

增强 TLS 1.3 中 Early data 安全性的协议

张兴隆¹, 程庆丰^{1,2}, 马建峰²

(1. 信息工程大学, 河南 郑州 450004;
2. 西安电子科技大学计算机学院, 陕西 西安 710071)

摘要: 将新型 0-RTT 密钥交换协议思想借鉴到 TLS 1.3 会话重用阶段, 构建 rFSOPKE 协议, 改进了 Early data 的加密和传输过程。rFSOPKE 协议可以在 Ticket 有效期内保护 Early data 的前向安全性并使其抵抗重放攻击。与改进前 Early data 的发送过程相比, 本协议大幅增强了 Early data 的安全性。在实现效率方面, 由于在发送 Early data 时增加了本协议的计算和传输开销, 所以实现效率有所降低。但是本协议可以根据应用场景的不同嵌入适合的算法, 所以可以选择更加高效的算法提高协议实现速度。

关键词: 0-RTT; Early data; 前向安全; 重放攻击; rFSOPKE

中图分类号: TP309

文献标识码: A

doi: 10.11959/j.issn.2096-109x.2017.00224

Protocol to enhance the security of Early data in TLS 1.3

ZHANG Xing-long¹, CHENG Qing-feng^{1,2}, MA Jian-feng²

(1. Information Engineering University, Zhengzhou 450004, China;
2. College of Computer Science, Xidian University, Xi'an 710071, China)

Abstract: The new 0-RTT Internet key exchange was drawn on the TLS 1.3 session resumption phase, the rFSOPKE protocol was constructed, and the Early data encryption and transmission process were improved. The rFSOPKE protocol can protect the forward security of Early data and protect it from replay attacks during the validity period of the Ticket. Compared with the previous Early data transmission process, rFSOPKE greatly enhanced the security of Early data. Due to the increase in the calculation and transmission overhead of this protocol when sending Early data, the efficiency of the protocol is reduced. However, rFSOPKE can embed the appropriate algorithm according to the different application environment, so more efficient algorithms should be chosen to improve the protocol implementation speed.

Key words: 0-RTT, Early data, forward security, replay attack, rFSOPKE

1 引言

传统网络协议中的认证密钥协商协议 (AKE, authenticated key exchange) 在完成正式的应用数

据传输之前需要产生很大延迟。学术界一般以往返时间 (RTT, round-trip time) 来表示这种时延。传统的高效 AKE 协议 (如 HMQV^[1]) 在完成会话密钥建立之前也需要至少 2 个消息的延迟, 即需

收稿日期: 2017-11-05; **修回日期:** 2017-11-28. **通信作者:** 程庆丰, qingfengc2008@sina.com

基金项目: 国家高技术研究发展计划 (“863” 计划) 基金资助项目 (No.2015AA016007); 密码科学技术国家重点实验室开放课题基金资助项目 (No.MMKFKT201514)

Foundation Items: The National High Technology Research and Development Program (863 Program)(No.2015AA016007) The National Key Laboratory Foundation of Cryptography (No.MMKFKT201514)

要 1-RTT。随着网络通信量的巨大增长,高延迟的 AKE 协议显然已经不能满足用户的需求了。

对连接效率的需求催生了对于 0-RTT(0-RTT, zero round-trip time) 密钥交换协议的研究。0-RTT 协议是一种允许客户端在零往返时间内发送经过加密的有效载荷的密钥交换协议。考虑到它的高效性,很多网络安全协议都将 0-RTT 纳入为一种应用模式。2015 年 7 月,谷歌为了解决浏览器高延迟问题,制定了一种低时延的网络传输协议——QUIC^[2]。QUIC 支持 0-RTT 握手并且已经在最新版本的 Google Chrome 和 Opera 网络浏览器中实施。2014 年 4 月, IETF 发布 TLS 1.3 第一个版本的草案,此后不断对各个版本进行修改完善,截至 2017 年 7 月 3 日,共有 22 个版本的草案面世。2015 年 7 月, TLS 1.3 第 7 个版本草案增添了对初始 0-RTT 的支持^[3]。

近年来,学术界对 0-RTT 协议的安全性研究呈上升态势,尤其是对 TLS 1.3 的研究更为突出。2014 年, Fischlin 和 Günther 等^[4]正式分析了 QUIC 协议的安全性,建立了一个关于多阶段密钥交换协议安全性的模型, Fischlin 和 Günther 等使用该模型对 QUIC 进行分析,包括其对 0-RTT 的支持,基本证明 QUIC 是一个充分安全的多阶段密钥交换协议。但是该模型不够完善,在该模型下无法证明 SSL/TLS 是安全的。2015 年, Krawczyk 和 Wee^[5]提出了用于 TLS 1.3 协议的密钥交换协议 OPTLS,具体给出了 OPTLS 的设计原理和密码性质分析。OPTLS 的主要目的就是保证 TLS 1.3 的“0-RTT”要求,同时使前向安全性成为强制性要求,并且使椭圆曲线成为协议的主要加密基础。为了满足这些要求,需要移除传统的以 RSA 算法为基础的设计思想。同时该协议统一和模块化的逻辑有助于协议的规范、分析、性能优化和未来维护。TLS 1.3 第 9 版本的草案(draft-ietf-tls-tls13-09) 采用了 OPTLS 框架的 4 种工作模式和密钥导出方法,但是 Krawczyk 和 Wee 并没有全面介绍 TLS 1.3 所有方面,在会话恢复和客户端认证方面并没有深入分析。2016 年, Cremers 和 Horvat 等^[6]使用形式化分析工具 Tamarin 对 TLS 1.3 第 10 版本草案^[7]进行分析,其中包括 0-RTT 握手模式,该分析无法论证 0-RTT 的安全性,反

之也证明了其存在不安全的可能。2017 年, Fischlin 和 Günther^[8]分析了 draft-ietf-tls-tls13-12 中基于 Diffie-Hellman 的 0-RTT 握手的密钥保密性和 draft-ietf-tls-tls13-14 中基于预共享密钥的 0-RTT 握手的密钥保密性。Fischlin 和 Günther 扩展了以前的安全模型,同时阐明重放攻击下 0-RTT 安全性的限制。2017 年, Hale 和 Jager 等^[9]建立了一个安全模型,提出强密钥独立性的概念。基于安全的非交互式密钥交换存在的通用假设,他们还给出了在这个模型中安全的 0-RTT 密钥交换协议的第一个结构体。总体来说,关于 0-RTT 握手消息,前向安全性和重放攻击是关于其安全性讨论的焦点。

基于层次化身份加密的前向安全思想于 2003 年由 Canetti 和 Halevi 等^[10]提出。2014 年, Pointcheval 和 Sanders^[11]提出了基于多线性映射研究非交互式密钥交换协议的前向安全问题。然而,这 2 种方法只是对粗粒度时间段提供前向保密,无法提供细粒度时间段的前向安全。2015 年, Hale 等^[12]给出了来自其他加密原语的 0-RTT 密钥交换的基础定义和通用结构。但是所有这些工作都只考虑到安全模型和结构,而不涉及 Early data 的前向安全性。2016 年, Cohn-Gordon 等^[13]在考虑使用密钥交换协议的密码安全性时,其中会话密钥在单个会话的生命周期内经常更新,可以保证细粒度时间段内的前向安全。2017 年, Günther 和 Hale 等^[14]提出了一种新的 0-RTT 密钥交换协议的构想,即利用同步时间来构建一个 0-RTT 密钥协商协议,该文章只讨论了协议架构而并未涉及具体算法。另外,近年来,关于 TLS 1.3 中 0-RTT 握手以及其他 0-RTT 密钥交换协议的研究还有很多^[15,16]。

本文借鉴了 Günther 和 Hale 等^[14]提出的新型 0-RTT 密钥交换协议的思想,将细粒度时间段内前向安全与具体网络协议结合起来,构建 rFSOPKE 协议,对 TLS 1.3 中 Early data 的加密过程和传输过程进行了改进,弥补了其在前向安全和重放攻击方面的缺陷。

2 预备知识

2.1 基于 Diffie-Hellman 密钥交换的 0-RTT

在实际的密钥交换之前,假设客户端已经与

服务器通话, 并且在该通信中获得了所谓的服务器配置。密码学上, 这个配置包括一个半静态的 Diffie-Hellman 共享 g^s , 服务器在一定时间内存储秘密指数。在 QUIC 中, 此服务器配置的身份验证是通过服务器公布的签名结构(脱机), 在 TLS 1.3 中, 它在先前的握手过程中被签名(在线)。

在随后执行的第一个消息中, 客户端发送一个短暂的 Diffie-Hellman 共享 g^x , 将 0-RTT 密钥 K_1 导出为 $K_1 = (g^s)^x = g^{xs}$, 因此可以立即发送用此密钥加密的数据。然后, 服务器计算与 g^x 相同的密钥(能够解密 0-RTT 数据), 并发送给客户端一个 g^y 来生成更强的共享密钥, 双方生成完整的加密密钥 $K_2 = g^{xy}$ 。基于 Diffie-Hellman 交换的 0-RTT 协议框架如图 1 所示。

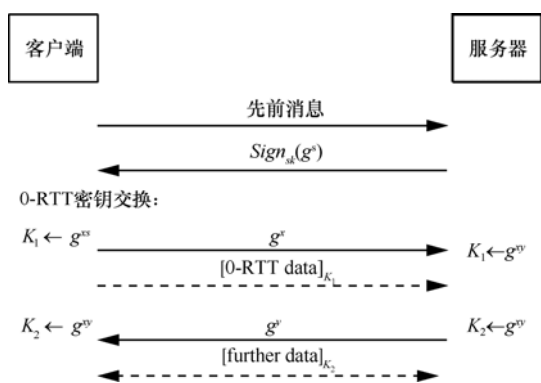


图1 基于 Diffie-Hellman 交换的 0-RTT 协议框架

2.2 基于预共享密钥的 0-RTT

在零往返时间建立密钥的另一个模式是基于预共享密钥 (PSK, pre-shared key)。从草案 13^[17]开始, 基于 Diffie-Hellman 的 0-RTT 的选项优先级被降低, PSK 模式成为 TLS 1.3 指定的 0-RTT 握手模式的基础。基于 PSK 模式下的 0-RTT 中, 0-RTT 密钥 K_1 从先前建立的密钥导出。例如, 在 TLS 1.3 中, 这个“先前建立的密钥”就是指在一般握手中为会话恢复而建立的密钥。当会话重用时, 客户端不需要与服务器进行通信就可以计算密钥, 可以用密钥 K_1 立即发送加密数据。用于后续加密消息的完整密钥 K_2 由预共享密钥和更多密钥信息导出。客户端和服务器通过更新 K_2 来保证会话安全。

2.3 当前 0-RTT 协议的缺陷

2.3.1 重放攻击

在重放攻击中, 攻击者的目的是使接收者再次接受相同的有效载荷。具体来说, 在基于

Diffie-Hellman 交换的 0-RTT 协议中重放攻击可以采取的方式为截取并重放客户端的 Diffie-Hellman 共享 g^x 或 0-RTT 数据。可以观察到, 如果没有进一步的对策, 敌手可以简单地重播记录的客户端消息 g^x , 使服务器得到与原始连接相同的密钥 K_1 , 然后重播客户端的 0-RTT 数据, 此时服务器可以正确解密消息, 因此将处理此消息, 敌手目的达成。

在 QUIC 协议中应对这种攻击的主要手段是通过存储所有 g^x , 在特定时间帧中用附加的随机数值编码, 以便检测和拒绝带相同随机数的请求。虽然这种方式可以解决重放攻击的问题, 但是该解决方案导致了大量的管理开销, 并且可以说只有某些服务器配置才可以接受。

2.3.2 前向安全性

前向安全本质上要求即使攻击者可以获取一个通信方的秘密密钥, 仍然不影响传输数据的安全性, 这已经成为现代密钥交换协议的一项标准。上述形式的 0-RTT 密钥交换不能为从客户端发送到服务器的 0-RTT 应用数据提供前向安全性。由于这样的数据是由密钥 K_1 所加密保护的, 攻击者窃取通信并随后危及服务器的秘密指数(可能在会话结束之后很长时间)可以容易地计算 K_1 , 从而解密 0-RTT 数据。QUIC 和 TLS 1.3 的设计文档中明确承认了这一缺点, 这也是升级到第二个前向密钥 K_2 的主要原因之一。值得注意的是, 不管是基于 Diffie-Hellman 密钥交换的 0-RTT 还是最新的基于 PSK 的 0-RTT, TLS 1.3 的 0-RTT 密钥的前向安全性都得不到保证^[18,19]。

2.4 TLS 1.3 会话重用流程

TLS 1.3 的主要改进如下(截至本文尚处在 draft-21)。

- 1) 在密钥协商方面禁用 RSA 算法, 使用 AEAD 算法^[20]。
- 2) 弃用 PRF 算法^[21], 使用 HKDF 密钥导出算法^[22]。
- 3) 移除更改密码规范协议。
- 4) 对会话重用机制进行完善, 将 PSK 的机制作为主要模式。
- 5) 在 Session ticket 中添加了过期时间。
- 6) 支持 0-RTT Early data 发送。

完整的 TLS 1.3 握手的消息流程如图 2 所示。客户端在 Client hello 中发送 Key_share,

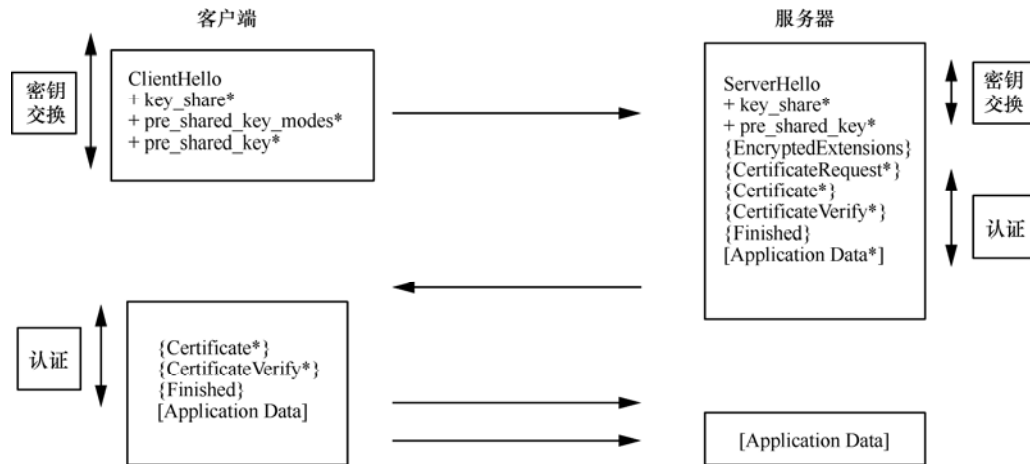


图 2 完整 TLS1.3 握手的消息流程

Key_share 包括每个公钥和对应的曲线。服务器端接收 Key_share 之后，生成 ECDHE 公私钥，然后协商出密钥 ECDHE secret。在此之后，服务器端同样在 Server hello 的拓展中发送 Key_share。

TLS 1.3 的一个变化是规定 Server hello 之后的握手消息需要加密。握手加密密钥的导出过程是通过 Early secret 和 ECDHE secret 导出 Server_handshake_traffic_secret，再从 Server_handshake_traffic_secret 中导出 key 和 iv，使用该 key 和 iv 对之后的消息进行加密。当会话结束，发送 Finished 消息时，服务器通过 Server_handshake_traffic_secret 导出 Server finished key，使用 Hmac 计算 Finished 消息后发送给客户端。导出对称密钥的过程如图 3 所示。

Finished 消息发送完后，客户端和服务端分别通过主密钥 Master secret 和整个握手消息的摘要计算用于会话重用的 Resumption secret。此后服务器发送一个 New session ticket 消息，该消息包含整个会话信息，使用 Server_application_

traffic_secret 加密。

收到 New session ticket 消息后，客户端将收到的 Ticket 和本地发送 Finished 后计算的 Resumption secret 组成 PSK (pre shared key)，保存在本地缓存中。当会话重用时，客户端在本地缓存中查找 Server Name 对应的 PSK，然后在 Client hello 消息中将 PSK 发送至服务器端。会话重用查找 PSK 过程如图 4 所示。

发送 Client hello 后，客户端使用 Resumption secret 导出的 Client_early_traffic_key 和 iv，对 Early data 进行加密后发送。

2.5 Günther 等所提出的 0-RTT 协议

2017 年，Günther 等^[14]提出了一种新的 0-RTT 密钥交换协议的构想，即利用同步时间来构建一个 0-RTT 密钥协商协议，只讨论了协议架构而未涉及具体算法。

Günther 等指出，为解决 0-RTT 密钥协商协议前向安全和重放攻击的问题，关键思想不是修改协议，而是修改服务器处理和存储密钥的方式。

```
MasterSecret = hkdfExtract(hash, nil, Handshake Secret)
Client_application_traffic_secret_0 = Derive-Secret(masterSecret, "client application
traffic secret", ClientHello...server Finished)
Server_application_traffic_secret_0 = Derive-Secret(masterSecret, "server application
traffic secret", ClientHello...server Finished)
```

图 3 导出对称密钥过程

```
Early Secret = HKDF-Extract(0, resumption secret)
Binder_key = deriveSecret(Early Secret, "resumption psk binder key", "")
Binder_macKey = hkdfExpandLabel(ctx.params.hash, binder_key, "finished", [byte{}], ctx.
params.hash.Size())
EarlyTrafficSecret = ctx.deriveSecret(Early Secret, "client early traffic secret", clientHello)
EarlyExporterSecret = ctx.deriveSecret(Early Secret, "early exporter master secret", ClientHello)
ClientEarlyTrafficKey, clientEarlyTrafficIv = ctx.makeTrafficKeys(earlyTrafficSecret)
```

图 4 会话重用查找 PSK 过程

考虑一个具有长期公私钥对 (pk, sk) 的服务器。客户端使用公钥 pk 加密会话密钥, 然后将密文 c_1 发送至服务器。当在密文 c_1 中接收到封装的会话密钥时, 服务器可以使用该方案进行如下操作。

1) 用 sk 解密 c_1 。

2) 服务器用 sk 导出密钥 $sk \setminus c_1$, $sk \setminus c_1$ 可以用来解密除了 c_1 的所有密文。

3) 服务器删除 sk 。

服务器在接收到其他密文时重复这样的操作, 即当服务器接收到 c_2 时, 从密钥 $sk \setminus c_1$ 导出 $sk \setminus c_1, c_2$, 该密钥只能解密除了 c_1, c_2 之外的所有密文。

这是可以保证前向安全的。假如敌手获取了会话密钥 $sk \setminus c_1, c_2$, 但是他无法通过此密钥解密 c_1, c_2 , 也就无法解密之前的所有消息。

作者为了解决密钥长度随着会话数量线性增长从而影响效率的问题, 提出了增加一个时间元件, 这个时间元件要求客户端和服务端维护一个同步时钟。在每个时间间隔内, 密钥长度线性增长, 但是在一个时间段结束时, 服务器可以“清空”密钥, 就是说将密钥长度减少到时间间隔的对数因子。

为了实现上述操作, 构建了一个特殊的可穿透的前向安全的密钥封装算法 (PFSKEM, puncturable forward-secret key encapsulation)。

定义 1 PFSKEM: 一个 PFSKEM 方案由以下概率性多项式时间算法组成。

1) $PFSKEM.KGen(1^\lambda)$: 输入安全参数 λ , 输出公钥 PK , 初始私钥 SK 。

2) $PFSKEM.Enc(PK, \tau)$: 输入公钥和时间间隔 τ , 输出密文 CT 和对称密钥 K 。

3) $PFSKEM.PunctCxt(SK, \tau, CT)$: 输入当前私钥 SK , 时间间隔 τ 和密文 CT , 输出一个新的私钥 SK' 。

4) $PFSKEM.Dec(SK, \tau, CT)$: 输入当前私钥 SK , 时间间隔 τ 和密文 CT , 输出一个对称密钥 K 或者是 O (当算法失败时)。

5) $PFSKEM.PunctInt(SK, \tau)$: 输入私钥 SK , 时间间隔 τ , 输出 $\tau+1$ 时的私钥 SK' 。

在此之前, 构建了一个一次性签名 (OTSIG, one-time signatures) 算法和一个层次化的基于身份的密钥封装系统 (HIBKEM, hierarchical identity-based key encapsulation scheme)。基于以上算法构建了一种前向安全的 0-RTT 密钥交换协

议 (FSOPKE, forward-secret one-pass key exchange)。

定义 2 FSOPKE: 一个支持 $\tau_{\max}$ 时间并且提供双向认证 (或者仅认证服务器) 的 FSOPKE 协议, 该协议包括以下 4 个算法。

1) $FSOPKE.KGen(1^\lambda, r, \tau_{\max}) \rightarrow (pk, sk)$ 。输入安全参数 $1^\lambda, r \in \{\text{client}, \text{server}\}, \tau_{\max} \in \mathbb{N}$, 输出为指定身份的公私钥对 (pk, sk) 。

2) $FSOPKE.RunC(sk, pk) \rightarrow (sk', k, m)$ 。输入私钥 sk 和公钥 pk , 输出为修改后的密钥 sk' , 会话密钥 $k \in \{0, 1\}^* \cup \{O\}$ 和消息 $m \in \{0, 1\}^* \cup \{O\}$ 。

3) $FSOPKE.RunS(sk, pk, m) \rightarrow (sk', k)$ 。输入私钥 sk 和公钥 pk , 以及消息 $m \in \{0, 1\}^*$, 输出为修改后的密钥 sk' , 会话密钥 $k \in \{0, 1\}^* \cup \{O\}$ 。

4) $FSOPKE.TimeStep(sk, r) \rightarrow sk'$ 。输入私钥 sk 和身份 $r \in \{\text{client}, \text{server}\}$, 输出是修改后的私钥 sk' 。

Günther 等采用可证明安全的思想对协议的正确性和安全性给出了证明, 证明了该协议满足前向安全并且可以抵抗重放攻击。

3 TLS 1.3 中 Early data 的安全性缺陷

分析 TLS 1.3 会话重用流程, 注意到在 Ticket 的有效期限内, Early data 无法抵抗重放攻击, 同时也不具有前向安全性。进一步查阅资料^[17]得知, TLS 1.3 的 Ticket 有效期限定在一周, 这对攻击者来说是有足够操作时间的。

1) 无法抵抗重放攻击

假如攻击者在同一个 Ticket 的有效期限内截获了会话重用时的 Client hello 消息, 并向服务器发送该消息从而进行重放攻击, 收到重放消息的服务器检查 Client hello 的 PSK 拓展, 将 Ticket 解密, 在验证 Ticket 未过期之后, 验证 HMAC 并且检查 Binder 是否正确。一般情况下, 这 2 个步骤都不会出现问题。

此后服务器通过 Ticket 中的 Resumption secret 导出 Early data 的解密密钥。至此, 服务器对 Ticket 以及相应拓展的验证通过解密密钥可以正确解密, 所以服务器正常处理消息, 敌手重放攻击目的达成。

2) 不具有前向安全性

假如攻击者在某一次服务器解密 Early data

时获取了 Early data 的解密密钥，那么由于在一个 Ticket 有效期内解密密钥没有发生变化，在这个 Ticket 有效期内的所有 Early data 都可以被敌手解密，无法满足前向安全性。

4 改进 TLS 1.3 会话重用阶段 Early data 密钥

本文考虑将 Günther 等所提出的 0-RTT 密钥协商协议 FSOPKE 应用于 TLS 1.3 协议的会话重用阶段，构建重用阶段 0-RTT 协议 rFSOPKE。首先设置客户端和服务端时钟同步。rFSOPKE 在 TLS 1.3 会话重用阶段运行如下。

1) 服务器发送 New session ticket，携带全部的 Session 信息和 $(l^1, \tau_{\max})$ ，即安全参数和最大时间间隔，其中 $\tau_{\max} \leq T$ (T 为 Ticket 有效期)。New session ticket 使用 Server_application_traffic_secret 加密。

2) 客户端和服务端运行 rFSOPKE.KGen 算法， $\text{rFSOPKE.KGen}(l^1, r, \tau_{\max}) \rightarrow (pk, sk)$ ，得到各自的公私钥对 (pk_c, sk_c) 和 (pk_s, sk_s) 。

3) 初始阶段客户端运行 rFSOPKE.RunC 算法， $\text{rFSOPKE.RunC}(sk, pk) \rightarrow (sk', k, m)$ ，得到修改后的密钥 sk' ，会话密钥 k 和消息 m 。

4) 客户端在 Client hello 消息的 PSK 拓展中附带消息 m ，服务器收到之后运行 rFSOPKE.RunS(sk, pk, m) 算法生成修改后的密钥 sk' ，会话密钥 k 。

5) 当时间 $\tau > \tau_{\max}$ 时，双方都执行 rFSOPKE.TimeStep 算法，生成修改之后的私钥 sk' 。

下面考虑 Early data 的发送过程。

会话重用时，客户端查找 PSK，然后在 Client hello 消息的 PSK 拓展中带上消息 m 。发送 Client hello 后，客户端使用会话密钥 k 对 Early data 加密后发送。导出会话密钥 k 的过程如图 5 所示。

$$\begin{array}{l} \text{rFSOPKE.KGen}(l^1, r, \tau_{\max}) \rightarrow (pk, sk) \\ \text{rFSOPKE.RunC}(sk, pk) \rightarrow (sk', k, m) \end{array}$$

图 5 导出 Early data 密钥的过程

此后使用对称密钥 k 加密 Early data 并发送至服务器端。在接收到第一个 Early data 时，服务器使用私钥 sk' 进行解密可得 Early data。

在 Ticket 有效期内，客户端每发送一次 Early data，其运行 rFSOPKE.RunC 算法对会话密钥更新一次；服务器运行 rFSOPKE.RunS 算法对私钥进行更新一次，同时得到相同的会话密钥。当 Ticket 过期时，客户端和服务端执行 rFSOPKE.TimeStep 算法生成各自新的私钥，然后重复步骤 2)~步骤 4)。直到客户端发送 End_of_early_data alert，标志着 Early data 发送结束，rFSOPKE 协议终止。

5 安全性及效率分析

5.1 安全性分析

假定攻击者 A 掌控网络，并且有任意篡改、窃听和重新排列消息顺序的能力。他通过以下查询与 rFSOPKE 协议进行交互。

1) $\text{NewSession}(u, \text{role}, \text{pid}, m)$ ：初始化会话身份 $u \in I$ ， $\text{role} \in \{\text{client}, \text{server}\}$ ，指定的会话搭档 $\text{pid} \in I \cup \{O\}$ ，如果 $\text{role} \neq \text{server}$ ，要求 $m = O$ 。

如果 $\text{role} = \text{client}$ ，调用 $\text{rFSOPKE.RunC}(sk_u, pk_{\text{pid}}) \rightarrow (sk_u, k, m)$ ，否则，调用 $\text{rFSOPKE.RunS}(sk_u, pk_{\text{pid}}, m) \rightarrow (sk_u, k)$ 。

注册新会话 $\pi_u^i (\text{role} = \text{role}, \text{id} = u, \text{pid} = \text{pid}, \text{trans} = m, \text{time} = \tau_u, \text{key} = k)$ ，如果 $\text{role} = \text{client}$ ，返回 m ，如果 $\text{role} = \text{server}$ ，返回 O 。

2) $\text{Reveal}(\pi_u^i)$ ：显示特定会话的会话密钥。

如果 $\pi_u^i.\text{key} \neq O$ ，显示密钥状态 $\pi_u^i.\text{keystate}$ ，并且返回密钥 key ，否则返回 O 。

3) $\text{Corrupt}(u)$ ：腐化身份 u 的长期状态，这个查询对每一个身份最多只能进行一次，并且在进行过此查询之后不可以再进行别的任何查询。

假设 $\text{Corrupt}(u)$ 是 A 进行的第 s 次查询，令 $s \rightarrow s_u^{\text{corr}}$ ，未腐化身份 $s_u^{\text{corr}} = \infty$ 。同样地，腐化身份时记录身份的当前时间 τ_u 并且设置 $\tau_u \rightarrow \tau_u^{\text{corr}}$ 。返回 sk_u 。

4) $\text{Tick}(u)$ ：通过 $\text{rFSOPKE.TimeStep}(sk_u) \rightarrow sk_u$ 转发身份状态并且记录时间 $\tau_u \rightarrow \tau_u + 1$ 。

5) $\text{Test}(\pi_u^i)$ ：允许敌手挑战导出的会话密钥，但是只能挑战一次。预言给出秘密比特 $b_{\text{test}} \in \{0, 1\}$ 供敌手在安全游戏中随机选择。

定理 rFSOPKE 协议的安全性：多项式时间敌手与 rFSOPKE 协议通过以上查询在以下游戏

$G_{A,rFSOPKE}^{rFSOPKE-sec}$ 中进行交互。

1) 挑战者为所有身份 $u \in I$ 生成密钥和状态 $rFSOPKE.KGen(1^\lambda) \rightarrow (pk_u, sk_u)$ ，并且随机选择 $b_{test} \in \{0,1\}$ 。

2) 敌手接收到所有用户的 (u, pk_u) ，并且可以实用查询 NewSession、Reveal、Corrupt、Tick 以及 Test。记录 Test 查询，即 s^{test} 次查询和查询的会话 π^t 。

3) 最终，敌手 A 停止查询并且输出猜测比特 $b \in \{0,1\}$ 。

挑战者输出 1（用 $G_{A,rFSOPKE}^{rFSOPKE-sec} = 1$ 表示），当 $b = b_{test}$ 并且以下条件满足时，则认为敌手在游戏中获胜。

1) $\pi^t.keystate = fresh$ ： A 还没有对会话进行 Reveal 查询。

2) $\pi_v^j.keystate = fresh$ （其中 π_v^j 和 π^t 是通信双方）： A 还没有对会话进行 Reveal 查询。

3) $s_u^{corr} > s^{test}$ ：会话持有者在 Test 查询进行前还没有被腐化。

4) 如果 $\pi^t.role = client$ ，对于 $v = \pi^t.pid$ ，有 $s_v^{corr} \neq \infty$ ，那么以下条件必须满足：存在与 π^t 相关联的会话 π_v^j ； $\tau^t < \tau_v^{corr}$ 。

5) 如果 $\pi^t.role = server, \pi^t.pid \neq O$ ，那么对于 $v = \pi^t.pid$ ， $s_v^{corr} > s^{test}$ ：指定服务器会话的客户端在 Test 查询之前还没有被腐化。

6) 如果 $\pi^t.role = server, \pi^t.pid = O$ ，存在与 π^t 相关联的会话 π_v^j ：必须至少存在一个诚实通信方以用来测试会话 π^t 。

按照 Günther 文章要求，满足攻击者优势 $Adv_{A,rFSOPKE}^{rFSOPKE-sec}(\lambda) := \left| \Pr[G_{A,rFSOPKE}^{rFSOPKE-sec} = 1] - \frac{1}{2} \right|$ 是可忽略的，定理得证。

5.1.1 前向安全和后向安全

在会话重用阶段运行 rFSOPKE 协议后，假如攻击者 A 获取当前会话密钥 k ，由于 A 无法得到服务器私钥，故无法运行 $rFSOPKE.RunS(sk, pk, m)$ 获取更新的会话密钥 k' ，解密接下来的 Early data，所以 Early data 满足后向安全性。同样，攻击者 A 也无法通过 $rFSOPKE.RunS$ 算法的逆算法得到此前任意阶段的会话密钥 $\bar{k}$ ，从而无法解密之前的所有消息。所以 Early data 满足前

向安全性。

5.1.2 重放攻击

假如敌手截获了某一次客户端发送的加密的 Early data 和其会话密钥 k ，由于此消息在之前某个阶段已经被服务器处理过了，根据服务器的设置，若攻击时间在一个 Ticket 有效期内，由于更新密钥之后的服务器无法解密重放消息，所以不会处理消息，重放攻击无法实现。若攻击时间已经超出 Ticket 有效时间，那么根据 TLS 1.3 的设置，服务器不会处理过期的消息，所以重放攻击也不会成功。

结合以上分析，将本文改进后的协议与改进前的协议对比，如表 1 所示。

表 1 Early data 安全性对比

协议	保密性	不可否认性	前向安全	抵抗重放攻击
改进前	√	√	×	×
改进后	√	√	√	√

5.2 效率分析

rFSOPKE 协议可以适应嵌入不同密钥交换以及加密签名算法，其计算开销也随着不同算法的应用而有所不同。故对协议效率的讨论主要体现在原本 TLS 1.3 会话重用时所耗时的情况。在准备发送 Early data 时，Resumption secret 导出加密密钥需要使用密钥导出算法；从改进的 Early data 发送阶段可以看出，Early data 的发送不再依赖之前握手所提供的密钥的导出密钥，这就节省了大量的运算时间。假设密钥导出函数使用 HKDF-SHA256，计算时间为 t_{HKDF} ；使用基于离散对数的一次性签名^[23]，签名时间设为 t_{Sig} ；加密体制采用基于 DDH 的 HIBE 体制^[24]，占用时间设为 t_{HIBE} 。

在 rFSOPKE 协议参与 Early data 发送时，增加的计算开销为 $t_{Sig} + t_{HIBE} - 2t_{HKDF}$ ，查阅资料后可知，增加的额外开销相对较大。但是本协议所增加的安全性是十分可观的，而且对于大量数据的传送具有优势，具体见表 2。考虑到一些特殊的安全性需求高的环境，计算开销的增加是可以忍受的。

6 结束语

本文致力于解决 TLS 1.3 协议会话重用时

表 2 公钥和密文空间的增量

λ	$ pk $	$ c $	S	$ sk $
80	7.8 kB	160 B	2^{10}	145.9 kB
80	7.8 kB	160 B	2^{16}	7.88 MB
80	7.8 kB	160 B	2^{20}	125.9 MB
128	18.62 kB	256 B	2^{10}	251.9 kB
128	18.62 kB	256 B	2^{16}	12.64 MB
128	18.62 kB	256 B	2^{20}	201.4 MB
256	70.02 kB	512 B	2^{10}	623.3 kB
256	70.02 kB	512 B	2^{16}	26.17 MB
256	70.02 kB	512 B	2^{20}	417 MB

Early data 在同一 Ticket 时间间隔内无法满足前向安全和无法抵抗重放攻击的问题，分析 Günther 等提出的新型 0-RTT 密钥交换协议，将其思想借鉴到 TLS 1.3 会话重用阶段，构造了 rFSOPKE 协议。通信双方运行此协议可以有效改进 Early data 的加密和传输过程，保护了 Early data 的前向安全性并且可以抵抗重放攻击。下一步的工作将围绕寻找适应的协议嵌入算法达到安全性和高效性的平衡展开。

参考文献：

[1] KRAWCZYK H. HMQV: a high-performance secure Diffie-Hellman protocol[C]//The International Cryptology Conference. 2005: 546-566.

[2] CUI Y, LI T, LIU C, et al. Innovating transport with QUIC: design approaches and research challenges[J]. IEEE Internet Computing, 2017, 21(2): 72-76.

[3] RESCORLA E. The transport layer security (TLS) protocol version 1.3-draft-ietf-tls-tls13-07[DB/OL].<https://tools.ietf.org/html/draft-ietf-tls-tls13-07>.

[4] FISCHLIN M, GÜNTHER F. Multi-stage key exchange and the case of Google's QUIC protocol[C]//The ACM CCS. 2014: 1193-1204.

[5] KRAWCZYK H, WEE H. The OPTLS protocol and TLS 1.3[C]//The IEEE European Symposium on Security and Privacy. 2016:81-96.

[6] CREMERS C, HORVAT M, SCOTT S, et al. Automated analysis and verification of TLS 1.3: 0-RTT, resumption and delayed authentication[C]// Security and Privacy. 2016:470-485.

[7] RESCORLA E. The transport layer security (TLS) protocol version 1.3-draft-ietf-tls-tls13-10[DB/OL].<https://tools.ietf.org/html/draft-ietf-tls-tls13-10>.

[8] FISCHLIN M, GÜNTHER F. Replay attacks on zero round-trip time: the case of the TLS 1.3 handshake candidates[C]//IEEE European Symposium on Security and Privacy. 2017:82-113.

[9] HALE B, JAGER T, LAUER S, et al. Simple security definitions for and constructions of 0-RTT key exchange[C]// The International Conference on Applied Cryptography and Network Security. 2017: 20-38.

[10] CANETTI R, HALEVI S, KATZ J. A forward-secure public-key encryption scheme[J]. Journal of Cryptology, 2007, 20(3): 265-294.

[11] POINTCHEVAL D, SANDERS O. Forward secure non-interactive key exchange[C]//The International Conference on Security and Cryptography for Networks . 2014: 21-39.

[12] HALE B, JAGER T, LAUER S, et al. Simple security definitions for and constructions of 0-RTT key exchange[C]//The International Conference on Applied Cryptography and Network Security. 2017: 20-38.

[13] COHN-GORDON K, CREMERS C, GARRATT L. On post-compromise security[C]//Computer Security Foundations Symposium. 2016: 164-178.

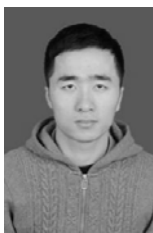
[14] GÜNTHER F, HALE B, JAGER T, et al. 0-RTT key exchange with full forward secrecy[C]//The International Conference on the Theory and Applications of Cryptographic Techniques. 2017: 519-548.

[15] DELIGNAT-LAVAUD A, FOURNET C, KOHLWEISS M, et al. Implementing and proving the TLS 1.3 record layer[C]//The 2017 Security and Privacy. 2017: 463-482.

[16] BHARGAVAN K, BLANCHET B, KOBEISSI N. Verified models and reference implementations for the TLS 1.3 standard candidate[C]//The 2017 Security and Privacy. 2017: 402-422.

- [17] RESCORLA E. The transport layer security (TLS) protocol version 1.3-draft-ietf-tls-tls13-13[DB/OL].<https://tools.ietf.org/html/draft-ietf-tls-tls13-13>.
- [18] RESCORLA E. The transport layer security (TLS) protocol version 1.3-draft-ietf-tls-tls13-12[DB/OL]. <https://tools.ietf.org/html/draft-ietf-tls-tls13-12>.
- [19] RESCORLA E. The transport layer security (TLS) protocol version 1.3-draft-ietf-tls-tls13-18[DB/OL]. <https://tools.ietf.org/html/draft-ietf-tls-tls13-18>.
- [20] KRAWCZYK H. The order of encryption and authentication for protecting communications (or: how secure is SSL?)[C]// CRYPTO. 2001: 310-331.
- [21] GOLDREICH O, GOLDWASSER S, MICALI S. How to construct random functions[C]//Foundations of Computer Science. 1984: 464-479.
- [22] KRAWCZYK H. Cryptographic extraction and key derivation: the HKDF scheme[C]//CRYPTO. 2010:631-648.
- [23] GROTH J. Simulation-sound NIZK proofs for a practical language and constant size group signatures[C]//The International Conference on Theory and Application of Cryptology and Information Security. 2006: 444-459.
- [24] BLAZY O, KILTZ E, PAN J. (Hierarchical) Identity-based encryption from affine message authentication[C]//The International Cryptology Conference. 2014: 408-425.

作者简介:



张兴隆 (1994-), 安徽滁州人, 信息工程大学硕士生, 主要研究方向为密码学和信息安全。



程庆丰 (1979-), 辽宁朝阳人, 博士, 信息工程大学副教授, 主要研究方向为密码学和信息安全。



马建峰 (1963-), 男, 陕西西安人, 西安电子科技大学教授、博士生导师, 主要研究方向为密码学、无线和移动安全。