AVISPA architecture

图 4 AVISPA 架构

AVISPA 工具采用 HLPSL(high level protocol specification language)[35−37]语言对协议建立分析模型,通过 HLPSL2IF 工具转化为 IF(intermediate format)语言,在 Dolev-Yao 安全模型下进行分析.AVISPA 工具可以直接读取 IF 语言,分析协议是否满足安全目标.如果协议不安全,分析工具将会显示导致不安全事件发生的攻击轨迹.

4.2 实验过程

4.2.1 基本角色(role)

HLPSL 是基于角色的形式化语言.协议中每个参与者分别定义为一个角色,见表 1,分别建立访问请求者($ar1$)、访问控制器 1($ac1$)、策略管理器 1($pm1$)、访问控制器 2($ac2$)、策略管理器 2($pm2$).

Table 1 Definition of the basic roles

表 1 基本角色定义

基本角色	定义
*ar*1	role *ar*1:(*AR*1,*AC*1,*PM*1,*PM*2:*agent*,*HH*,*HK*:*hash_func*,*SND*,*RCV*:*channel*(*dy*))
*ac*1	role *ac*1:(*AR*1,*PM*1,*AC*1,*AC*2,*PM*2:*agent*,*HH*,*HK*:*hash_func*,*SNDPM*,*RCVPM*,*SNDAC*,*RCVAC*:*channel*(*dy*))
*pm*1	role *pm*1:(*AR*1,*PM*1,*AC*1,*AC*2,*PM*2:*agent*,*HH*,*HK*:*hash_func*,*SND*,*RCV*:*channel*(*dy*))
*ac*2	role *ac*2:(*AR*1,*PM*1,*AC*1,*AC*2,*PM*2:*agent*,*HH*,*HK*:*hash_func*,*SNDPM*,*RCVPM*,*SNDAC*,*RCVAC*:*channel*(*dy*))
*pm*2	role *pm*2:(*AR*1,*PM*1,*AC*1,*AC*2,*PM*2:*agent*,*HH*,*HK*:*hash_func*,*SND*,*RCV*:*channel*(*dy*))

4.2.2 会话场景

我们定义了 4 种会话场景来验证 TCA-SNI 协议是否符合安全目标.首先,我们定义了一个正常的会话过程,其中包含所有合法的角色(场景 1).然后,定义了入侵者伪装访问请求者(场景 2)、访问控制器 1(场景 3)或访问控制器 2(场景 4)的情况.上述场景的定义见表 2,其中,*hh*,*hk* 表示不同的散列算法.

Table 2 Summary of session configurations

表 2 场景定义

场景	定义
场景 1	*session*(*ar*1,*pm*1,*ac*1,*ac*2,*pm*2,*hh*,*hk*)
场景 2	*session*(*i*,*pm*1,*ac*1,*ac*2,*pm*2,*hh*,*hk*)
场景 3	*session*(*ar*1,*pm*1,*i*,*ac*2,*pm*2,*hh*,*hk*)
场景 4	*session*(*ar*1,*pm*1,*ac*1,*i*,*pm*2,*hh*,*hk*)

4.2.3 安全目标

为了评估 TCA-SNI 协议的安全性,我们需要确定协议需要达成的安全目标.AVISPA 提供了不同的关键字表示安全目标[13].在本文实验中,关键字描述如下.

(1) 秘密检测.消息 *T* 是代理 *A* 产生的,且是代理 *A* 与代理 *B* 之间的秘密.如下所示:

$$secret(T,t,\{A,B\}).$$

其中,*t* 是在定义安全目标时使用的标识符.

(2) 强认证检测.如下所示,*request* 关键字声明代理 *B* 确实收到了来自代理 *A* 的消息 *T*,*witness* 关键字声明代理 *A* 向代理 *B* 发送了消息 *T*.

$$request(A,B,t,T),witness(B,A,t,T),$$

其中,*t* 是在定义安全目标时使用的标识符.

为了评估 TCA-SNI 协议安全性,我们定义了如下的安全目标.

(1) 本协议根据通信双方沟通的随机数,通过散列算法 *hk* 生成通信密钥,只需保证通信双方掌握的随机数一致即可.因此,需要验证随机数使用散列算法 *hh* 计算出的散列值是否一致,并且验证使用公钥加密传输的随机数是否是保密的.HLPSL 描述如下:

*ar*1:*secret*(*sct*1,*Spm*1*ar*1′,{*AR*1,*PM*1});*request*(*AR*1,*PM*1,*hpm*1*ar*1,*Hpm*1*ar*1′);
secret(*sct*6,*Spm*2*ar*1′,{*AR*1,*PM*2});*request*(*AR*1,*PM*2,*hpm*2*ar*1,*Hpm*2*ar*1′).
*ac*1:*secret*(*sct*2,*Spm*1*ac*1′,{*AC*1,*PM*1});*request*(*AC*1,*PM*1,*hpm*1*ac*1,*Hpm*1*ac*1′);
secret(*sct*5,*Spm*2*ac*1′,{*AC*1,*PM*2});*request*(*AC*1,*PM*2,*hpm*2*ac*1,*Hpm*2*ac*1′).
*ac*2:*secret*(*sct*3,*Spm*1*ac*2′,{*AC*2,*PM*1});*request*(*AC*2,*PM*1,*hpm*1*ac*2,*Hpm*1*ac*2′);
secret(*sct*4,*Spm*2*ac*2′,{*AC*2,*PM*2});*request*(*AC*2,*PM*2,*hpm*2*ac*2,*Hpm*2*ac*2′).

(2) 本协议由策略管理器进行身份认证及平台完整性评估,返回对应的访问决策,因此需验证访问决策被成功地接收,不被攻击.HLPSL 描述如下:

*ar*1:*request*(*AR*1,*PM*1,*rpm*1*ar*1,*Rac*1);*request*(*AR*1,*PM*2,*rpm*2*ar*1,*Rac*2).
*ac*1:*request*(*AC*1,*PM*1,*rpm*1*ac*1,*Rpm*1*ac*2);*request*(*AC*1,*PM*2,*rpm*2*ac*1,*Rpm*2*ac*2);
request(*AC*1,*PM*1,*rtespm*1*ac*1,*RTESpm*1*pm*2).
*ac*2:*request*(*AC*2,*PM*2,*rpm*2*ac*2,*Rpm*2*ac*1);*request*(*AC*2,*PM*1,*rpm*1*ac*2,*Rpm*1*ac*1);
request(*AC*2,*PM*2,*rtespm*2*ac*2,*RTESpm*2*pm*1).

4.2.4 实验结果

本文使用 HLPSL 语言对 TCA-SNI 协议过程进行了描述,并开源了 HLPSL 文件:https://git.io/vhvsR.

将 HLPSL 模型导入 AVISPA 工具,使用 OFMC 和 CL-AtSe 分析工具搜索攻击.实验结果见表 3.

根据表 3 的结果,OFMC 和 AtSe 分析工具对 TCA-SNI 协议的分析结果是安全的(SUMMARY: SAFE),而且

没有发现协议缺陷.如果检测到协议缺陷,“SUMMARY”字段会显示“UNSAFE”,并在“DETAILS”字段提示“ATTACK_FOUND”.分析过程中,由 HLPSL 描述转换为 IF 形式描述保存在“PROTOCOL”字段给出的路径中文件名为“13153928339581249 8.if”的文件中.结果中的“BACKEND”字段给出所用的后端分析工具类型,“STATISTICS”字段显示了分析工具所执行的时间及搜索的节点数或状态数量.

Table 3 Summary of test outputs

表 3 测试结果摘要

方法	结果摘要	方法	结果摘要
OFMC	SUMMARY SAFE DETAILS BOUNDED_NUMBER_OF_SESSIONS PROTOCOL C:\NPLab\temp\131539067826346508.if GOAL as_specified BACKEND OFMC COMMENTS STATISTICS parseTime:0.00s searchTime:−73.21s visitedNodes:238769 nodes depth:6 plies	CL-AtSe	SUMMARY SAFE DETAILS BOUNDED_NUMBER_OF_SESSIONS TYPED_MODEL PROTOCOL C:\NPLab\temp\131539283395812498.if GOAL As Specified BACKEND CL-AtSe STATISTICS Analysed:25966485states Reachable:4987948states Translation:0.32seconds Computation:979.76seconds

5 总 结

在促进大中小企业融通发展、深入推进“互联网+”的进程中,对网络的可信互连提出了更高的要求.本文基于可信连接架构的思想,提出了一种支持网络间互连的可信连接协议(TCA-SNI).本文分析了可信连接架构,对该架构无法适用于网络间可信认证的不足进行了扩展.以 TPCM 芯片为可信基础,对终端进行动态可信评估,达到局域网内部可信的状态.根据网络安全策略不同的特点,提出了双向四元可信认证架构,通过策略管理器的协同工作,网络间构成可信通信环境,可进行基础操作.若外部终端需要向网络内发送高级操作命令,则需要进行终端深度认证,保障终端在进行高级操作时可信.

本文对 TCA-SNI 协议的工作流程进行了详细的说明.TCA-SNI 协议需要通过终端可信入网、网间可信连接、终端深度认证这 3 个阶段,达到网络可信互连的目标.可信评估过程主要由通信密钥协商、身份认证、平台完整性评估组成.基于安全考虑,发起连接请求的终端不能与策略管理器直接通信,需要由访问控制器进行转发.为避免数据包遭到篡改或丢弃,TCA-SNI 采用了确认机制,并对逻辑通信通道进行加密.在网间可信连接阶段后期,策略管理器对网络的安全策略进行互相评估,在安全策略不同的情况下,使得网间通信环境可信.

本文针对 TCA-SNI 协议的安全性进行了分析.使用扩展 SVO 逻辑对 TCA-SNI 协议进行形式化描述,提出了协议安全目标,使用推理规则和公理进行逻辑分析,证明此协议理论上有较高的安全性.随后,对 TCA-SNI 协议进行了攻击测试.使用 HLPSL 描述语言对 TCA-SNI 进行建模,建立基本角色,定义协议工作场景并设定安全目标,导入 AVISPA 安全协议分析工具,使用 Dolev-Yao 攻击者模型进行攻击测试.测试结果表明,TCA-SNI 不存在协议缺陷,证明此协议可以抵御真实网络中的攻击.

References:

[1] Zhang DW, Shen CX, Liu JQ, Zhang FF, Li L, Cheng LC. TC assurance architecture for cybersecurity infrastructure based on active defense. Strategic Study of Chinese Academy of Engineering, 2016,18(6):58−61 (in Chinese with English abstract).

[2] Shen CX, Zhang DW, Liu JQ, Ye H, Qiu S. The strategy of TC 3.0: A revolutionary evolution in trusted computing. Strategic Study of Chinese Academy of Engineering, 2016,18(6):53−57 (in Chinese with English abstract).

[3] Shen CX. Building cyber security defense by trusted computing 3.0. Information and Communications Technologies, 2017,3(3): 290−298 (in Chinese with English abstract).

[4] Chen Z, Deng FC, Luo AA, Jiang X, Li GD, Zhang RH, Lin C. Application level network access control system based on TNC architecture for enterprise network. In: Proc. of the 2010 IEEE Int'l Conf. on Wireless Communications, Networking and Information Security (WCNIS). IEEE, 2010. 667−671.

[5] Wu K, Bai ZY. A clientless endpoint authentication scheme based on TNC. IJ Information Technology and Computer Science, 2010,2:9−16.

[6] Rehbock S, Hunt R. Trustworthy clients: Extending TNC to Web-based environments. Computer Communications, 2009,32(5): 1006−1013.

[7] Zhao B, Zhu XY, Xiang S, Yang B. C-TNC: trusted cloud access protocol for Openstack. Journal of Huazhong University of Science and Technology, 2016,44(3):83−88 (in Chinese with English abstract).

[8] Song SY. TNC improvement based on bidirectional authentication and enhanced IMC. Communications Technology, 2017,50(8): 1776−1783 (in Chinese with English abstract).

[9] Liu YL, Jin ZG. SAEW: A security assessment and enhancement system of wireless local area networks (WLANs). Wireless Personal Communications, 2015,82(1):1−19.

[10] Ye M, Luo WB. Application of trusted network connect (TNC) architecture. Information Security and Communications Privacy, 2006,1:58−60 (in Chinese with English abstract).

[11] Zhou Y. Improved mechanism for trusted network connect. Journal of Computer Applications, 2014(s2):99−101 (in Chinese with English abstract).

[12] Li M, Li Q, Zhang GQ, Yan X. The implementation and application of trusted connect architecture. Journal of Information Security Research, 2017,3(4):332−338 (in Chinese with English abstract).

[13] Liu Y. Research on trusted connection protocol between networks based on trusted connect architecture [MS. Thesis]. Beijing: Beijing University of Technology, 2018.

[14] Syverson PF, Oorschot PCV. On unifying some cryptographic protocol logics. In: Proc. of the 1994 IEEE Computer Society Symp. on Research in Security and Privacy. IEEE, 1994. 14−28.

[15] You I, Hori Y, Sakurai K. Enhancing SVO logic for mobile IPv6 security protocols. Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications, 2011,2(3):26−52.

[16] Chen L, Shi MX. Security analysis and improvement of Yahalom protocol. In: Proc. of the 3rd IEEE Conf. on Industrial Electronics and Applications (ICIEA 2008). IEEE, 2008. 1137−1140.

[17] Burrows M, Abadi M, Needham R. A logic of authentication. ACM Sigops Operating Systems Review, 1989,23(5):1−13.

[18] Gong L, Needham R, Yahalom R. Reasoning about belief in cryptographic protocols. In: Proc. of the 1990 IEEE Computer Society Symp. on Research in Security and Privacy. IEEE, 1990. 234−248.

[19] Paul CVO. Extending cryptographic logics of belief to key agreement protocols. In: Proc. of the 1st ACM Conf. on Computer and Communications Security. ACM Press, 1993. 232−243.

[20] Bodei C, Buchholtz M, Degano P, Nielson F, Nielson HR. Static validation of security protocols. Journal of Computer Security, 2005,13(3):347−390.

[21] Wu KG, Chen M. Improving SVO logic. Journal of Harbin Engineering University, 2007,28(5):542−547 (in Chinese with English abstract).

[22] Han JH, Guo YB, Wang YD. On methods and techniques for formal analysis of security protocols. Journal of Information Engineering University, 2008,9(3):272−276 (in Chinese with English abstract).

[23] Thakur T, Dogra S, Sood Y. A review and comparative analysis of modal logics: BAN, GYN and SVO. Int'l Journal of Research and Engineering, 2015,4(5):1197−1201.

[24] Imamoto K, Sakurai K. Design and analysis of diffie-hellman-based key exchange using one-time ID by SVO logic. Electronic Notes in Theoretical Computer Science, 2005,135(1):79−94.

[25] Xiao YY, Su KL. Verification of e-commerce payment protocol authentication properties based on SVO logic. Computer Engineering and Applications, 2014,50(8):6−10 (in Chinese with English abstract).

[26] Li BT, Luo JZ. Formal analysis of timeliness in non-repudiation protocols. Ruan Jian Xue Bao/Journal of Software, 2006,17(7): 1510−1516 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/17/1510.htm [doi: 10.1360/jos171510]

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

[28] Basagiannis S, Katsaros P, Pombortsis A. An intruder model with message inspection for model checking security protocols. Computers and Security, 2010,29(1):16−34.

[29] Debar H, Viinikka J. Intrusion detection: Introduction to intrusion detection and security information management. In: Proc. of the FOSAD 2004/2005. LNCS 3655, Berlin, Heidelburg: Springer-Verlag, 2005. 207−236.

[30] Xue R, Feng DG. The approaches and technologies for formal verification of security protocols. Chinese Journal of Computers, 2006,29(1):1−20 (in Chinese with English abstract).

[31] Hurtado Alegría JA, Bastarrica MC, Bergel A. Avispa: A tool for analyzing software process models. Journal of Software Maintenance & Evolution Research & Practice, 2014,26(4):434−450.

[32] Viganò L. Automated security protocol analysis with the AVISPA tool. Electronic Notes in Theoretical Computer Science, 2006, 155:61−86.

[33] Ocenasek P, Sveda M. AVISPA: Towards practical verification of communication properties. IFAC Proceedings Volumes, 2009, 42(1):153−156.

[34] Lai YX, Chen YN, Zou QC, Liu ZH, Yang Z. Design and analysis on trusted network equipment access authentication protocol. Simulation Modelling Practice and Theory, 2015,51:157−169.

[35] Armando A, Basin D, Boichut Y, Chevalier Y, Compagna L, Cuéllar J, Drielsma PH, Héam PC, Kouchnarenko O, Mantovani J, Mödersheim S, Oheimb DV, Rusinowitch M, Santiago J, Turuani M, Vigano L, Vigneron L. The AVISPA tool for the automated validation of internet security protocols and applications. Computer Aided Verification, 2005, 281−285.

[36] Dadeau F, Héam PC, Kheddam R, Maatoug G, Rusinowitch M. Model-Based mutation testing from security protocols in HLPSL. Software Testing Verification & Reliability, 2015,25(5-7):684−711.

[37] Benerecetti M, Cuomo N, Peron A. TPMC: A model checker for time-sensitive security protocols. Journal of Computers, 2009,4(5): 366−377.

附中文参考文献:

[1] 张大伟,沈昌祥,刘吉强,张飞飞,李论,程丽辰.基于主动防御的网络安全基础设施可信技术保障体系.中国工程科学,2016,18(6): 58−61.

[2] 沈昌祥,张大伟,刘吉强,叶珩,邱硕.可信 3.0 战略:可信计算的革命性演变.中国工程科学,2016,18(6):53−57.

[3] 沈昌祥.用可信计算 3.0 筑牢网络安全防线.信息通信技术,2017,3(3):290−298.

[7] 赵波,朱向玉,向骕,杨冰.C-TNC:适用于 Openstack 的可信云接入协议.华中科技大学学报,2016,44(3):83−88.

[8] 宋生宇.基于双向认证和增强型 IMC 对 TNC 的改进.通信技术,2017,50(8):1776−1783.

[10] 叶茂,罗万伯.TNC 架构的应用研究.信息安全与通信保密,2006,1:58−60.

[11] 周勇.改进的可信网络连接机制.计算机应用,2014(s2):99−101.

[12] 李明,李琴,张国强,颜湘.可信网络连接架构 TCA 的实现及其应用.信息安全研究,2017,3(4):332−338.

[13] 刘岩.基于 TCA 的网络间可信连接协议的研究[硕士学位论文].北京:北京工业大学,2018.

[21] 吴开贵,陈明.对 SVO 逻辑方法的改进.哈尔滨工程大学学报,2007,28(5):542−547.

[22] 韩继红,郭渊博,王亚弟.安全协议形式化分析方法.信息工程大学学报,2008,9(3):272−276.

[25] 肖茵茵,苏开乐.电子商务支付协议认证性的 SVO 逻辑验证.计算机工程与应用,2014,50(8):6−10.

[26] 黎波涛,罗军舟.不可否认协议时限性的形式化分析.软件学报,2006,17(7):1510−1516. http://www.jos.org.cn/1000-9825/17/1510.htm [doi: 10.1360/jos171510]

[30] 薛锐,冯登国.安全协议的形式化分析技术与方法.计算机学报,2006,29(1):1−20.

赖英旭(1973－),女,辽宁抚顺人,博士,教授,主要研究领域为网络接入控制,病毒防御技术,网络安全工程,可信计算理论及其应用.

刘静(1978－),女,博士,助理研究员,CCF 专业会员,主要研究领域为安全接入,可信计算,网络入侵检测.

刘岩(1992－),男,硕士,主要研究领域为安全接入,可信计算.