软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2021,32(9):2849−2866 [doi: 10.13328/j.cnki.jos.005973]

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

强安全模型下 TLS1.3 协议的形式化分析与优化*

陆思奇[1,2], 周思渊[1], 毛 颖[2,3]

[1](解放军信息工程大学,河南 郑州 450001)

[2](信息安全国家重点实验室(中国科学院 信息工程研究所),北京 100093)

[3](中国科学院大学 网络空间安全学院,北京 100093)

通讯作者: 毛颖, E-mail: maoying@iie.ac.cn

摘 要: TLS 协议在 TCP/IP 体系中的传输层和应用层之间工作,通过提供机密性、完整性、必选的服务器认证以及可选的客户端认证等一系列安全服务,有效保护了传输层的安全.TLS1.3 协议为了降低网络延迟,增加了对 0-RTT 数据的支持,通过客户端缓存服务器的长期公钥,在第 1 条消息中,直接利用该长期公钥生成一个会话密钥发送部分应用层数据.针对 3 种 0-RTT 模式,使用 Scyther 工具对其进行了形式化分析,得出了在 CK 安全模型下,0-RTT 数据的两种攻击,并基于其中的 1-RTT semi-static 模式提出了一种优化协议.通过安全性证明和形式化分析,证明了该优化协议在 CK 安全模型下能够抵抗针对 0-RTT 数据的 KCI 攻击和重放攻击.

关键词: TLS1.3;形式化分析;Syther;CK 安全模型;KCI 攻击

中图法分类号: TP311

中文引用格式: 陆思奇,周思渊,毛颖.强安全模型下 TLS1.3 协议的形式化分析与优化.软件学报,2021,32(9):2849−2866. http://www.jos.org.cn/1000-9825/5973.htm

英文引用格式: Lu SQ, Zhou SY, Mao Y. Formal analysis and optimization of TLS1.3 protocol in strong security model. Ruan Jian Xue Bao/Journal of Software, 2021,32(9):2849−2866 (in Chinese). http://www.jos.org.cn/1000-9825/5973.htm

Formal Analysis and Optimization of TLS1.3 Protocol in Strong Security Model

LU Si-Qi[1,2], ZHOU Si-Yuan[1], MAO Ying[2,3]

[1](PLA Information Engineering University, Zhengzhou 450001, China)

[2](State Key Laboratory of Information Security (Institute of Information Engineering, Chinese Academy of Sciences), Beijing 100093, China)

[3](School of Cyber Security, University of Chinese Academy of Sciences, Beijing 100093, China)

Abstract: TLS protocol works between the transport layer and application layer in TCP/IP system. The safety of transport layer is effectively protected by providing a series of security services such as confidentiality, integrity, authentication server required, as well as optional client authentication. In order to reduce network latency, TLS1.3 protocol adds the support for 0-RTT data, through caching long-term public key of server by client, and the long-term public key is directly used to generate a session key to send part of application layer data in the first message. For three kinds of 0-RTT mode, this study uses Scyther tools for formal analysis to obtain two attack paths of the 0-RTT data in CK security model, and an optimized protocol is proposed based on the 1-RTT semi-static mode. Through security proof and formal analysis, it is proved that the protocol is resistant to KCI attacks and replay attack against 0-RTT data in CK security model.

Key words: TLS1.3; formal analysis; Scyther; CK security model; KCI attacks

网络安全协议能够解决信息的传输安全问题,以互联网工程任务组(Internet engineering task force,简称

* 基金项目: 国家自然科学基金(61472414, 61772514, 61602061)
Foundation item: National Natural Science Foundation of China (61472414, 61772514, 61602061)
收稿时间: 2018-03-15; 修改时间: 2018-08-30; 采用时间: 2019-03-18

IETF)为代表的有关组织,为了弥补基本的 TCP/IP 协议簇在安全方面的设计缺陷,对现有的 TCP/IP 协议簇进行设计和改进,从而构成了包含多层网络安全协议的 TCP/IP 协议簇安全架构,传输层安全协议 TLS(transport layer security)就是这个安全架构中重要的组成部分.

TLS 协议是在安全套阶层协议 SSL(secure socket layer)获得了推广使用之后,由 IETF 基于 SSL3.0 协议[1]制定的正式行业标准[2],TLS1.2[3]为当前大部分浏览器和使用 HTTPS 协议的 Web 服务主要使用的版本.然而,随着网络环境日趋复杂,TLS1.2 也开始暴露出越来越多的漏洞,既有加密算法的漏洞,如 RC4 的漏洞[4],也有协议设计中的漏洞,如 Lucky 13[5]、POODLE[6]、Logjam[7]、三次握手攻击[8],以及实现过程中的漏洞,如 Heartbleed[9]、状态机攻击[10]等.虽然很多安全漏洞已经通过不定期更新和补丁解决,但由于 TLS 庞大的规模以及许多不同的设置,这些反复的修改增加了 TLS 的复杂性和部署难度.因此,IETF 的 TLS 工作组开始着手设计“下一代 TLS”——TLS1.3.目前,TLS1.3 正处于草案阶段[11],吸引了协议研究者们的广泛关注,既有针对 TLS1.3 攻击方法的研究,如证书转发攻击[12]、中间人攻击[13]、DROWN[14]、Client AuthenticationAttack [15,16],也有对 TLS1.3 的安全性分析和建议,如 OPTLS[17],TLS-Attacker 模型[18]等.

为了满足更高的安全需求,设计开发出更加安全稳定的网络安全协议,对网络安全协议进行更加严格、可信的分析,一直以来就是信息安全和密码学领域的一大热点.形式化方法使用数学的方法描述和推理计算机系统,具有准确的语法以及语义,并通过精确的数学手段和强大的分析工具实现技术支持.简而言之,就是规范语言加形式推理[19].形式化方法遍历搜索协议在运行过程中所有可能的状态,往往能够挖掘出安全协议中难以发现的细微漏洞,是目前非常流行的安全协议分析方法.

本文参考了 Krawczyk 等人[17]关于 OPTLS 的工作,使用形式化分析工具 Scyther 分析了 CK 模型下对 Early Data 的具体攻击.一方面,Early Data 不具有 PFS 安全,数据可能被攻击者窃取;另一方面, Early Data 无法抵抗 KCI 攻击.根据分析结果,本文对 TLS1.3 协议进行了优化,有效提高了协议的安全性.

本文第 1 节介绍 TLS1.3 和 Syther 等预备知识.第 2 节为使用 Scyther 工具对 TLS1.3 协议进行分析的过程和结论.第 3 节基于 TLS1.3 中的 1-RTT semi-static 模式设计一种优化的协议,在 CK 安全模型下从安全性证明和形式化分析的角度证明优化协议能够抵抗 KCI 攻击,并进行效率分析.第 4 节对本文的工作进行小结,进一步明确后续研究方向.

1 预备知识

1.1 Scyther简介

Scyther 是一款形式化分析安全协议的工具,该工具由牛津大学的 Cremers 教授及其团队[20]所开发.这款工具综合运用了定理证明以及模型检测等分析技术,具有如下特点.

(1) 能够多协议并行分析;
(2) 能够通过无限会话来验证协议;
(3) 能够通过描述协议,产生所有协议可能行为的有限表示;
(4) 能够通过组合敌手能力构造各种安全模型.

Scyther 对协议分析者最友好的一面是其基于按钮的使用界面以及可视化的分析结果,这使得协议的研究过程能够更加方便和直观.除此之外,Scyther 通过控制主体执行轮数和约束状态空间对搜索状态进行双重限制,与之前仅通过约束状态空间的方法相比,减少了攻击漏报的可能性,而且不会因为状态爆炸而出现无分析结果的情况.Scyther 的协议形式化语言 SPDL 简单易学,尽可能分割我们在安全协议形式化分析中的要点,即明确区分了协议的静态描述、动态行为以及敌手模型.更进一步讲,该语言模型实现了多方协议并行运作的直观掌控、安全声明的局部化、局部常量与角色实例的绑定以及初始敌手知识集合的简明定义.

Scyther 曾经被用来分析过许多著名协议,除了 Cremers[21,22]曾经用其分析过 IKEv1 和 IKEv2 协议套件和 ISO/IEC 9798 系列认证协议,Yang 和 Oleshchuk 等人[23,24]还使用 Scyther 分析了群认证协议,Ray 和 Chowdhury 等人[25]则使用 Scyther 分析了物联网协议.该工具最新的版本为 Scyther v1.1.3(标准版本,如果需要支持更多敌

手模型,可以选择 Scyther Compromise,其最新的版本为 Scyther Compromise-0.9.2).除此之外,该团队还开发了两款相关的工具,分别是 Scyther-proof[26]和 Tamarin[27].

本文使用的 Scyther 工具版本为 Scyther Compromise-0.9.1,该工具可在 Linux,Windows 和 Mac OS X 系统下运行.文中实验所选择的系统环境为英特尔 i5 的 2.5GHz 处理器、Ubuntu-16.04 操作系统、4G 内存的笔记本电脑.对 Ubuntu 原有源进行添加和更新后,安装 python,graphviz,python-wxgdk3.0 插件,解压后即可使用.

1.2 CK安全模型

CK(canetti-krawczyk)模型[28]对敌手的能力进行了形式化的描述,反映了敌手在开放网络环境中拥有的真实攻击能力:除了能够调度协议事件以及控制通信链路以外,攻击者还能够访问协议中使用或生成的一些秘密信息.具体来说,CK 模型定义了一个“中间人”攻击者 A,A 全面控制通信链路,能够拦截和修改消息、延迟或阻止消息传递、插入自己的消息、在不同的会话中交互消息以及安排所有会话的激活和会话消息的传递.此外,A 还被允许通过以下查询获取信息.

(1) 会话状态暴露(session-state reveal):针对还未完成的会话(如输出会话密钥之前),A 能够通过此查询得到特定会话的状态(如临时 DH 值的秘密指数,即临时私钥);

(2) 会话密钥查询(session-key query):在单个会话完成后,A 能够通过此查询得到会话密钥(在现实中,该查询可通过密码学分析、已知密钥攻击等方法实现);

(3) 腐化参与者(party corruption):A 能够攻陷实体,得到长期私钥以及和会话具体相关的信息.此时,A 可控制通信方所有行为.

这种模型允许攻击者显示一方测试会话时的长期私钥或显示一方的临时私钥,能帮助敌手获得更强的攻击能力.

1.3 TLS1.3的安全性质

TLS1.3 握手协议旨在通过认证密钥交换机制(AKE)来协商密钥,这些密钥可以被记录层用来提供关键的安全保证,包括消息的保密性和完整性.在 0-RTT 握手的情况下,应用程序数据由 PSK 保护,作为客户端第 1 条消息的一部分.

本文分析 TLS1.3 协议以下安全性质.

1. 会话密钥安全:一个密钥协商协议被称为会话密钥安全的,如果它对于任何攻击者都满足如下性质:
   (1) 未被攻陷的协议参与者在完成匹配的会话后生成的会话密钥相同;
   (2) 攻击者从协议输出的会话密钥和一个随机值中区分出会话密钥的概率不超过 $0.5+\varepsilon$(ε为一个可以忽略不计的概率);
2. 完美前向安全 PFS(perfect forward secrecy)[29]:即使已经泄露了用来产生会话密钥的长期密钥,也不会影响到之前使用的会话密钥的安全性,从而确保了之前通信内容也是安全的.这意味着假设存在一个敌手能够腐化服务器获得长期私钥,在这之前的会话密钥仍然能够保证安全性,或者导出密钥时不使用长期密钥,而是使用临时参数完成密钥生成;
3. 抗密钥泄露伪装攻击 KCI(key compromise impersonation):如果参与协议的用户 A 的长期私钥泄露,则攻击者能够利用 A 的私钥伪装成 A 的身份与其他用户执行协议,而不能伪装成其他用户与 A 成功完成协议.

1.4 GDH问题

设 Z_p^* 为一乘法群,g 为其生成元,则 Z_p^* 上的 CDH,DDH 问题描述如下.

(1) CDH 问题:输入 g,g^x,g^y,输出 g^{xy};

(2) DDH 问题:输入 g^x,g^y,g^z,判定 $g^{xy}=g^z$ 是否成立.

在此基础上,定义 GDH(gap diffie-hellman)问题[30]:在假定拥有解决 DDH 问题的有效方法的前提下,求解

CDH 问题.即:输入 g,g^x,g^y 和一个解决了 DDH 问题的 PPT 算法,输出 g^{xy}.

2 TLS1.3 协议的形式化分析

2.1 TLS握手协议

绝大多数的安全协议都采用了“认证密钥协商+对称加密传输”的结构,TLS 也是如此.因此,TLS 的主体就是使用对称加密传输信息的记录协议,以及为记录协议生成双方共享密钥的握手协议.除此之外,TLS 还有两个比较简单的辅助协议:用来报错和安全断连的警告协议和用来通知对方从握手协议切换到记录协议的更改密码规范协议.其中,后者因为作用不大,已经在 TLS1.3 草案中被删除.以上的 4 个协议构成了图 1 的 TLS 协议分层框架.以 TCP 为例,记录协议在 TCP 流上提供分包,其他协议都封装在记录协议的包里,然后在 TCP 层传输.

Fig.1 TLS protocol stack

图 1 TLS 协议栈

在 TLS 的协议栈中,握手协议是最关键的部分,也是 TLS1.3 主要修改的部分.因此,本文针对 TLS1.3 的分析主要是对 TLS1.3 握手协议的分析.对于 TLS1.3 握手协议的细节,后面将在第 2.3 节中进行详细的介绍,这里只对基本的 TLS 握手协议流程进行描述.

建立 TCP 连接之后,开始进入 TLS 的握手阶段.在这个阶段,客户端和服务器交换参数并进行密钥协商,传统的握手协议流程如图 2 所示(图中白底框表示可选的消息,或者在某些情况可能会发送的消息).

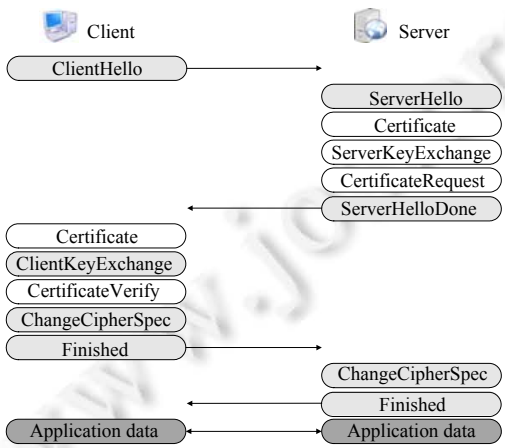


Fig.2 Basic handshake protocol flow

图 2 基本握手协议流程

2.2 1-RTT握手和0-RTT数据

TLS1.3 在 TLS1.2 的基础上做了许多重大改进,比如在记录协议层禁用 AEAD 之外的其他算法、在握手协议层去除静态 RSA 和 DH 密钥协商以及去除更改密码规范协议等等.上述这些改进的目的同之前 TLS 的升级一样,都是为了增强 TLS 的安全性.除此之外,IETF 还考虑了一个更现实的问题,那就是提升协议的运行速度.在互联网普及的今天,Web 服务的成功与否与网页加载速度的快慢有着密切联系,Amazon 公司曾在实验中发现:

如果网页加载延迟 100ms,销售量就会随之减少 1%.而对于服务器在地理位置上相距很远的用户来说,网络的延迟则更加不能忽略.因此,人们希望既不牺牲安全性又能够尽量地降低协议的运行时间.针对这一需求,TLS1.3 做出了如下改进.

(1) RTT 握手支持;

(2) RTT 数据支持.

RTT 是 Round Trip Time 的缩写,即消息的往返时间.对于 TLS1.2 来说,完成一次握手需要两个 RTT:第 1 个是 ClientHello 和 ServerHello,用于协商密钥交换算法以及各种不同的密码参数;第 2 个是 ClientKeyExchange 和 ServerKeyExchange,也就是使用前一个 RTT 协商的算法和参数进行密钥交换.TLS1.3 为了减少 RTT,决定取消第 1 个 RTT,让客户端缓存最近服务器使用的算法和参数,直接使用该缓存的算法和参数,与服务器进行密钥交换.如果客户端使用的算法和参数错误,则服务器返回 HelloRetryRewuest 消息,告知正确的算法和参数,让客户端重启握手,这就是 1-RTT 握手.该握手方式的可行性来源于 TLS1.3 对算法的削减,许多不安全的算法被抛弃,剩下的密钥协商算法其实只有屈指可数的几种.而可用的密钥协商算法越少,那么 1-RTT 握手成功的可能性就越高;并且即使客户端使用了错误的算法,也只需要增加一个 RTT 的代价,和 TLS1.2 相比没有增加额外的 RTT.这样,TLS1.3 基本没有副作用,便大大降低了 TLS 握手协议所需要的平均时长.

除了上述的 1-RTT 握手,TLS1.3 还借鉴了 Google 的 QUIC 协议[31],设计了一个 0-RTT 数据的解决方案.所谓 0-RTT,是指在握手协议的第 1 条消息中就允许客户端发送一些数据而不必等到握手协议结束.具体的解决方法是:允许客户端缓存服务器的长期公钥,在第 1 条消息中直接利用该长期公钥生成一个会话密钥发送部分应用层数据,另一部分应用层数据则等到完整的握手协议之后发送.在现实应用中,对于近期曾经访问过的网站,用户可以在第 1 次向服务器发送消息时就带上一部分应用数据,从而达到提高网页加载速度的目的.

值得注意的是:0-RTT 数据只是在 1-RTT 握手的基础上添加的数据,这个数据也被称为 Early Data,握手的其他部分依然保持 1-RTT 原有的消息.另外,这些数据不能抵抗重放攻击,该漏洞可通过一些独立于密钥交换的其他机制解决,如果想要利用 0-RTT 机制,则应该在应用层提供重放保护.

2.3 支持0-RTT数据的模式

TLS1.3 有 3 种模式支持 0-RTT 数据:1-RTT semi-static,PSK,PSK-DHE.图 3 为包含 0-RTT 数据的消息交互过程,表 1 表示各部分交互消息的意义,表 2 为 3 种模式所使用的密钥导出表.

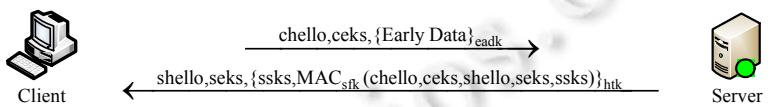


Fig.3 Handshake process that contains 0-RTT data flow

图 3 包含 0-RTT 数据的握手流程

Table 1 Protocol notation

表 1 符号定义

chello,shello	密钥协商参数
ceks,seks	临时公钥 g^x,g^y
ssks	服务器长期公钥 g^s
$ssks^+$	签名服务器密钥

Table 2 Key derivation table

表 2 密钥导出表

模式	ceks	seks	ssks	sfk	eadk	atk	htk	cost
1-RTT semi-static	g^x	g^y	g^s	g^{xs}	g^{xs}	g^{xs},g^{xy}	g^{xy}	2DH
PSK	–	–	–	psk	psk	psk	psk	–
PSK-DHE	g^x	g^y	–	psk	psk	psk,g^{xy}	g^{xy}	1DH

由图可知,客户端 Client 通过 chello 消息传递密钥协商的参数,通过 ceks 传递密钥协商过程中的临时公钥 g^x,并利用服务器的长期公钥加密(或客户端临时私钥和服务器长期公钥计算得出的 eadk=g^{xs})保护 Early Data 发送给服务器 Server,服务器端接收到消息后,向客户端 Client 回复密钥协商的参数 shello、临时公钥 g^y 以及用共享密钥 g^{xy} 保护的消息,该消息包括服务器的长期公钥和对协议传递消息的确认.

2.3.1 1-RTT semi-static 模式

首先分析 1-RTT semi-static 模式.在使用 Scyther 引擎提供的 SPDL 语言进行描述时,为了区别临时私钥和长期私钥,本文采用在临时密钥生成时加入随机数 *tempx* 和 *tempy* 的方法,即用(*sk*(*c*),*tempx*),(*sk*(*s*),*tempy*)表示客户端、服务器的临时私钥,用 *g*1(*sk*(*c*),*tempx*),*g*1(*sk*(*s*),*tempy*)表示客户端、服务器的临时公钥.因为 *tempx* 和 *tempy* 是保密的,这样在长期私钥被泄露时,能够保证临时私钥仍然是安全的.此外,本文使用 CK 模型作为安全模型,这种模型允许攻击者显示一方测试会话时的静态私钥(长期私钥)或显示一方的临时私钥,能帮助敌手获得更强的攻击条件和能力.Scyther 工具在 Settings 标签里面提供了强安全模型的设置面板,根据 Cas Cremers[32,33]对 CK 等强安全模型的研究,CK 模型允许敌手得知通信方长期私钥和协议执行过程的中间状态,用以解释 CK 模型下的前向保密性 Long-term Key Reveal after claim(PFS)、密钥泄露伪装攻击 Actor(KCI[34])和状态泄露攻击 State Reveal.此外,双方协商的共享密钥还要求抵抗已知会话密钥泄露攻击 Session-Key Reveal.因此,敌手模型的设置如图 4 所示.最后点击“Verify automatic claims”进行自动验证,结果如图 5 所示.

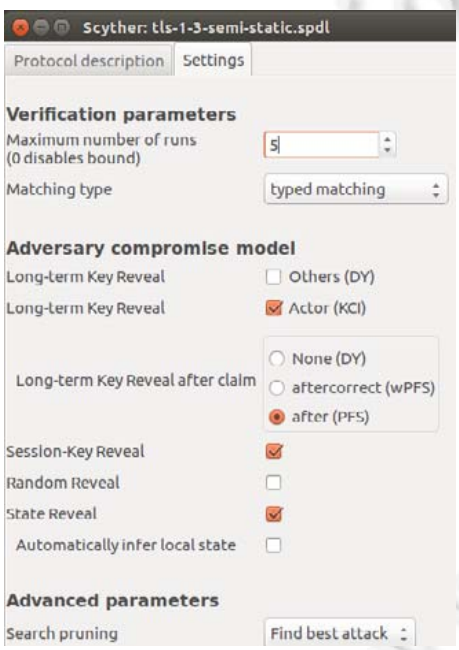


Fig.4 Setting attacker model

图 4 设置敌手模型

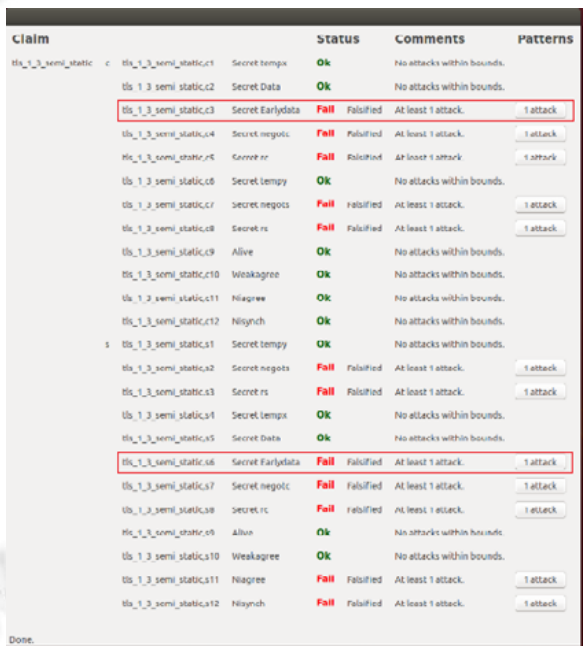


Fig.5 Attack paths of 1-RTT semi-static mode

图 5 1-RTT semi-stati 模式的攻击路径

从运行结果可以看出:Scyther 工具分别在客户端和服务器的角度对协议执行过程中所有参数的机密性进行了验证,并对双方的双向认证过程中的一致性 Niagree 和同步性 Nisynch 进行验证.关于 Early Data 参数的机密性验证,结果是不安全的(fail),分别在客户端和服务器端存在两种攻击,点击“1 attack”按钮可查看详细的攻击路径,分别如图 6 和图 7 所示.此外,Scyther 还分析出了一些其他的攻击,如随机数 rc/rs 和协议参数 negotc/ negots 的机密性,这些参数在协议交互过程中本就是明文通信的,它们的泄露并不影响认证密钥协商过程的安全性,所以它们的安全性不在考虑范围之内.而关于认证方面,Niagree 和 Nisynch 所展示的攻击则与图 7 中的攻击路径完全相同,这也说明密钥泄露伪装攻击 KCI 破坏的是双方的认证安全.

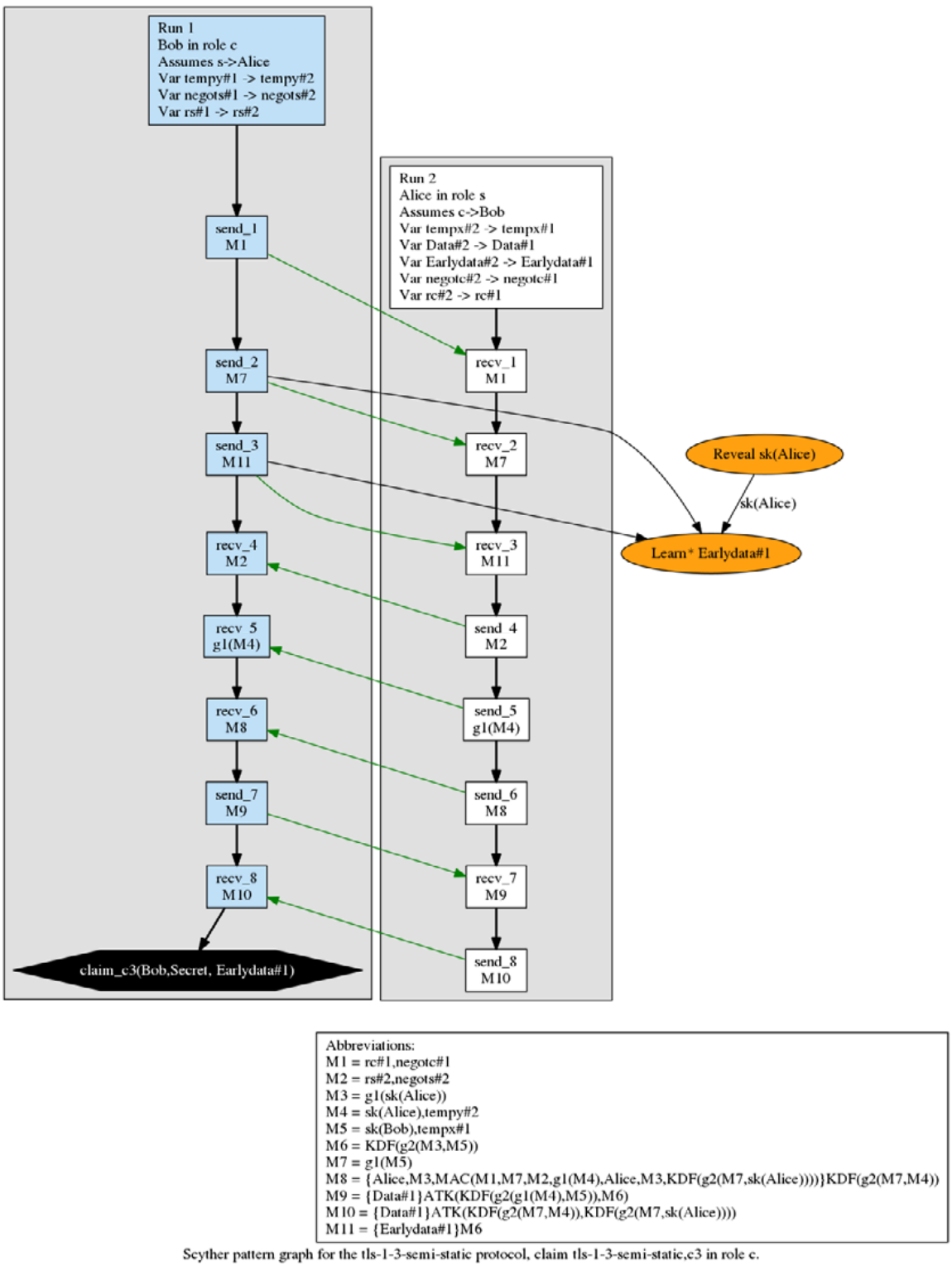
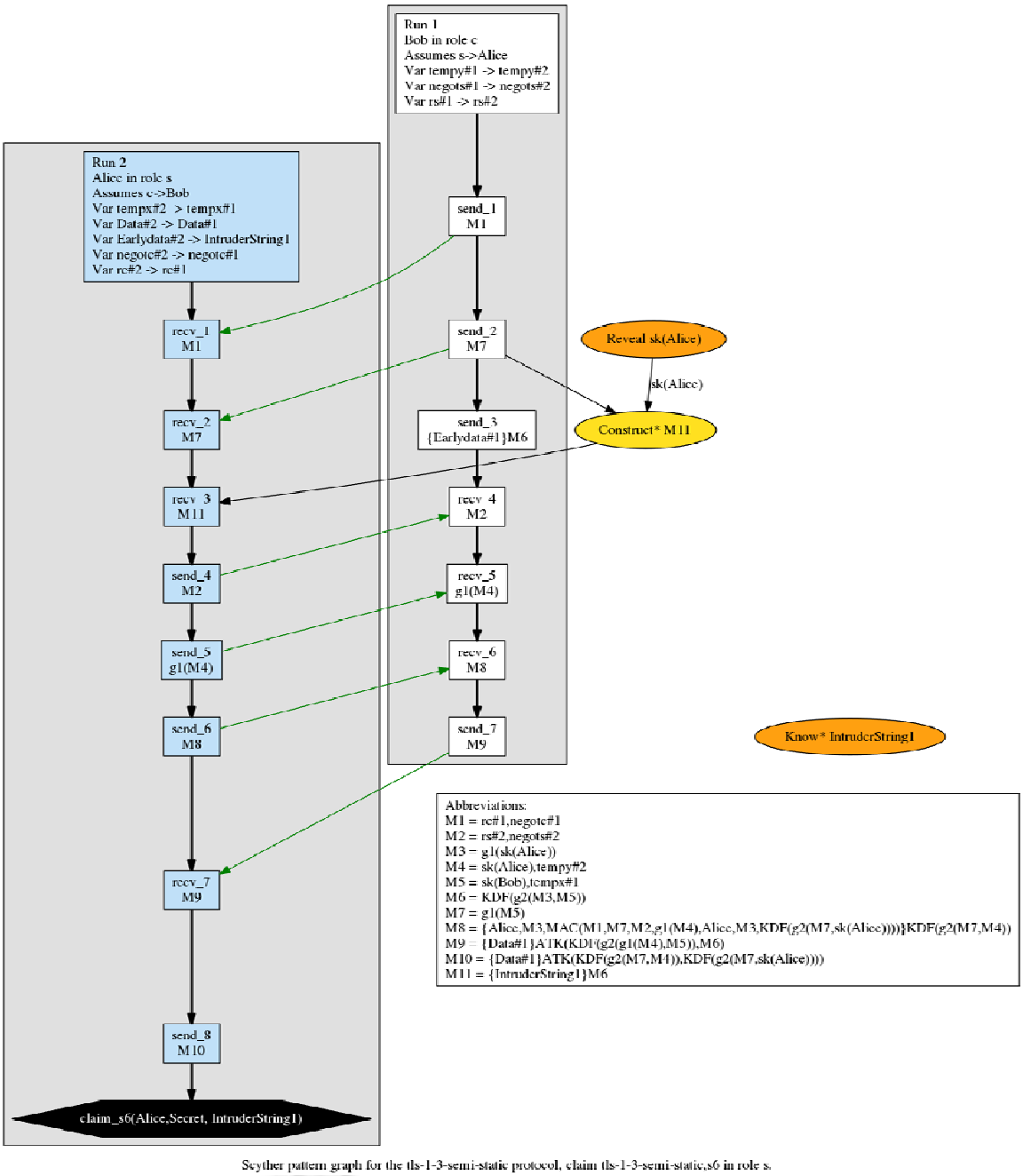


Fig.6 Early Data leaked in 1-RTT semi-static mode

图 6 1-RTT semi-static 模式下的 Early Data 泄露



Scyther pattern graph for the tls-1-3-semi-static protocol, claim tls-1-3-semi-static,s6 in role s.

Fig.7 KCI attack in 1-RTT semi-static mode

图 7 1-RTT semi-static 模式下的 KCI 攻击

图 6 中的攻击描述的是 1-RTT semi-static 模式下 Early Data 的泄露过程,两个 run 之间交互的路径表示正常的协议交互流程.可以观察到:在扮演客户端角色的 Bob(*C*)向扮演服务器角色的 Alice(*S*)发送消息 M7(即 Bob 的公钥)时,敌手可以截获这条消息,再通过 CK 模型腐化 Alice 获得 Alice 的长期私钥,从而计算出用于加密 Early Data 的密钥 eadk.这时再截获 M11 消息,即可对其中加密的 Early Data 进行解密.

具体攻击描述如下.

(1) 客户端 *C* 与服务器 *S* 正常通信,*C* 使用自己的私钥和 *S* 中的长期公钥计算出密钥 eadk,用 eadk 加密在第 1 条消息中发送的 Early Data.此外,*C* 还需要向 *S* 发送自己的公钥;

(2) 攻击者 *A* 截获 *C* 向 *S* 发送的 *C* 的公钥以及加密后的 Early Data 密文;

(3) *A* 使用 *C* 的公钥和通过腐化参与者获取的 *S* 的长期私钥计算出密钥 eadk;

(4) *A* 使用 eadk 解密 Early Data,获得明文.

图 7 中的攻击路径描述的是 1-RTT semi-static 模式下 KCI 攻击的流程.攻击者在扮演客户端的角色 Bob(*C*)向扮演服务器的角色 Alice(*S*)发送消息 M7(即 Bob 的公钥)时将其截获,再通过 CK 模型腐化 Alice 获得 Alice 的长期私钥,从而计算出密钥 eadk.此时,攻击者可以自己伪造一个新的 Early Data,用 eadk 加密发送给 Alice,则 Alice 会将这条伪造的消息当作是 Bob 发送的 Early Data,并继续进行之后正常的协议流程.这样,攻击者就能够成功冒充 Bob 向 Alice 发送消息.

具体攻击描述如下.

(1) 客户端 *C* 与服务器 *S* 正常通信,*C* 使用自己的私钥和 S 中的长期公钥计算出密钥 eadk,用 eadk 加密在第 1 条消息中发送的 Early Data.此外,*C* 还需要向 *S* 发送自己的公钥;

(2) 攻击者 *A* 截获 *C* 向 *S* 发送的 *C* 的公钥;

(3) *A* 使用 *C* 的公钥和通过通过腐化参与者获取的 *S* 的长期私钥计算出密钥 eadk;

(4) *A* 可以任意伪造 Early Data,使用 eadk 加密发送给 *S*,*S* 则会认为这条消息来自于 *C*.

综上所述,根据 Scyther 分析的结果,1-RTT semi-static 模式下的 0-RTT 数据存在如下漏洞.

(1) 0-RTT 数据没有 PFS 安全;

(2) 无法抵抗 KCI 攻击,即:如果服务器的长期私钥被攻击者窃取了,则攻击者可以伪装成任意客户端向服务器发送任意伪造的 Early Data.当然,攻击者拥有服务器的长期私钥,模拟服务器与客户端通信也是平凡的;

(3) 0-RTT 中的数据能够被跨连接重放.

2.3.2 PSK 模式

如表 2 所示,在 PSK 模式中,所有的密钥都由一个预共享密钥 psk 导出.因此,在用 SPDL 语言描述时,将 psk 定义为 const 常量,表示敌手有获取它的能力.点击“Verify automatic claims”进行自动验证,得到如图 8 所示结果.

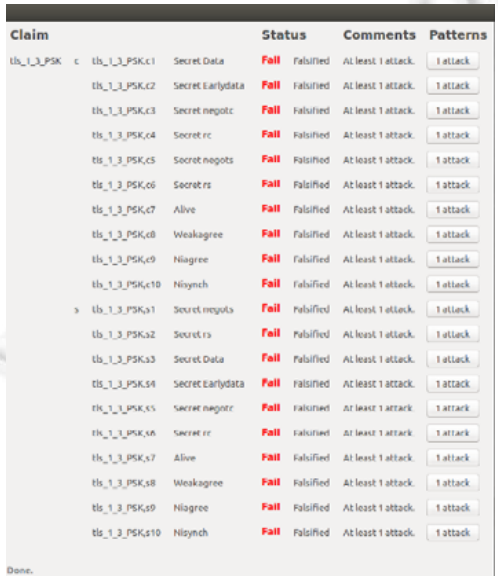


Fig.8 Attack paths of PSK mode

图 8 PSK 模式的攻击路径

从运行结果中可以看出:如果 psk 被泄露,则 Early Data 和 Application Data 都将失去安全性,即二者均没有达到 PFS 安全的要求.这是由于 PSK 模式下,双方没有临时私钥和临时公钥参与密钥协商过程,在强安全模型下,整个协议仅使用一个 psk 导出密钥会导致攻击者能够伪造协议中的任何消息,包括 Finished 消息中的 MAC,攻击路径如图 9 所示.

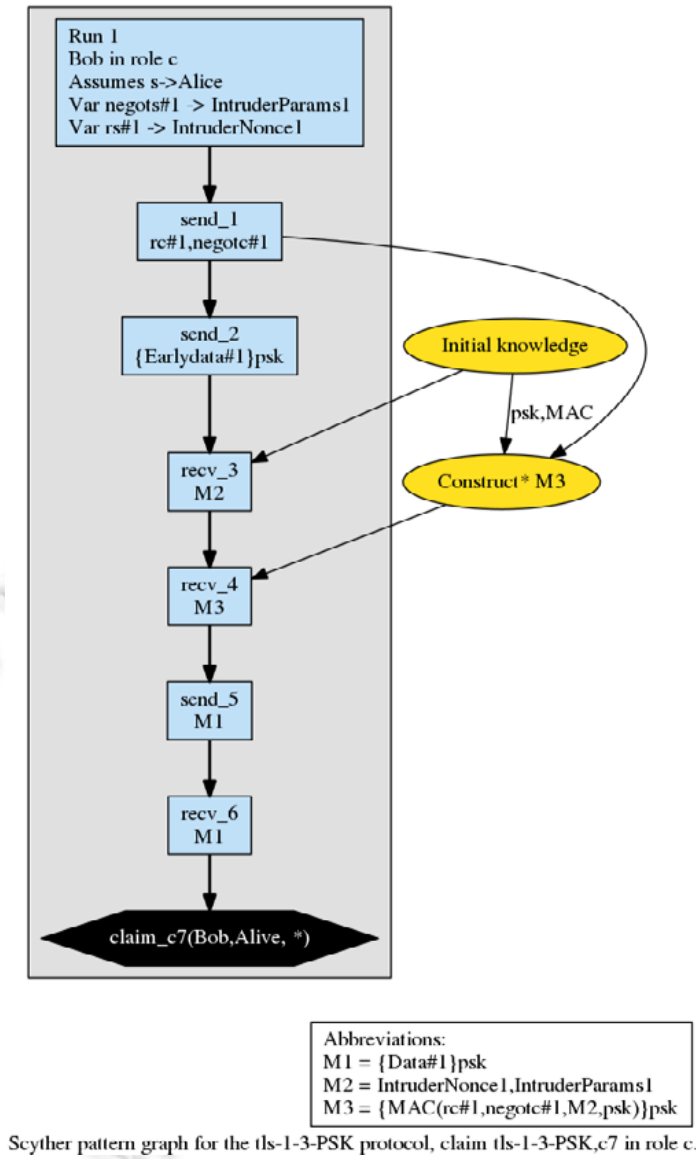


Fig.9 MAC forged in PSK mode

图 9 PSK 模式下的 MAC 伪造

在该攻击中,一个已知 psk 的攻击者可以截获扮演客户端的角色 Bob(*C*)的 ClientHello(图中表示为 chello)消息,并伪造一个 ServerHello(图中表示为 shello)消息返回给 Bob;最后计算出对应的 MAC,加密发送给 Bob,其中,计算 MAC 的密钥 sfk 和握手密钥 htk 都由 psk 导出.

具体攻击描述如下.

(1) 攻击者 *A* 通过一定方法获取客户端 *C* 与服务器 *S* 的预共享密钥 psk;

(2) *A* 劫持 *C* 与 *S* 的会话,拦截所有 *C* 向 *S* 发送的消息;

(3) *A* 伪造一个 ServerHello 消息返回给 *C*;

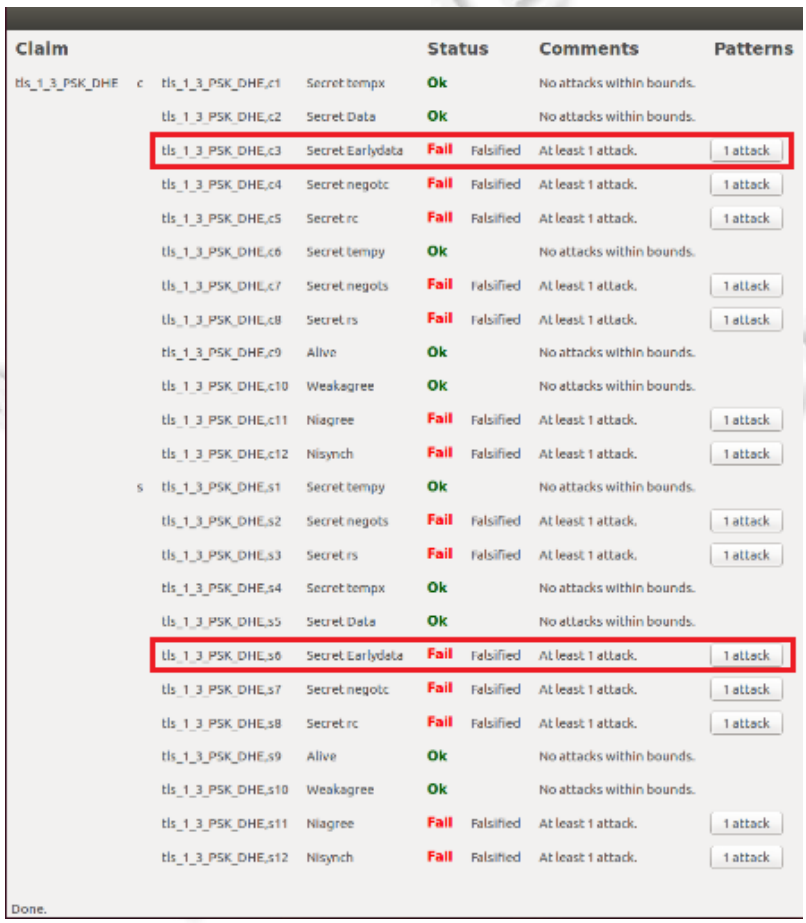
(4) *A* 使用 psk 计算一个 MAC,MAC 的内容包括 *C* 发送的 ClientHello 消息、*A* 伪造的 ServerHello 消息以及 psk 的值;

(5) *A* 将 MAC 用 psk 加密发送给 *C*,完成 1-RTT 握手,伪装成服务器与 *C* 建立会话.

可以说,这种模式的安全性完全取决于 psk 的保密性,一旦 psk 被攻击者获取,整个协议流程都将变得极其不安全.

2.3.3 PSK-DHE 模式

PSK-DHE 在 PSK 模式的基础上增加了一轮 DH 交换,使 Application Data 满足 PFS 安全的要求.为了区分临时私钥和长期私钥,采用与第 2.3.1 节相同的方式表示客户端与服务器的临时私钥.运行结果如图 10 所示.



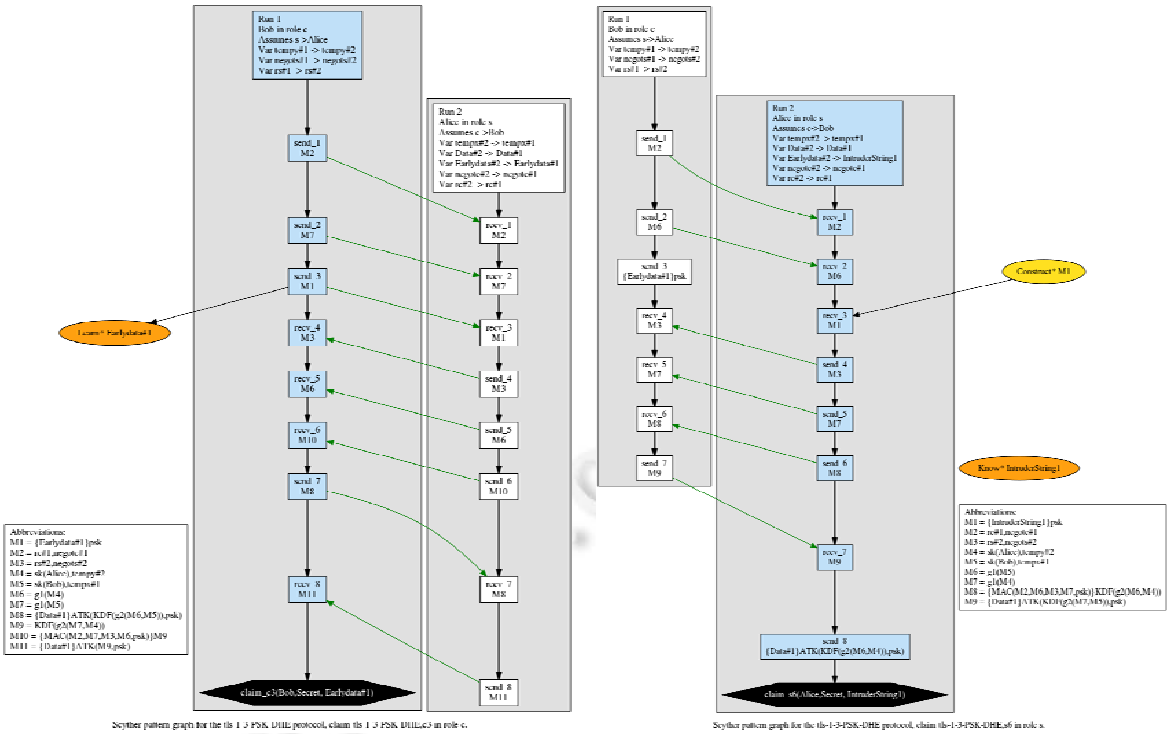
Claim				Status		Comments	Patterns
tls_1_3_PSK_DHE	c	tls_1_3_PSK_DHE,c1	Secret tempx	Ok		No attacks within bounds.	
		tls_1_3_PSK_DHE,c2	Secret Data	Ok		No attacks within bounds.	
		tls_1_3_PSK_DHE,c3	Secret Earlydata	Fail	Falsified	At least 1 attack.	1 attack
		tls_1_3_PSK_DHE,c4	Secret negotc	Fail	Falsified	At least 1 attack.	1 attack
		tls_1_3_PSK_DHE,c5	Secret rc	Fail	Falsified	At least 1 attack.	1 attack
		tls_1_3_PSK_DHE,c6	Secret tempy	Ok		No attacks within bounds.	
		tls_1_3_PSK_DHE,c7	Secret negots	Fail	Falsified	At least 1 attack.	1 attack
		tls_1_3_PSK_DHE,c8	Secret rs	Fail	Falsified	At least 1 attack.	1 attack
		tls_1_3_PSK_DHE,c9	Alive	Ok		No attacks within bounds.	
		tls_1_3_PSK_DHE,c10	Weakagree	Ok		No attacks within bounds.	
		tls_1_3_PSK_DHE,c11	Niagree	Fail	Falsified	At least 1 attack.	1 attack
		tls_1_3_PSK_DHE,c12	Nisynch	Fail	Falsified	At least 1 attack.	1 attack
	s	tls_1_3_PSK_DHE,s1	Secret tempy	Ok		No attacks within bounds.	
		tls_1_3_PSK_DHE,s2	Secret negots	Fail	Falsified	At least 1 attack.	1 attack
		tls_1_3_PSK_DHE,s3	Secret rs	Fail	Falsified	At least 1 attack.	1 attack
		tls_1_3_PSK_DHE,s4	Secret tempx	Ok		No attacks within bounds.	
		tls_1_3_PSK_DHE,s5	Secret Data	Ok		No attacks within bounds.	
		tls_1_3_PSK_DHE,s6	Secret Earlydata	Fail	Falsified	At least 1 attack.	1 attack
		tls_1_3_PSK_DHE,s7	Secret negotc	Fail	Falsified	At least 1 attack.	1 attack
		tls_1_3_PSK_DHE,s8	Secret rc	Fail	Falsified	At least 1 attack.	1 attack
		tls_1_3_PSK_DHE,s9	Alive	Ok		No attacks within bounds.	
		tls_1_3_PSK_DHE,s10	Weakagree	Ok		No attacks within bounds.	
		tls_1_3_PSK_DHE,s11	Niagree	Fail	Falsified	At least 1 attack.	1 attack
		tls_1_3_PSK_DHE,s12	Nisynch	Fail	Falsified	At least 1 attack.	1 attack

Done.

Fig.10 Attack paths of PSK-DHE mode

图 10 PSK-DHE 模式的攻击路径

从图 10 中可以看出:Data 的 Secret 属性为 OK,即 Application Data 满足 PFS 安全.由于 PSK-DHE 模式的架构与 1-RTT semi-static 模式类似,只是把使用服务器长期密钥进行的一轮 DH 交换改为使用预共享密钥机制,因此可以看到:PSK-DHE 模式中,Early Data 的安全性类似 1-RTT semi-static 模式,仍然没有达到 PFS 安全,也不能抵抗 KCI 攻击.攻击路径如图 11(a)和图 11(b)所示,攻击流程与 1-RTT semi-static 模式下攻击相同.



(a) PSK-DHE 模式下的 Early Data 泄露 (b) PSK-DHE 模式下的 KCI 攻击

Fig.11 Early Data leaked and KCI attack in PSK-DHE mode

图 11 PSK-DHE 模式下的 Early Data 泄露和 KCI 攻击

3 优化协议方案

从上述结果可以看出:对于一个能够获取服务器长期私钥的敌手来说,0-RTT 数据将变得不再安全.事实上,这一问题对于 Google 的 QUIC 协议同样存在,但是因为 QUIC 协议主要应用在谷歌浏览器上,在访问网站时,通常使用 GET 作为建立连接的第 1 个请求,所以 0-RTT 中的数据敏感性可能并不是很高,故其安全性可以选择忽视.然而对于使用 POST 作为第 1 个请求的连接,比如微信的短连接,发送给服务器端的都是上层安全性要求很高的业务数据,在这种情况下,0-RTT 数据的安全问题必须考虑.对此,服务器长期私钥对应的公钥其缓存时间必须严格限制.通常,服务器通过 ServerConfiguration 消息将长期公钥发送给客户端.ServerConfiguration 的数据结构如下.

```
struct {
        opaque configuration_id ⟨1,...,2^16−1⟩;
        uint32 expiration_date;
        NamedGroup group;
        opaque server_key ⟨1,...,2^16−1⟩;
        EarlyDataType early_data_type;
        ConfigurationExtension extensions ⟨1,...,2^16−1⟩;
} ServerConfiguration;
```

其中,expiration_date 就是该 ServerConfiguration 的有效使用期限,被设定为小于 7 天.也就是说,客户端缓存服务器半静态公钥的时间不得超过 7 天.

3.1 优化协议的结构

从 ServerConfiguration 的有效使用期限可以看出,客户端需要频繁地缓存半静态公钥.这对于握手协议的效率来说造成了一定的影响,并且也无法从根本上解决 0-RTT 数据的安全性问题.关于 0-RTT 数据的安全性问题,根据第 2 节中形式化分析的结果,主要包括没有 PFS 安全和无法抵抗 KCI 攻击.本文基于 1-RTT semi-static 模式设计了一种新的支持 0-RTT 数据的协议,该协议可以有效抵抗 KCI 攻击,提高了 0-RTT 数据的安全性.协议的消息交互过程如图 12 所示,由图可见:相比原协议传输,优化后的协议在原 Early Data 消息的基础上,添加了客户端对 Early Data 和时间戳的签名.

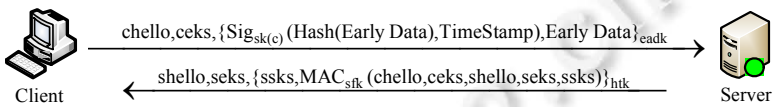


Fig.12 Handshake process of the optimized protocol

图 12 优化协议的握手流程

服务器在收到客户端的第 1 条消息后,首先计算出 eadk 对 0-RTT 数据进行解密,然后使用客户端的公钥对其中的签名进行解密,解密后对时间戳进行验证.此外,将时间戳 TimeStamp 放在签名中,同样防止了敌手对时间戳的篡改,从而排除重放攻击的可能,再与 Early Data 的 Hash 值进行对比:如果二者不匹配,则认为这不是一个合法的客户端发送的 0-RTT 数据,服务器应该选择立即丢弃这个数据.

3.2 优化协议的安全性证明

本文采用了 CK 模型对上述优化协议进行分析,该模型在可信的计算环境下能够简化协议的设计,使协议更加具有普遍性和实用性.

使用基于 game 的方法对优化协议的安全性进行分析,证明在签名算法的安全性和 Gap-DH 问题的复杂性以及密钥导出函数的安全性的前提下,该协议能够抵抗 KCI 攻击.证明过程注意以下两点.

(1) session-key query 查询的密钥为 Early Data 的密钥 eadk,而不是 Application Data 的密钥 atk;

(2) 定义匹配的会话为(chello,ssks)相等的会话(只针对 eadk,不针对 atk).

定理 1. 在假设服务器证书机制的安全性、求解 Gap-DH 的困难性、密钥导出函数的安全性、客户端签名机制的安全性的前提下,优化协议能够抵抗 KCI 攻击.

- 正确性

假设一个诚实的协议参与者拥有抗重放机制来产生 chello,那么 chello 唯一识别 ceks=g^s.也就是说,对于一对匹配的会话,协议双方的 chello,ceks,ssks 都相同,因此计算出相同的 eadk.

- 安全性

接下来定义一系列 Game.

(1) Game 0:优化协议的真实执行过程;

(2) Game 1:服务器证书不可伪造.游戏规则与 Game 0 相同,在该游戏中,攻击者使用随机值 ssks*代替 g^s,因为签名或者其他任何用于认证 ssks 的方案是安全的,因此攻击者无法赢得 Game 1.也就是说,攻击者不能随意伪造服务器的证书;

(3) Game 2:eadk 的安全性.该游戏发生在 s 被泄露之前,游戏规则与 Game 0 相同,但是攻击者不能使用 party corruption 查询.在该游戏中,攻击者查询 g^{xs},或者从随机值和 g^{xs} 中区分出 g^{xs}.根据 Gap-DH 困难性假设:在给出 g,g^x,g^s 和 DDH 查询的前提下,计算 g^{xs} 是困难的.即,攻击者赢得 Game 2 的概率等同于求解 Gap-DH 难题;

(4) Game 3:eadk 的安全性.该游戏发生在 s 被泄露之前,游戏规则与 Game 0 相同,但是攻击者不能使用 party corruption 查询.在该游戏中,攻击者在一个会话完成后,用随机值代替另一个会话的 eadk*.因为

密钥导出函数是安全的.

- 对 eadk 的 session-key query 查询不影响 $eadk^*$,因为 $chello^* \neq chello$;
- atk,sfk 甚至 atk^*的泄露,都不会影响 $eadk^*$.

因此,攻击者无法赢得 Game 3.

(5) Game 4:Earlydata 的真实性和可靠性.该游戏规则与 Game 0 相同,攻击者在该游戏中用随机值代替 $Sig_{sk(c)}(Hash(Earlydata))$.因为用于签名 Earlydata 的密钥 sk(c)和算法是安全的,所以攻击者无法赢得 Game 4.也就是说,攻击者可以通过 party corruption 使 *s* 泄露,并计算出 eadk,进一步得到 Earlydata 的内容,但是伪造 $Sig_{sk(c)}(Hash(Earlydata))$是困难的,因此保证了 Early Data 消息来源的真实性和可靠性.

综上所述,优化协议能够防止攻击者伪装成任意客户端伪造 Early Data 消息,从而有效抵抗 KCI 攻击.

3.3 优化协议的形式化分析

利用 Scyther 工具对优化后的协议进行分析,这里在编写形式化语言时需要特别注意对签名密钥的描述.签名的公私钥对必须具备两条基本属性.

(1) 在 CK 模型下用于签名的私钥仍然是保密的;

(2) 签名的公钥是公开的.

因为 Scyther 对于角色 *C* 来说只有一个私钥就是 sk(c),为了区分临时私钥和长期私钥,之前在第 2.3.1 节中提到了在 sk(c)后加一个随机数的解决方案,也就是用(sk(c),sigx)的形式来表示签名密钥(sigx 为随机数).但是这里因为公钥是公开的,那么也就是说敌手必须要知道(pk(c),sigx),而 pk(c)是已知的,所以这种表达方式会让敌手得到 sigx,再加上设置了 CK 模型的前提下,*C* 的私钥 sk(c)也是已知的,那么(sk(c),sigx)同样能够被敌手得知.因此,本文使用 sk(c,sigx)来表示签名私钥.经过实验证明:在敌手已知 pk(c,sigx)和 pk(c),并且设置 CK 模型使得 sk(c)也是不安全的前提下,sk(c,sigx)仍然是安全的.

根据之前的实验发现:使用“Verify automatic claims”会分析出很多不影响实验结果的攻击路径,不仅降低了工具的分析效率,而且很多攻击路径对实验也没有帮助,因此,针对性地在 SPDL 语言中加入 claim 语句.

- claim(c,Secret,Earlydata);
- claim(s,Secret,Earlydata);
- claim(s,Niagree);
- claim(s,Nisynch);

在搜索攻击的设置选项里,选择“Find all attacks”而不是“Find best attack”,以防漏报.观察运行结果,发现所有的攻击路径描述的都是同一种攻击,如图 13 所示.

在一次运行过程中,攻击者截获客户端 Bob(*C*)的公钥并腐化服务器 Alice(*S*)得到其长期私钥,计算出这一次通信的 eadk,从而得到 Early Data 和对应的签名;在另一次运行过程时,攻击者可以将这两条消息作为 0-RTT 里面的数据,用这一次通信的 eadk 加密发送给客户端 Bob,完成对 Early Data 的重放攻击.

具体攻击如下.

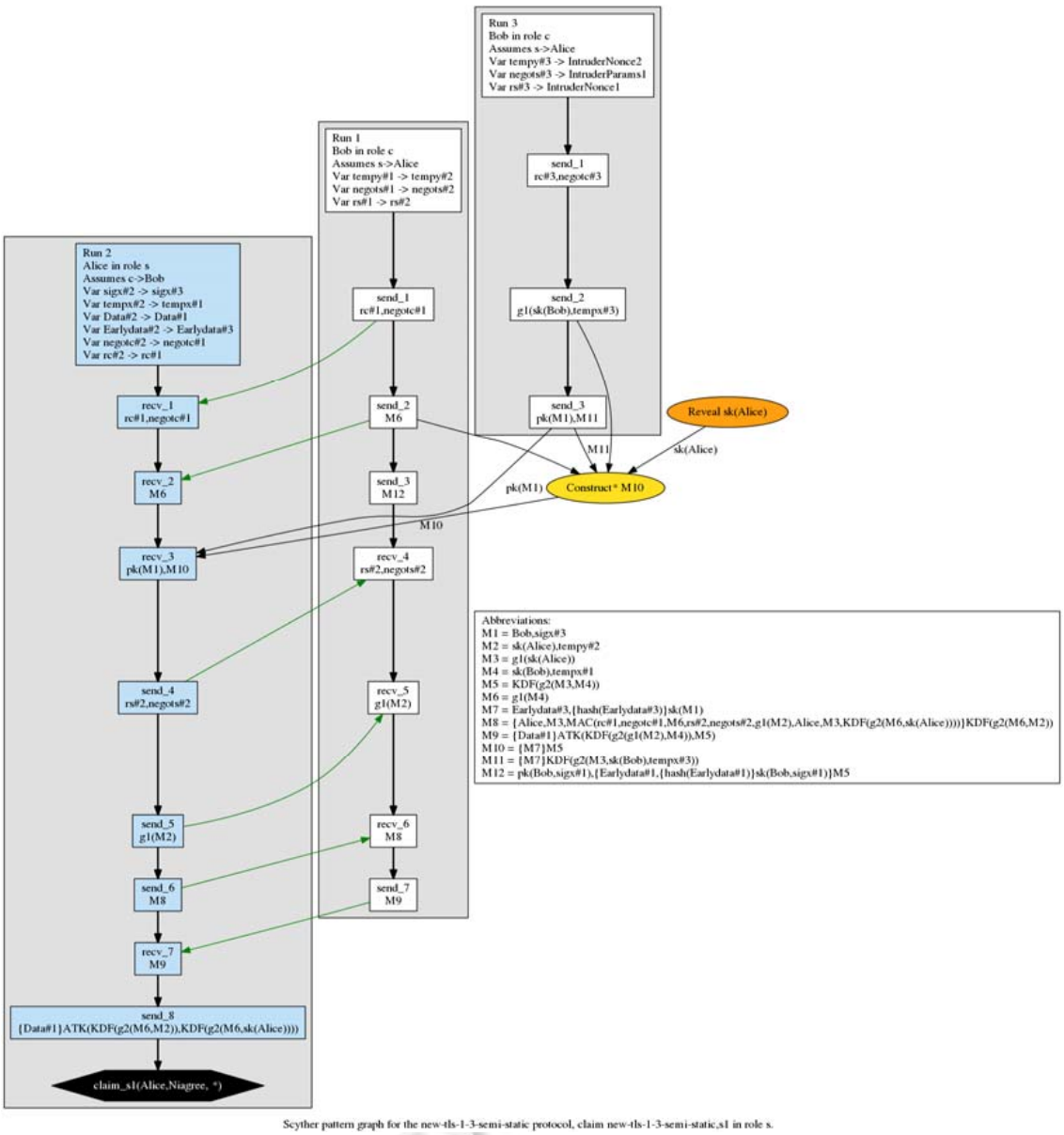
(1) 客户端 *C* 与服务器 *S* 正常通信,*C* 使用此次通信 *C* 的私钥和 *S* 的长期公钥计算出密钥 eadk,用 eadk 加密在第 1 条消息中发送的 Early Data 以及 Early Data 摘要的签名;此外,*C* 还需要向 *S* 发送此次通信 *C* 的公钥;

(2) 攻击者 *A* 截获 *C* 向 *S* 发送的此次通信 *C* 的公钥以及加密后的 Early Data 和 Early Data 摘要的签名;

(3) *A* 使用 *C* 此次通信的公钥和通过腐化参与者获取的 *S* 的长期私钥计算出加密此次通信的 eadk;

(4) *A* 使用 eadk 解密获得 Early Data 和 Early Data 摘要的签名;

(5) 在 *C* 和 *S* 的另一次通信中,*A* 可以用步骤(4)中获得的 Early Data 和 Early Data 摘要的签名替换本次通信中的 Early Data 和 Early Data 摘要的签名,完成消息的重放.



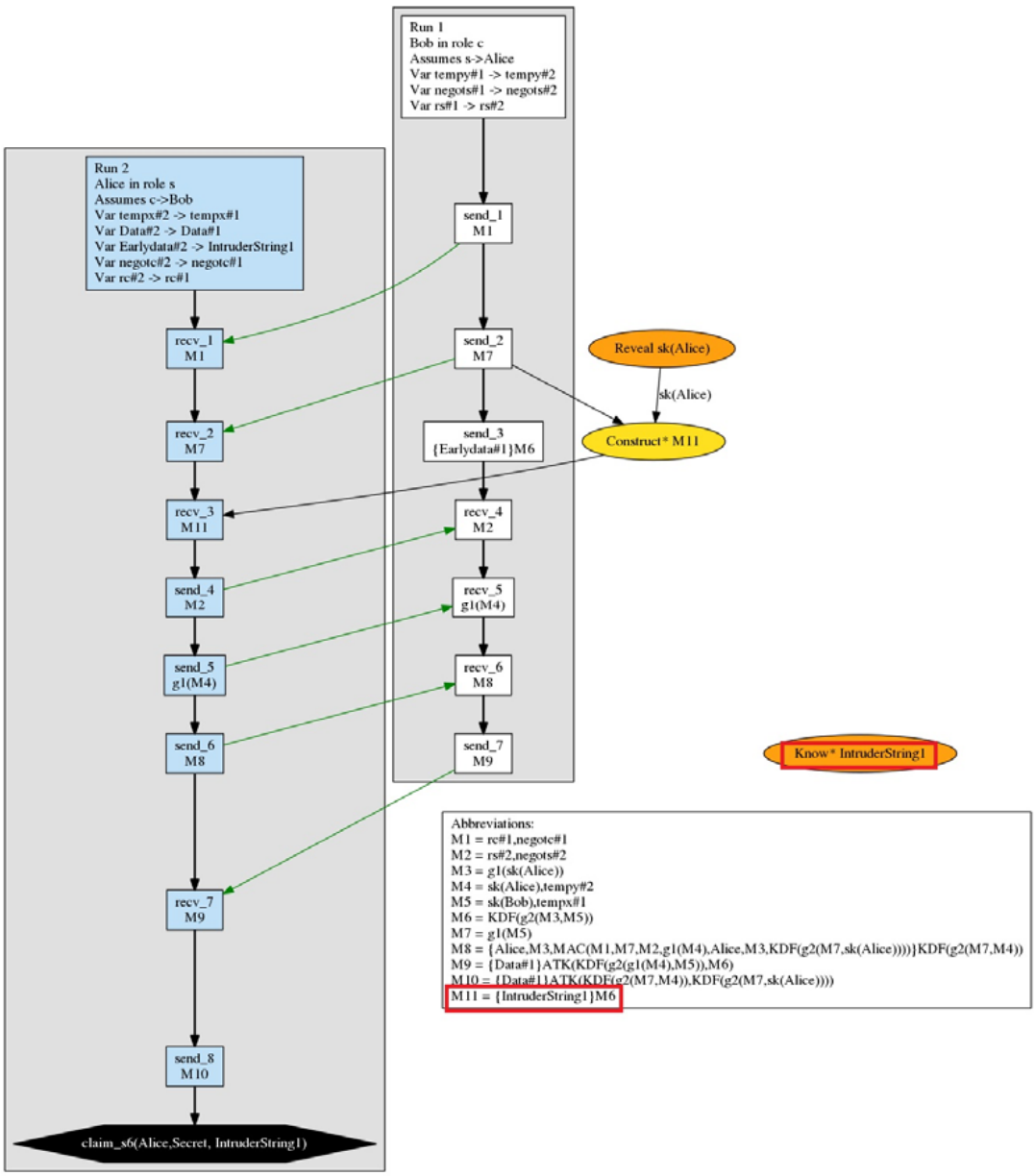
Scyther pattern graph for the new-tls-1-3-semi-static protocol, claim new-tls-1-3-semi-static,s1 in role s.

Fig.13 Raplay attack to Early Data in the optimized protocol

图 13 优化协议中 Early Data 的重放攻击

对比 1-RTT semi-static 模式中 claim(s,Niagree)所描述的 KCI 攻击(如图 14 所示)可以发现:优化协议中,攻击者发送的 Early Data 数据仍然为合法客户端所发送的数据,攻击者只能够对其简单重放;而 1-RTT semi-static 模式中的 Early Data 数据为 IntruderString,即攻击者自己定义的消息,在这种情况下,攻击者不仅可以重放消息,甚至可以对重放的消息进行任意篡改,其风险显然远大于优化协议中的重放攻击.

综上所述,相比之前的 1-RTT semi-static 模式,该优化协议能够有效抵抗 KCI 攻击,验证了之前的证明结果.PSK-DHE 模式与 1-RTT semi-static 模式类似,同样可以运用该优化协议的思路,在 Early Data 上加一层签名,即可保证 0-RTT 中的数据能够抵抗 KCI 攻击.表 3 总结了 1-RTT semi-static 模式和优化协议的安全性对比.



Scyther pattern graph for the tls-1-3-semi-static protocol, claim tls-1-3-semi-static,s6 in role s.

Fig.14 KCI attack in 1-RTT semi-static mode

图 14 1-RTT semi-static 模式下的 KCI 攻击

Table 3 Security comparison

表 3 安全性对比

模式	PFS	抗重放	防止伪造	防止冒充
1-RTT semi-static	×	×	×	×
优化模型	×	√	√	√

可以看出:优化协议有效提高了 0-RTT 数据的安全性,防止了攻击者冒充合法用户与服务器建立非法链接.对于重放攻击的问题,在签名中添加时间戳,有效防止了敌手对时间戳的篡改和伪造,服务器通过对时间戳的验证实现了抗重放机制.

3.4 优化协议的效率分析

为了分析优化协议的效率,表 4 对比了 1-RTT semi-static 模式和优化协议所需要的主要密码工具.

Table 4 Main password tools comparison

表 4 主要密码工具对比

模式	DH	MAC	SIG
1-RTT semi-static	1	1	1
优化模型	1	1	2

从表中可以看出:优化协议与 1-RTT semi-static 模式相比,多使用了一个签名(在传输服务器的长期公钥证书的时候还需要另一个签名,这个签名是二者都必需的),并且为 Early Data 增加一个签名没有增加发送的消息数,仍然在 0-RTT 发出,因此不会增加协议的运行时间.此外,增加的签名也不会影响到其他消息,特别是 Finished 消息,因为签名的数据不包括在 MAC 的内容中,因此不会影响到 MAC 的计算效率.需要注意的是:优化协议使用的签名算法建议采用固定的算法,这样就不需要再增加 chello 消息中协议参数的内容,从而防止增加 MAC 计算的内容.

综上所述,优化协议只增加了一个签名的代价,对于现代计算机的处理速度以及网络的传输速度来说,这个代价是非常小的,然而换来的却是能够抵抗 KCI 攻击的安全性提升.

4 结束语

针对 TLS1.3 协议中 3 种支持 0-RTT 数据的模式,本文使用形式化分析工具 Scyther 对其进行了分析,找到了 0-RTT 数据的泄露和 KCI 攻击两种攻击,进一步提出了一种优化协议,该优化协议相比原来的 3 种模式能够抵抗 KCI 攻击和重放攻击,通过增加一个签名的微小代价,有效提高了 0-RTT 数据的安全性,特别在使用 POST 作为第 1 个请求的连接时,这样的安全性提升是必要的.

References:

[1] Freier A, Karlton P, Kocher P. The secure sockets layer (SSL) protocol version 3.0. IETF RFC-6101, 2011.
[2] Dierks T, Allen C. The TLS protocol version 1.0. IETF RFC-2246, 1999.
[3] Dierks T, Rescorla E. The transport layer security (TLS) protocol version 1.2. IETF RFC-5246, 2008.
[4] Alfardan N, Bernstein D, Paterson K, *et al*. On the security of RC4 in TLS. In: Proc. of the Usenix Conf. on Security. USENIX Association, 2013. 305−320.
[5] Fardan N, Paterson K. Lucky thirteen: Breaking the TLS and DTLS record protocols. In: Proc. of the IEEE Symp. on Security & Privacy. IEEE Computer Society, 2013. 526−540.
[6] Möller B, Duong T, Kotowicz K. This POODLE bites: Exploiting the SSL 3.0 fallback. Security Advisory, 2014.
[7] Adrian D, Bhargavan K, Durumeric Z, *et al*. Imperfect forward secrecy: How diffie-hellman fails in practice. In: Proc. of the ACM Sigsac Conf. on Computer and Communications Security. ACM, 2015. 5−17.
[8] Bhargavan K, Lavaud A, Fournet C, *et al*. Triple handshakes and cookie cutters: Breaking and fixing authentication over TLS. In: Proc. of the IEEE Symp. on Security & Privacy. IEEE Computer Society, 2014. 98−113.
[9] Gujrathi S. Heartbleed bug: An OpenSSL heartbeat vulnerability. Int'l Journal of Computer Sciences and Engineering, 2014,2(5): 61−64.
[10] Beurdouche B, Bhargavan K, Delignat-Lavaud A, *et al*. A messy state of the union: taming the composite state machines of TLS. In: Proc. of the IEEE Symp. on Security & Privacy, 2015. 535−552.
[11] Rescorla E. The transport layer security (TLS) protocol version 1.3. IETF draft-ietf-tls-tls13-20, 2017.
[12] Bhargavan K, Leurent G. Transcript collision attacks: breaking authentication in TLS, IKE and SSH. British Journal of Psychiatry the Journal of Mental Science, 2016,41(7):8−13.
[13] Jager T, Schwenk J, Somorovsky J. On the security of TLS 1.3 and QUIC against weaknesses in PKCS#1 v1.5 encryption. In: Proc. of the ACM Sigsac Conf. on Computer and Communications Security. ACM, 2015. 1185−1196.
[14] Paar C, Adrian D, Kasper E, *et al*. DROWN: Breaking TLS using SSLv2. In: Proc. of the Usenix Security Symp. USENIX Association, 2016. 689−706.
[15] Cremers C, Horvat M, Hoyland J, *et al*. A comprehensive symbolic analysis of TLS 1.3. In: Proc. of the 2017 ACM SIGSAC Conf. on Computer and Communications Security. ACM, 2017. 1773−1788.

[16] Bhargavan K, Blanchet B, Kobeissi N. Verified models and reference implementations for the TLS 1.3 standard candidate. In: Proc. of the 2017 IEEE Symp. on Security and Privacy (SP). IEEE, 2017. 483−502.
[17] Krawczyk H, Wee H. The OPTLS protocol and TLS1.3. In: Proc. of the IEEE Symp. on Security & Privacy. IEEE Computer Society, 2016. 81−96.
[18] Somorovsky J. Systematic fuzzing and testing of TLS libraries. In: Proc. of the ACM Sigsac Conf. on Computer and Communications Security. ACM, 2016. 1492−1504.
[19] 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).
[20] Cremers C. Scyther: Semantics and verification of security protocols. Netherlands: Eindhoven University of Technology, 2006.
[21] Cremers C. Key exchange in IPsec revisited: Formal analysis of IKEv1 and IKEv2. In: Proc. of the European Symp. on Research in Computer Security (ESORICS 2011). Berlin: Springer-Verlag, 2011. 315−334.
[22] Basin D, Cremers C, Meier S. Provably repairing the ISO/IEC 9798 standard for entity authentication. Journal of Computer Security, 2013,21(6):817−846.
[23] Yang H, Prinz A, Oleshchuk V. Formal analysis and model checking of a group authentication protocol by Scyther. In: Proc. of the Euromicro Int'l Conf. on Parallel, Distributed, and Network-based Processing. IEEE Computer Society, 2016. 553−557.
[24] Yang H, Oleshchuk V, Prinz A. Verifying group authentication protocols by scyther. Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications (JoWUA), 2016,7(2):3−19.
[25] Ray B, Chowdhury M, Abawajy J. Secure object tracking protocol for the Internet of things. IEEE Internet of Things Journal, 2016, 3(4):1−10.
[26] Meier S, Cremers C, Basin D. Strong invariants for the efficient construction of machine-checked protocol security proofs. In: Proc. of the IEEE Computer Security Foundations Symp. IEEE Computer Society, 2010. 231−245.
[27] Meier S, Schmidt B, Cremers C, *et al*. The TAMARIN prover for the symbolic analysis of security protocols. In: Proc. of the Int'l Conf. on Computer Aided Verification. Berlin: Springer-Verlag, 2013. 696−701.
[28] Canetti R, Krawczyk H. Analysis of key-exchange protocols and their use for building secure channels. In: Proc. of the Advances in Cryptology (EUROCRYPT 2001). Berlin: Springer-Verlag, 2001. 453−474.
[29] Günther C. An identity-based key-exchange protocol. In: Proc. of the Workshop on the Theory and Application of of Cryptographic Techniques. Berlin: Springer-Verlag, 1989. 29−37.
[30] Cha J, Cheon J. An identity-based signature from gap diffie-hellman groups. In: Proc. of the Int'l Workshop on Theory and Practice in Public Key Cryptography: Public Key Cryptography. Berlin: Springer-Verlag, 2002. 18−30.
[31] Fischlin M, Günther F. Multi-stage key exchange and the case of Google's QUIC protocol. In: Proc. of the 2014 ACM SIGSAC Conf. on Computer and Communications Security. ACM, 2014. 1193−1204.
[32] Cremers C. Formally and practically relating the CK, CK-HMQV, and eCK security models for authenticated key exchange. Cryptology Eprint Archive, 2009.
[33] Cremers C. Examining indistinguishability-based security models for key exchange protocols: The case of CK, CK-HMQV, and eCK. In: Proc. of the ACM Symp. on Information, Computer and Communications Security. ACM, 2011. 80−91.
[34] Gorantla M, Boyd C, Nieto J. Modeling key compromise impersonation attacks on group key exchange protocols. ACM Trans. on Information & System Security, 2009,14(4):1−24.

附中文参考文献:

[19] 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.

陆思奇(1990－),男,讲师,主要研究领域为密码学,信息安全,安全协议形式化分析.

毛颖(1994－),女,博士生,主要研究领域为密码学,信息安全,安全协议形式化分析.

周思渊(1990－),男,研究实习员,主要研究领域为密码学,信息安全.