软件学报 ISSN 1000-9825, CODEN RUXUEW
[doi: 10.13328/j.cnki.jos.007614] [CSTR: 32375.14.jos.007614]

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

TLS 1.3 协议安全研究综述*

柏军洋[1], 张 宇[1,2], 张 宾[1], 张伟哲[1,2]

[1](鹏城实验室, 广东 深圳 518055)

[2](哈尔滨工业大学 网络空间安全学院, 黑龙江 哈尔滨 150001)

通信作者: 张宇, E-mail: yuzhang@hit.edu.cn

摘 要: TLS 协议在保障网络通信的隐私性、完整性和可靠性方面发挥着至关重要的作用. 近年来, 工业界和学术界积极推动 TLS 协议相关研究和发展, 尤其在 TLS 1.3 中取得了巨大进展. 然而, 随着网络环境的日益复杂化和攻击手段的不断进步, TLS 1.3 的安全性也面临着严峻挑战, 如重放攻击、前向安全风险, 以及 OpenSSL 软件实现漏洞. 这些攻击和漏洞不仅严重威胁用户隐私与企业数据安全, 也对互联网的信任体系乃至整个社会的数字经济产生深远影响. 首先对 TLS 1.3 协议发展历程和原理进行详细介绍. 其次, 对 TLS 1.3 协议相关安全研究进行分类梳理和对比分析, 从协议机制、软件实现和应用配置这 3 个方面系统分析和归纳. 最后, 总结 TLS 1.3 协议安全研究现状和瓶颈挑战, 为未来的研究方向提供建议.

关键词: TLS 1.3; 握手安全; 0-RTT; 形式化分析; 模糊测试

中图法分类号: TP393

中文引用格式: 柏军洋, 张宇, 张宾, 张伟哲. TLS 1.3协议安全研究综述. 软件学报. http://www.jos.org.cn/1000-9825/7614.htm

英文引用格式: Bai JY, Zhang Y, Zhang B, Zhang WZ. Survey on Security Research of TLS 1.3 Protocol. Ruan Jian Xue Bao/Journal of Software (in Chinese). http://www.jos.org.cn/1000-9825/7614.htm

Survey on Security Research of TLS 1.3 Protocol

BAI Jun-Yang[1], ZHANG Yu[1,2], ZHANG Bin[1], ZHANG Wei-Zhe[1,2]

[1](Pengcheng Laboratory, Shenzhen 518055, China)

[2](School of Cyberspace Science, Harbin Institute of Technology, Harbin 150001, China)

Abstract: TLS protocol plays a critical role in ensuring the privacy, integrity, and reliability of network communications. In recent years, both industry and academia have actively promoted research and development related to TLS protocol, achieving significant progress, particularly in TLS 1.3. However, with the increasing complexity of network environments and the continuous evolution of attack methods, the security of the TLS 1.3 protocol also faces severe challenges, including replay attacks, risks to forward secrecy and vulnerabilities in software implementations such as OpenSSL. These attacks and vulnerabilities not only pose serious threats to user privacy and enterprise data security but also profoundly impact the trust system of the Internet and the broader digital economy. This study first provides a detailed introduction to the development and underlying principles of TLS 1.3. Subsequently, recent security research on the TLS 1.3 protocol is systematically categorized, compared, and analyzed from three aspects: protocol mechanism, software implementation, and application configuration. Finally, the current state and challenges in the TLS 1.3 protocol security research are summarized, and suggestions for future research directions are provided.

Key words: TLS 1.3; handshake security; 0-RTT; formal analysis; fuzzing

TLS (transport layer security) 协议是应用程序间安全通信的基础协议, 在保障网络通信的隐私性、完整性和可靠性方面发挥着关键作用. 随着网络技术的不断发展, TLS 协议在移动网、物联网、车联网以及算力网安全传

* 基金项目: 国家重点研发计划 (2024YFB31NL00105)
收稿时间: 2025-08-15; 修改时间: 2025-09-30, 2025-11-20; 采用时间: 2025-12-09; jos 在线出版时间: 2026-04-09

输中扮演着越来越重要的角色. 根据 W3Techs 权威统计数据[1], 截至 2024 年 7 月, 互联网排名前 100 万的网站中有 92% 使用了 HTTPS (hypertext transfer protocol secure) 协议[2], 4.6% 使用了 QUIC (quick UDP Internet connections) 协议[3].

TLS 作为 HTTPS 和 QUIC 等应用层协议的安全传输基础, 受到了国内外学者的广泛关注. 2014 年, IETF (Internet Engineering Task Force) 正式提出 TLS 1.3 协议第 1 版草案, 并于 2018 年发布了正式标准. 在此期间, 工业界和学术界积极推动 TLS 1.3 相关研究和发展, 围绕草案进行大量安全分析和证明, 通过协议安全分析模型对草案中的握手机制和密钥安全性进行严格的理论分析和形式化证明, 不断改进草案中的握手和会话安全性, 在握手效率和安全增强方面取得了显著突破, 极大地促进了其实际应用. SSLLabs 调查显示[4], HTTPS 网站中有 99.9% 支持 TLS 1.2[5], 有 70.1% 已部署 TLS 1.3[6].

然而, 随着网络环境的日益复杂化和攻击手段的不断进步, TLS 1.3 安全性也面临着严峻挑战, 如协议机制方面存在 0-RTT 重放攻击[7]和前向安全风险[8], 软件实现 (如 OpenSSL) 方面存在缓冲区溢出及拒绝服务漏洞[9], 应用配置方面存在降级攻击[10,11]、重协商攻击[12]. 这些攻击和漏洞不仅严重威胁用户隐私与数据安全, 也对互联网的信任体系乃至整个社会的数字经济产生深远影响. 降低 TLS 1.3 协议安全性、软件健壮性所带来的风险和损失, 及时检测和修复 TLS 1.3 协议机制缺陷及软件实现漏洞, 是网络空间安全的基础性研究课题.

目前已经有一些研究对 TLS 协议安全进行了综述, 如表 1 所示. 文献 [13] 对 TLS 1.2 协议版本存在的设计问题和实现缺陷进行了系统综述. 文献 [14] 对 TLS 1.3 协议 21 个草案版本的特性及相关研究进展进行了分析, 并指出现有研究存在的不足, 如缺乏协议实现和攻击方面的研究, 强调 0-RTT 安全及记录协议分析等是未来研究方向. 文献 [15] 对 TLS 劫持技术进行了全面分析和对比. 首先探讨了 TLS 1.3 对于 HTTP 协议明文传输的安全保护意义; 然后对 TLS 劫持技术的目的和动机进行了剖析和分类, TLS 劫持主要通过中间人 (man-in-the-middle, MITM) 来攻击 TLS 会话密钥和证书以绕过 TLS 安全认证, 并从法律、安全隐私、性能等方面对现有的 TLS 劫持技术进行了详细对比; 最后对 TLS 劫持技术未来演进方向进行了展望. 文献 [16−18] 对现有的网络协议漏洞挖掘技术进行系统地分析、总结和比较, 结合协议智能模糊测试、混合模糊测试、符号化分析技术对国内外相关研究工作进行归纳和梳理, 分析了现有漏洞挖掘技术的能力现状和不足, 并展望了协议软件漏洞挖掘技术的发展和潜在方向.

表 1 TLS 相关综述对比

综述文献	关注层面	研究内容	覆盖范围	分析方法	主要结论
[13]	协议机制和软件实现	机制设计缺陷和软件实现缺陷	TLS 1.2	对相关论文进行归纳分类	TLS 1.2存在设计缺陷和实现漏洞
[14]	协议机制	协议机制原理和进展	TLS 1.3草案	对相关论文和协议草案版本进行对比分析	TLS 1.3 0-RTT握手引入了新的安全问题, 值得重点关注
[15]	应用配置	协议攻击模型(中间人劫持)	TLS 1.3	对TLS劫持攻击相关技术和工具进行系统对比分析	TLS 1.3给网络监管和内容合规等带来了新挑战
[16]	软件实现	模糊测试方法	TLS 1.2	对现有的有状态协议模糊测试工具进行对比分析	TLS协议具有高安全性和高复杂性, 单一的模糊测试技术无法全面覆盖TLS协议
[17]	软件实现	漏洞挖掘方法	TLS 1.2	对现有的网络协议漏洞挖掘技术进行系统对比分析	现有漏洞挖掘方法存在挖掘效率低、可扩展性差问题, TLS软件漏洞检测能力较弱
[18]	软件实现	软件测试方法	TLS 1.2–TLS 1.3	对TLS协议软件测试方法进行系统性梳理、评估	当前的TLS软件自动化测试方法呈现方法异质性, 局限于证书验证等特定组件

上述综述主要从协议进展、劫持攻击、软件缺陷或者协议模糊测试的角度对 TLS 1.3 之前版本或草案进行归纳分析, 为理解 TLS 协议安全机制与挑战提供了重要基础, 但倾向于单一层面或特定技术路径的研究分析, 缺乏针对 TLS 1.3 协议的系统而全面的综述分析. 本文从协议机制、软件实现、应用配置这 3 个层面对 TLS 1.3 安全相关研究进行系统梳理和深入剖析 (如图 1 所示), 以帮助研究人员更透彻地了解 TLS 1.3 从协议草案设计到标准建立, 再到代码实现和配置应用过程中存在的各种安全风险和检测工具, 以及更全面地了解该领域的最新进展和安全研究现状, 并为未来的研究提供明确方向.

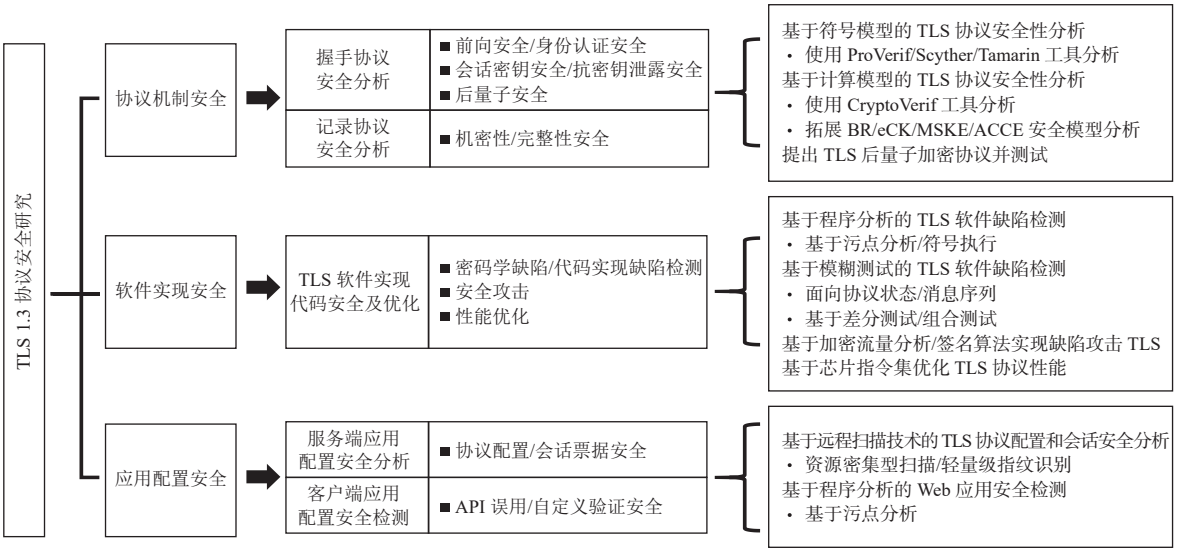


图 1　TLS 1.3 协议安全研究

本文第 1 节介绍 TLS 1.3 协议的发展历史, 对各草案版本的重要安全特性进行简要概述. 第 2 节介绍 TLS 1.3 协议结构、协议基本原理, 阐述了 0-RTT 数据传输机制以及 0-RTT 机制存在的安全风险. 第 3 节主要介绍 TLS 1.3 协议机制安全分析研究进展. 首先阐述了协议威胁模型及安全属性, 然后简单介绍了目前业界流行的协议安全分析模型和分析流程, 并对现有的协议安全分析相关文献进行了系统梳理和归类. 第 4 节介绍 TLS 1.3 协议软件缺陷检测相关研究, 主要对基于程序分析和模糊测试的缺陷检测技术进行详细分析和对比. 第 5 节介绍 TLS 1.3 协议配置及应用安全. 第 6 节对 TLS 1.3 协议安全研究发现及现状进行总结, 并展望未来的研究方向. 为方便读者阅读, 本文将英文缩写汇总如表 2 所示.

表 2　本文中的英文缩写及说明

英文缩写	英文全称	中文名称
FTP	File transfer protocol	文件传输协议
HTTP	Hypertext transfer protocol	超文本传输协议
HTTPS	Hypertext transfer protocol secure	安全超文本传输协议
QUIC	Quick UDP Internet connections	快速UDP网络连接
TCP	Transmission control protocol	传输控制协议
UDP	User datagram protocol	用户数据报协议
IP	Internet protocol	网际互连协议
SSL	Secure socket layer	安全套接层
TLS	Transport layer security	传输层安全
DTLS	Datagram transport layer security	数据包传输层安全
IETF	Internet engineering task force	互联网工程任务组
MITM	Man-in-the-middle	中间人攻击
RTT	Round-trip time	往返时间
RSA	Rivest-Shamir-Adleman	RSA非对称加密算法
CBC	Cipher-block chaining	密码分组链接
PRF	Pseudo-random function	伪随机函数
MD5	Message-digest algorithm	MD5消息摘要算法
SHA	Secure hash algorithm	安全散列算法
AEAD	Authenticated encryption with additional data	带有关联数据的身份认证加密

表 2 本文中的英文缩写及说明 (续)

英文缩写	英文全称	中文名称
AES	Advanced encryption standard	高级加密标准
PSK	Pre-shared key	预共享密钥
MAC	Message authentication code	消息认证码
HKDF	Extended key derivation function	扩展密钥导出函数
DH	Diffie-Hellman	DH密钥交换协议
DHE	Ephemeral Diffie-Hellman	临时DH密钥交换协议
(EC)DHE	Elliptic curve Diffie-Hellman ephemeral	椭圆曲线DH临时密钥交换算法
(EC)DSA	Elliptic curve digital signature algorithm	椭圆曲线数字签名算法
PFS	Perfect forward secrecy	前向安全性
KCI	Key compromise impersonation	抗密钥泄露伪装攻击
AKE	Authenticated key exchange	认证密钥交换
PKCS	Public-key cryptography standards	公开密钥加密标准
DFA	Deterministic finite automaton	确定性有限状态自动机
FSM	Finite state machine	有限状态机
CVE	Common vulnerabilities and exposures	通用漏洞披露
DSA	Digital signature algorithm	数字签名算法
MTI	Minimum transport integration	最小传输集成
GCM	Galois/counter mode	GCM认证加密模式
CCM	Counter with CBC-MAC	CCM认证加密模式
PKEM	Puncturable key encapsulation mechanism	可穿刺密钥封装机制
PPRF	Puncturable pseudorandom functions	可穿刺伪随机函数
SNI	Server name indication	服务器名称标识

1 TLS 1.3 协议发展历程

TLS 协议由 1994 年网景公司提出的内部草案 SSL 发展而来. 随后, IETF 在 1996 年 5 月正式启动了 SSL 协议的标准化工作并重新命名为 TLS, 之后不断更新优化以增强安全性, 并逐步演化至当前最新的 TLS 1.3 版本. 其发展历程如图 2 所示.

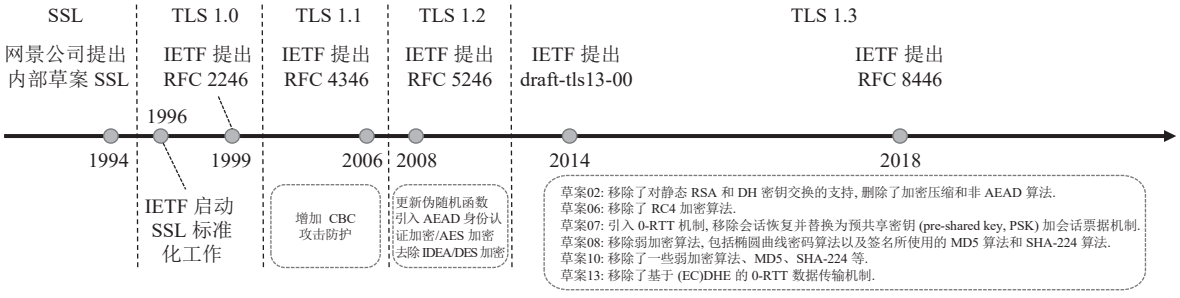


图 2 TLS 及 TLS 1.3 协议发展历程

1.1 TLS 发展历程

TLS 从 1.0 到 2.0 版本, 每一次版本迭代都会引入更安全的加密算法并废弃过时的安全机制.

1999 年 1 月, TLS 1.0 版本[19]正式发布. 此后, IETF 不断改进和完善 TLS 协议, 相继提出了 TLS 1.1 版本[20]和 1.2 版本[5]. 其中, TLS 1.1 添加了针对 CBC 块攻击的防护; TLS 1.2 更新了 PRF 伪随机函数, 修改了加密套件和握手协议中的算法, 将原来的 MD5 算法和 SHA-1 算法替换为 SHA-256 算法, 不再支持通过协商使用 SSLv2 不安全协议, 此外还引入了 AEAD 身份认证加密技术和 AES 加密等.

TLS 1.2 是目前大多数浏览器和 WEB 网站使用的 TLS 协议版本. 然而, 随着网络环境的日趋复杂和攻击手段不断提高, TLS 1.2 逐渐暴露出一些缺陷. 例如, 利用 RC4 加密算法漏洞进行攻击[21]; 以及利用协议设计漏洞进行填充攻击, 包括针对 CBC 加密模式的 Lucky13 攻击[22,23]、POODLE 攻击[24], 针对握手协议的 Logjam 攻击[25]、三次握手攻击[26]、Raccoon 攻击[27], 以及 DROWN[28]和 ALPACA[29]等跨协议攻击; 此外, 还有利用软件实现漏洞进行攻击, 如 Heartbleed 攻击[30]、状态机攻击[31]等. 因此, IETF 开始着手设计“下一代 TLS”, 即 TLS 1.3.

1.2 TLS 1.3 安全特性改进

TLS 1.3 标志着加密协议的一次重大升级, 其设计简化了握手过程, 并显著增强了安全性. 2014 年 4 月, TLS 1.3 第 1 版草案发布. 随后, IETF 连续发布了 28 个草案版本, 引起了学术界和工业界的广泛讨论和分析. 最终, IETF 在 2018 年 8 月正式推出了 TLS 1.3 标准 RFC 8446[6]. 相比于 TLS 1.2, TLS 1.3 在安全性和效率方面的优势逐步凸显.

表 3 汇总了所有 TLS 1.3 草案版本的安全特性和改进, 其中比较重要的版本及安全更新包括: 02 版草案移除了对静态 RSA 和 DH 密钥交换的支持, 删除了加密压缩和非 AEAD 算法; 06 版草案移除了 RC4 加密算法; 07 版草案引入了 0-RTT 握手机制, 移除会话恢复并替换为 PSK 预共享密钥和 ST 会话票据机制; 08 版草案移除了一些弱加密算法, 包括椭圆曲线密码算法以及签名所使用的 MD5 算法和 SHA-224 算法; 10 版草案将客户端和服务器密钥共享合并到了单个扩展中; 13 版本移除了基于 (EC)DHE 的 0-RTT 数据传输机制.

表 3　TLS 1.3 协议各草案版本安全特性汇总

草案版本	发布时间	草案中的主要安全特性改进
草案01	2014-04-17	基于TLS 1.1修改, 标志TLS 1.3协议设计工作初始化. 增加AEAD加密; 删除IDEA/DES加密套件
草案02	2014-07-07	基于TLS 1.1/1.2修改. 引入1-RTT握手; 删除对压缩/静态RSA/DH密钥交换/非AEAD算法的支持
草案03	2014-10-27	合并RFC 4492中对ECC算法的支持; 添加明确的HelloRetryRequest; 删除格林位威治时间
草案04	2015-01-03	修改密钥计算方法; 删除ChangeCipherSpec协议/重新协商等机制
草案05	2015-03-09	禁止SSL协商
草案06	2015-06-29	删除RC4加密算法和显示初始化向量IV
草案07	2015-07-08	引入0-RTT握手; 增加HKDF; 删除会话恢复模式并替换为PSK+ST会话票据模式
草案08	2015-08-28	删除一些弱安全性的椭圆曲线算法; 删除签名所用的MD5算法和SHA-224算法
草案09	2015-10-05	增加MTI加密套件; 将握手消息改为使用RSA-PSS签名, 删除DSA签名并弃用SHA-1签名
草案10	2015-10-19	将客户端和服务器密钥共享合并到单个扩展中
草案11	2015-12-28	统一认证模式, 增加握手后客户端身份认证; 增加版本降级攻击防御机制; 密钥更改时重置序列号
草案12	2016-03-22	提供PSK加密套件列表; 支持服务器发送0.5-RTT数据; 强制票据有效期为一周; 定义密钥导出器
草案13	2016-05-22	删除0-RTT客户端认证和基于(EC)DHE的0-RTT握手
草案14	2016-07-11	允许Fatal_alert警告后的会话恢复; 制定0-RTT限制规则
草案15	2016-08-17	禁止空的会话票据; 禁止在握手消息中同时发送应用数据
草案16	2016-09-22	修改版本协商机制; 修改完善RSA-PSS和EdDSA签名; 禁止使用0-RTT和PSK进行身份认证
草案17	2016-10-20	重构PSK密钥协商模式; 增加0-RTT密钥导出器; 阐明基于PSK的0-RTT模式的使用条件
草案18	2016-10-26	删除非必要的resumption_psk共享密钥; 修正拓展表中的签名算法; 禁止不同SNI的会话恢复
草案19	2017-03-10	重构证书扩展; 添加状态机, 描述客户端和服务器握手时的合法状态转换; 强制对等公钥验证
草案20	2017-04-28	重新启用握手后客户端身份验证; 添加关于加密流量分析和侧信道攻击的讨论
草案21	2017-07-03	更新状态机以显示重新生成密钥事件; 添加关于0-RTT机制和防重放攻击的讨论
草案22	2017-11-29	允许不同SNI的会话恢复; 允许空的ticket_nonce
草案23	2018-01-05	添加关于静态RSA的安全攻击说明, 以及如何在TLS所有版本禁用静态RSA
草案24	2018-02-15	强制要求ClientHello消息中的版本字段为0303
草案25	2018-03-02	将记录协议头部字段添加到AEAD附加数据中, 以确保其完整性
草案26	2018-03-04	禁止服务器与TLS 1.3之前版本协商时发送supported_versions扩展
草案27	2018-03-18	强制要求服务器必须处理ClientHello消息supported_versions扩展不包含0304版本的情况
草案28	2018-03-20	添加关于PSK身份暴露的安全攻击说明

1.3 小 结

TLS 协议从 SSL 发展而来, 历经多个版本发展, 每次版本迭代都进行了安全和性能优化. 其中, TLS 1.3 标志着传输层安全协议的重大进步, 通过移除过时的加密算法、优化握手过程和引入更高效的密钥交换机制, 来提升协议安全性和握手效率. 本文将在第 2 节中详细介绍 TLS 1.3 工作原理和 0-RTT 机制.

2 TLS 1.3 协议原理及 0-RTT 数据传输

TLS 作为构建网络的必要组成部分, 是保障整个网络系统安全传输的基础, 其主要目标是确保应用程序间数据通信的机密性和完整性. TLS 协议通常由规范层 (specification)、实现层 (implementation) 以及配置层 (configuration) 组成. 规范层由 IETF 等标准组织制定, 详细规定了协议的工作原理、数据格式、加密算法的选择及使用方式, 以及安全握手过程中的所有细节. 基于规范层的定义, 不同的开发者或厂商会创建具体的 TLS 实现, 如 OpenSSL. 这些实现提供了应用程序编程接口 (application programming interface, API) 供程序调用, 以建立安全连接. 实现层主要负责按照规范执行加密和解密操作, 此外还须处理诸如证书验证、错误处理和性能优化等问题. 配置层则是在实际应用部署中, 管理员根据需要对 TLS 服务端和客户端进行个性化设置, 包括选择支持的协议版本、启用或禁用特定的加密套件、设定证书链等功能.

2.1 TLS 1.3 基本结构及工作原理

TLS 1.3 在 TCP/IP 协议栈中介于应用层和传输层之间, 可以为上层的应用层协议 (如 HTTP、FTP 等) 提供安全连接, 由位于上层的握手协议 (见第 2.1.1 节)、警告协议 (见第 2.1.3 节) 和位于下层的记录协议 (见第 2.1.2 节) 组成, 如图 3 所示.

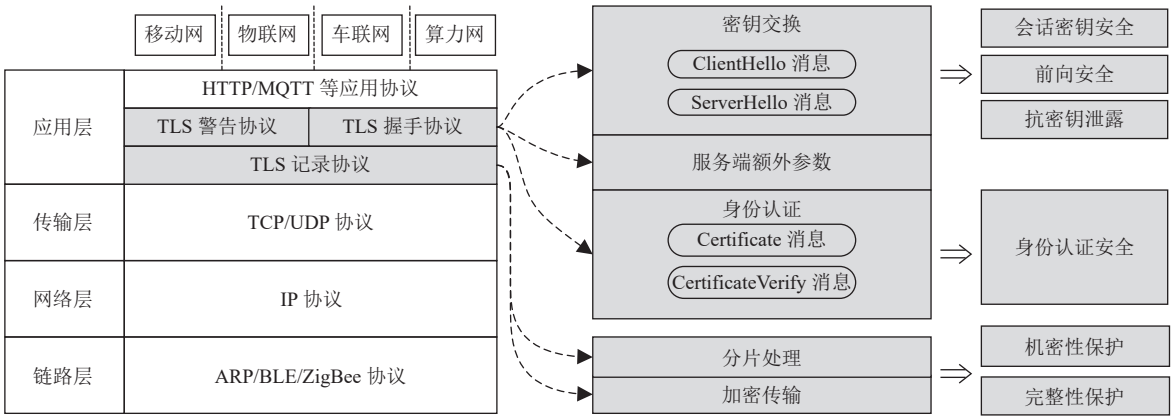


图 3 TLS 1.3 协议基本结构及安全性保证

TLS 1.3 遵循客户端-服务器模型, 当客户端 (通信发起方) 和服务端 (通信接收方) 第 1 次建立连接时, 通信双方通过握手协议对对端进行身份认证, 并与对端进行密钥交换, 通过协商协议版本、密码算法和算法所需参数来实现会话安全保证. 当握手协议完成后, 记录协议使用握手协议协商好的算法和参数对消息流进行分块加密和传输, 以提供消息机密性和完整性保护. 警告协议主要负责在通信发生异常时, 将错误告知对端.

2.1.1 握手协议

握手协议用于协商 TLS 安全连接的参数, 从而保证 TLS 会话的机密性与完整性. 这些参数包括 TLS 客户端和服务端选择的协议版本、加密算法和建立共享密钥、身份认证等. 一旦握手完成, 通信双方便可使用已经建立的密钥来加密保护应用层通信.

TLS 1.3 握手协议的详细工作流程如图 4 所示, 包括密钥交换、服务端额外参数和身份认证这 3 个阶段.

(1) 密钥交换

通信初始化时, 双方首先协商协议版本、密码算法和加密参数. 该阶段的通信为明文传输, 之后的所有通信内

容都会加密. 具体来说, 在密钥交换阶段, 客户端首先发送 ClientHello 消息, 其中包含一个随机数、协议版本、客户端支持的密码算法套件, 以及一组 DH 共享密钥或 PSK 共享密钥标签. 服务端接收到 ClientHello 消息后, 会依据该消息来选择适当的加密算法和参数, 并通过 ServerHello 消息响应给客户端. TLS 1.3 支持 3 种密钥交换机制, 包括 (EC)DHE、PSK 以及基于 (EC)DHE 的 PSK.

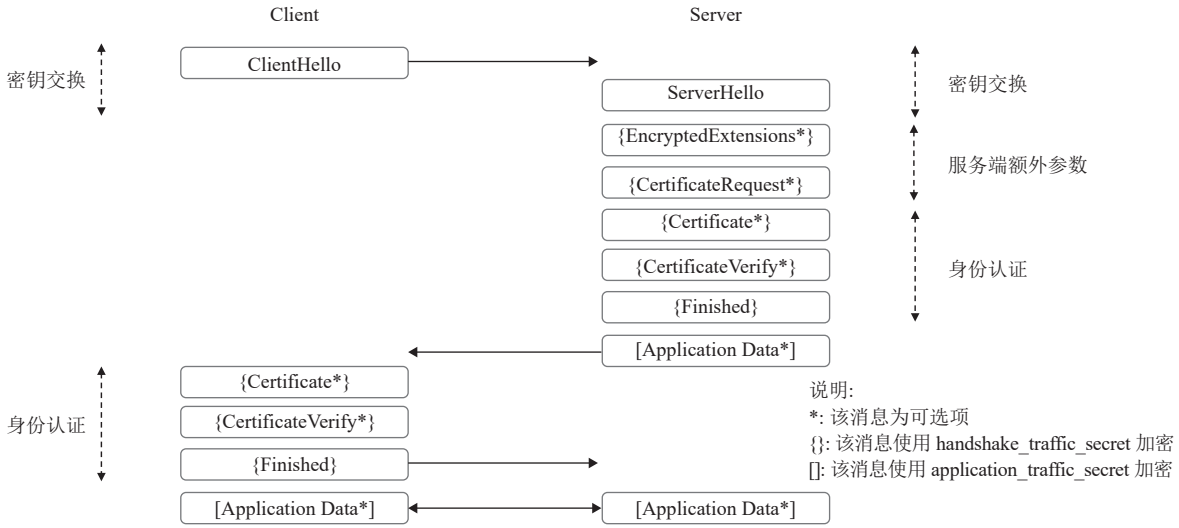


图 4　TLS 1.3 握手协议基本工作流程

TLS 1.3 支持 5 种密码套件, 表 4 对每种套件的加密算法和安全、性能进行了简单对比. TLS 1.3 默认使用 TLS_AES_128_GCM_SHA256 加密套件.

表 4　TLS 1.3 密码套件对比

密码套件	密钥交换	加密算法	认证加密	哈希算法	使用方式	性能	安全性
TLS_AES_128_GCM_SHA256		AES-128	GCM	SHA-256	强制使用	高	较高
TLS_AES_256_GCM_SHA384		AES-256	GCM	SHA-384	可选	较高	高
TLS_CHACHA20_POLY1305_SHA256	ECDHE	ChaCha20	Poly1305	SHA-256	可选	高	较高
TLS_AES_128_CCM_SHA256		AES-128	CCM	SHA-256	可选	中	较高
TLS_AES_128_CCM_8_SHA256		AES-128	CCM	SHA-256	可选	中	较高

(2) 服务端额外参数

随后, 服务端发送 EncryptedExtensions 和 CertificateRequest 消息以建立服务端额外参数. 这些消息是 TLS 连接过程中的第 1 条加密消息, 由握手层流密钥 (handshake_traffic_secret) 加密传输. EncryptedExtensions 消息用于响应 ClientHello 消息中的额外扩展, 这些扩展不用于协商加密参数, 但特定于单个证书的参数除外. CertificateRequest 消息仅在服务端需要验证客户端身份时使用.

(3) 身份认证

最后, 客户端和服务端交换身份验证消息来认证通信双方 (客户端认证为可选项), 并保证握手消息的完整性. 这些消息同样采用握手层流密钥加密传输. 具体来说, 双方各自发送 Certificate、CertificateVerify 和 Finished 消息, Certificate 消息包含了一个或多个 X.509 证书链. CertificateVerify 消息使用 Certificate 消息证书公钥对应的私钥来对整个握手进行签名. Finished 消息包含了之前所有握手消息的 MAC 认证码. MAC 可以防止通过篡改握手消息实现 TLS 降级攻击.

当 3 个阶段完成后, 通信双方便可使用应用层流密钥 (application_traffic_secret) 对应用数据进行加密传输.

2.1.2 记录协议

记录协议位于握手协议下层, 主要用握手协议产生的安全参数来封装上层协议的应用数据. 记录协议首先将应用数据进行分片处理, 转换为可管理的块, 然后利用 AEAD 身份认证加密算法进行加密传输. 接收方接收数据后对其进行解密和验证, 重组后再传送给上层协议. 具体工作过程如下.

(1) 分片处理

上层应用数据分片或合并成易于处理的数据分组 (块). 记录协议的数据类型有 3 种, 包括握手消息、应用数据和警告消息. 需要注意的是警告消息不能被分片处理.

(2) 加密传输

将明文数据通过 AEAD 算法加密为密文. TLS 密文数据结构包括 4 个部分: 数据类型、协议版本、消息长度和加密内容. 其中数据类型字段被设置为应用数据, 真实的数据类型可以从明文的数据类型获取, 该字段是为了兼容 TLS 1.3 之前版本而存在.

2.1.3 警告协议

警告协议与握手协议位于同一层, 其目的是以简单的通知机制告知通信双方出现的异常情况及严重性级别. 警告消息分为关闭警告和致命警告. 关闭警告用来指示连接的正常有序关闭或者 0-RTT 早期数据发送结束, 对通信过程没有影响. 致命警告则用来指示连接的非正常关闭, 收到这类警告消息后通信双方应立即中断会话, 不再收发消息.

2.2 TLS 1.3 握手 RTT 改进

TLS 1.3 在 TLS 1.2 的基础上做了许多重大改进, 比如在记录协议禁用 AEAD 之外的其他加密算法、在握手协议层去除静态 RSA 和 DH 密钥协商, 这些改进大大增强了 TLS 协议的整体安全性. 此外, TLS 1.3 还引入了 1-RTT 握手及 0-RTT 数据传输, 极大提高了握手效率.

RTT 是 round-trip time 的简写, 即消息往返时间. 对于 TLS 1.2 来说, 完整握手共需两个 RTT. 第 1 个 RTT 发送 ClientHello 和 ServerHello 消息, 用于协商密钥交换算法以及密码参数. 第 2 个 RTT 发送 ClientKeyExchange 和 ServerKeyExchange 消息, 也就是使用前一个 RTT 协商的算法和参数进行密钥交换. 握手完成后, 客户端和服务端便可以安全发送应用数据. 为了提高数据传输效率, TLS 1.2 引入了会话恢复机制, 只需 1 个 RTT 就能完成快速连接. 其基本原理是在客户端与服务端握手过程中创建一个会话 ID (session ID) 或会话票据 (session ticket), 并在 TLS 连接断开后将会话的安全参数保存一段时间. 如果客户端想快速恢复会话, 只需将对应的会话 ID 或票据通过 ClientHello 消息发送给服务端, 服务端如果同意恢复会话, 便将相同的会话 ID 通过 ServerHello 消息返回给客户端. 接着, 双方便可以使用之前协商过的共享密钥发送应用数据.

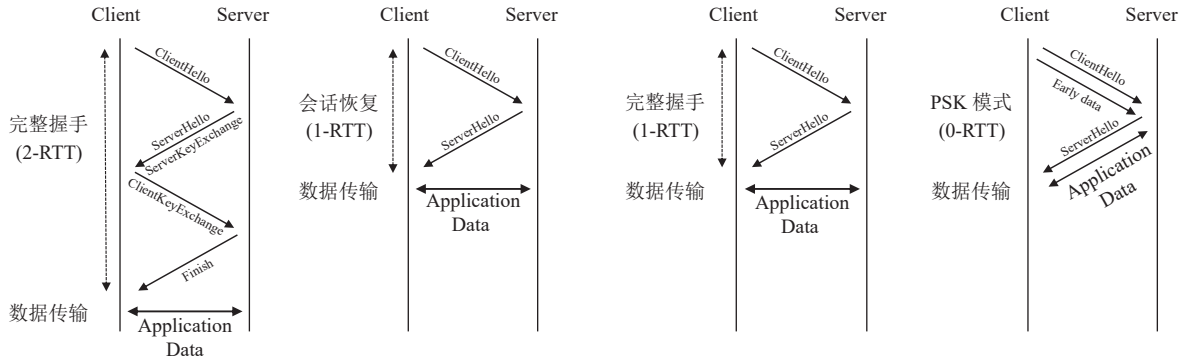
相比 TLS 1.2, TLS 1.3 进一步对握手机制进行了优化, 删除了 TLS 1.2 握手过程中的 ServerKeyExchange 和 ClientKeyExchange 步骤, 并且通过 KeyShare 扩展来优化密钥交换过程, 使得 TLS 1.3 只需 1 个 RTT 即可完成握手. 此外, TLS 1.3 借鉴 QUIC 协议设计了一个 0-RTT 数据传输方案. 0-RTT 是一种允许客户端在零往返时间内发送加密数据的协议机制, 即在握手协议的第一条消息中就允许客户端发送应用数据而不必等到握手协议结束.

TLS 1.3 中的 0-RTT 具体解决方法是通过 PSK 缓存最近服务端使用的加密算法和参数, 在第 1 条消息中直接利用缓存的密钥发送部分应用层数据 (又称为 Early data 或早期数据), 另一部分应用数据则等到完整的握手协议之后发送. 0-RTT 早期数据不能抵抗重放攻击, 攻击者 (如中间人) 可以拦截早期数据并将它们多次重新发送给服务器. 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防重放攻击保护措施.

后文图 5 是 TLS 1.2、TLS 1.3 完整握手和会话恢复的 RTT 对比.

2.3 0-RTT 数据传输机制

在 TLS 1.3 草案 13 之前, TLS 1.3 支持两种 0-RTT 数据传输机制, 包括基于 (EC)DHE 的 0-RTT 和基于 PSK 的 0-RTT. 但由于前向安全问题, 基于 (EC)DHE 的 0-RTT 机制已经被移除. 下面分别介绍这两种 0-RTT 机制.



(a) TLS 1.2 首次连接 (2-RTT) (b) TLS 1.2 会话恢复 (1-RTT) (c) TLS 1.3 首次连接 (1-RTT) (d) TLS 1.3 会话恢复 (0-RTT)

图 5 TLS 1.2 和 TLS 1.3 协议 RTT 对比

2.3.1 基于 (EC)DHE 的 0-RTT 机制

基于 (EC)DHE 的 0-RTT 机制工作流程如图 6 所示. 假设客户端已经与服务端建立过 (EC)DHE 密钥交换, 并且在该通信中获得了所谓的服务端配置. 在密码学上, 这个配置实际上是一个半静态的 DH 共享 g^s, 服务端在一定时间内存储该密钥.

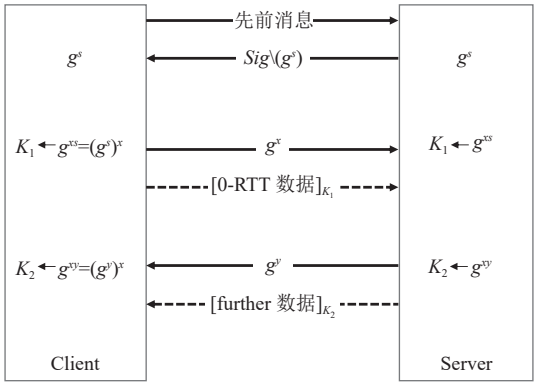


图 6 基于 (EC)DHE 的 0-RTT 机制

在随后执行的第 1 个消息中, 客户端发送一个短暂 (临时) 的 DH 共享 g^x, 将 0-RTT 密钥 K_1 导出为 $K_1=(g^s)^x=g^{xs}$. 因此可以用此密钥加密 0-RTT Early data 数据并发送. 然后, 服务端计算与 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 $K_2=g^{xy}$. 此后的数据便采用 K_2 加密.

2.3.2 基于 PSK 的 0-RTT 机制

基于 PSK 的 0-RTT 机制是目前 TLS 1.3 标准中默认指定的 0-RTT 机制. 该机制假设通信双方如果之前已建立过一次完整握手, 那么在会话恢复时无需重新握手, 客户端可选择直接恢复会话并发送早期数据给服务端.

如图 7 所示, 基于 PSK 的 0-RTT 主要包含两个阶段. 首先, 客户端发起会话恢复连接. 客户端在 ClientHello 消息中声明其希望使用之前存储的 PSK 预共享密钥进行连接, 并通过 pre_shared_key 扩展列出可用的 PSK 标识. 为了提供前向安全性, 客户端还会同时在 key_share 扩展中提供一个新的临时 DH 公钥. 在此期间, 客户端还可以发送早期数据, 其加密密钥 K_1 从先前建立的密钥导出. 在 TLS 1.3 中, “先前建立的密钥”是指通信双方在之前握手时协商出的用于未来会话的 PSK, 包括通过带外方式建立的密钥和在一般 (EC)DHE 握手中为会话恢复而建立的密钥, 分别称为 PSK 和 PSK-(EC)DHE. 服务端收到该消息后会检查 PSK 是否有效, 并在 ServerHello 消息中提

供自己的临时 DH 公钥. 随后, 客户端和服务端基于临时 DH 公钥和 PSK 协商出会话密钥用于后续通信. 此后的消息便使用会话密钥导出的密钥 K_2 进行加密传输. 客户端和服务端通过更新 K_2 来保证会话安全.

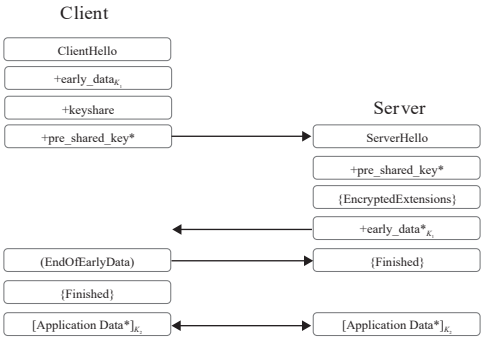


图 7 基于 PSK 的 0-RTT 机制

2.3.3 0-RTT 数据安全性

相比 TLS 完整握手后传输的 1-RTT 数据, 0-RTT 数据的安全属性较弱. 首先, 0-RTT 数据不能保证前向安全性, 因为它仅根据使用提供的 PSK 派生的密钥进行加密. 其次, 0-RTT 数据不能保证连接之间不会重放. 完整握手后再传输的 1-RTT 数据重放保护是通过服务器的 Random 值提供的, 但 0-RTT 数据不依赖于 ServerHello, 因此无法抵抗重放攻击.

2.4 小 结

TLS 1.3 在 TLS 1.2 基础上精简了握手过程并强化了安全特性, 提高了互联网通信安全性和握手效率. 其引入的 0-RTT 机制允许客户端在首次连接时发送加密的应用程序数据而无需等待服务器响应, 从而显著减少了连接延迟. 然而, 0-RTT 也带来了潜在的安全风险, 包括前向安全和重放攻击, 需要应用程序实施额外防护以确保数据传输安全. 0-RTT 安全是 TLS 1.3 安全领域中的热门研究方向, 学术界针对 0-RTT 安全展开了一些研究并提出了一些缓解方案, 本文将在第 3.2.1 节中对这些方案进行阐述.

3 TLS 1.3 协议机制安全

协议安全性分析一直是协议安全领域的核心研究问题, 依赖形式化方法对协议顶层设计安全性进行系统分析和证明, 通常基于严格的数学基础理论, 在计算机领域对目标系统进行规约、分析和验证. 形式化方法在分析高安全性需求的关键系统时, 具有不可比拟的重要性, 尤其在协议安全分析领域中更为突出. 普遍而言, 形式化方法多用于规范层安全分析, 而实现层和配置层分析则更多使用软件测试方法.

目前针对 TLS 1.3 的安全分析主要集中在协议草案发布期间. 从 2014 年 4 月发布第 1 版草案, 到 2018 年 8 月协议标准正式成立, 研究人员围绕协议草案中的握手安全性及会话安全性展开了深入分析和研究, 包括安全分析模型、安全属性验证和证明.

3.1 威胁模型与假设

在正式介绍 TLS 1.3 协议安全分析前, 首先对密钥交换安全分析的基础框架 CK (Canetti-Krawczyk) 模型[32]进行简单介绍. CK 模型对敌手的能力进行了形式化描述, 反映敌手在开放网络环境中拥有的基本攻击能力, 即拥有对通信链路的完全控制. 具体来说, 攻击者可以监听、拦截和修改消息, 随意延迟或阻止消息传递、插入自己的消息. 此外, 攻击者还具有以下额外能力.

(1) 会话状态揭露: 针对还未完成的会话, 攻击者能够通过此查询得到特定会话的状态, 如临时 DH 值.

(2) 会话密钥查询: 在单个会话完成后, 攻击者能够通过此查询得到会话密钥.

(3) 腐化参与者: 攻击者能够攻陷实体, 得到长期私钥以及和会话相关的具体信息.

该模型允许攻击者显示通信一方测试会话时的长期私钥或显示一方的临时私钥, 能帮助敌手进一步获得更强的攻击能力.

3.2 协议安全属性

加密协议的安全属性是形式化评估其安全性的核心指标, 定义了协议在面临特定威胁模型时所能提供的安全保障. TLS 加密协议的机密性、完整性、前向安全等安全性质, 可确保数据传输过程的安全与隐私, 保障了通信双方信息交换的安全可靠. TLS 1.3 握手协议主要提供了会话密钥安全、完全前向安全、抗密钥泄露伪装攻击、身份认证安全这 4 个安全属性, 记录协议主要提供了机密性和完整性保护.

(1) 会话密钥安全

作为 AKE 密钥交换协议的核心目标, TLS 1.3 确保了生成的会话密钥满足以下安全要求.

● 密钥一致性: 所有未被攻陷的协议参与方, 在完成匹配的会话后, 应协商出相同的会话密钥.

● 密钥伪随机性: 攻击者在任何概率多项式时间内无法区分从密钥交换协议输出的真实会话密钥和一个相同长度的随机字符串, 即攻击者无法从协议交互中获取关于会话密钥的任何有效信息.

(2) 完全前向安全 (PFS)[33]

PFS 是指即使协议参与方用来产生会话密钥的长期密钥 (私钥) 在未来被泄露, 也不会影响之前使用的会话密钥的安全性. 此属性通过在密钥协商过程中强制使用临时密钥实现, 确保会话密钥的生成不依赖于长期密钥.

(3) 抗密钥泄露伪装攻击 (KCI)

如果协议参与方的长期私钥泄露, 攻击者虽可冒充该方与其他方通信, 但不能冒充其他任意方与该密钥泄露方成功建立会话. 此属性保障了密钥泄露方在与其他诚实方通信时的身份真实性.

(4) 身份认证安全

协议应确保通信双方均能验证对方的身份, 即客户端确认的服务器身份是真实的, 且服务器确认的客户端身份也是真实的.

(5) 机密性保护

攻击者无法从加密记录中恢复出明文内容.

(6) 完整性保护

攻击者无法修改、伪造或重放记录数据而不被接收方检测.

3.3 协议安全性分析

协议安全分析是对通信协议设计机制进行系统性审查的过程, 从而验证协议安全属性是否能得到保证. 常用的协议安全分析模型, 主要有符号模型和计算模型两类. 符号模型是利用符号化的公式表示安全协议, 并通过逻辑推导的方式来分析协议安全性的方法, 其基础是 Dolev-Yao (DY) 模型[34]; 计算模型则是依赖于计算复杂度的分析模型, 通过将密码协议自身与协议所采用的具体密码算法结合在一起, 来计算多项式时间内任意攻击下的安全保证. 通常, 符号模型通过攻击者假设, 巧妙地避开了复杂的密码算法计算, 也导致很多协议被符号方法证明安全后又发现漏洞, 而计算模型的分析过程更符合真实世界, 因此被认为更可靠. 为了简化分析过程, 业界实现了自动分析工具, 具有代表性的符号模型分析工具有 AVISPA[35]、ProVerif[36]、Maude-NPA[37]、Scyther[38]、Tamarin[39], 计算模型分析工具有 CryptoVerif[40]. 此外, 研究者还通过拓展特定的安全模型, 如 BR (Bellare-Rogaway) 模型[41]、CK 模型[32]、eCK 增强模型[42]、MSKE (multi-stage key exchange) 模型[43]、ACCE (authenticated and confidential channel establishment) 模型[44], 以更精确的分析 TLS 等大型复杂加密协议, 从而提升安全分析准确性.

TLS 1.3 协议安全分析的基本流程如图 8 所示. 研究者选定目标协议草案或标准后, 首先分析协议功能模块及其安全属性, 如基于 (EC)DHE 的 1-RTT 完整握手和基于 PSK/PSK-DHE 的 0-RTT 握手功能所提供的会话密钥安全性、前向安全性、身份认证安全和抗密钥泄漏攻击, 以及 AEAD 算法提供的机密性和完整性保护; 然后基于符号模型或计算模型建立目标协议的分析模型, 同时对目标协议功能模块所声明的安全属性进行形式化建模; 最后通过证明安全或不存在敌手攻击的方式验证协议安全属性是否可满足.

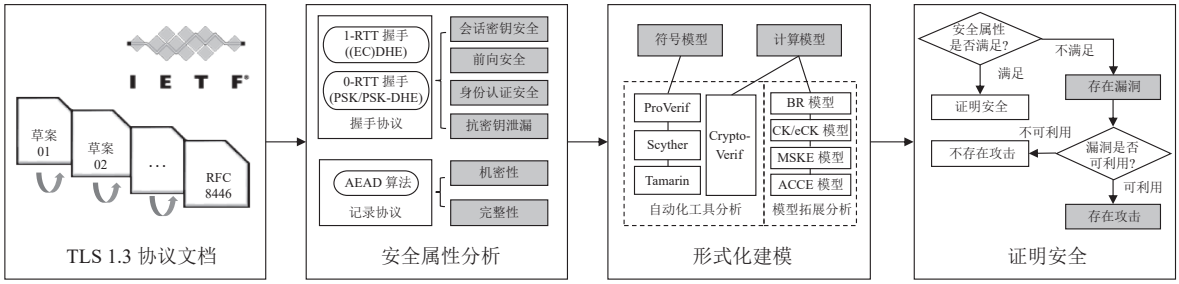


图 8 TLS 1.3 安全分析基本流程

3.3.1 握手协议分析

TLS 1.3 握手协议安全分析可以划分为两个阶段: 草案阶段和正式标准阶段. 在第 1 阶段, 研究者主要对 2015–2017 年发布的草案进行握手协议分析, 证明了 1-RTT 握手机制的安全性, 验证了静态 RSA 和 DH 密钥协商算法存在安全缺陷, 以及 0-RTT 机制存在前向安全和重放攻击风险. 这些分析成果有效地帮助后续草案和正式标准进一步提升了 TLS 1.3 整体安全性, 废弃了不安全算法并添加了 0-RTT 安全风险说明和推荐的防护措施. 在第 2 阶段, 研究者将 TLS 1.3 同其他安全协议进行了深入比较分析, 并对正式标准进行了更完备的分析和证明, 包括密钥派生算法和密钥协商算法, 从而弥补了第 1 阶段研究的不足.

(1) 草案分析阶段

2015 年, Dowling 等人[45]对草案 05 和草案 DH-based 进行了详细的密码学分析, 基于 BR 模型提出了一种增强的 MSKE 安全分析模型, 并基于此模型证明了草案中基于临时 DH 密钥交换握手协议的会话密钥安全性和身份认证安全性. 作者认为草案 DH-based 的安全分析可覆盖草案 07 的设计. Dowling 等人验证了多阶段密钥交换安全概念可以与任意对称密钥协议组合, 因此在记录协议中所使用的会话密钥也是安全的. 此外, Dowling 等人将 TLS 1.3 会话恢复阶段看成对称密钥协议, 证明了该组合分析方法也可以用于分析会话恢复的安全性. 但是, Dowling 等人并未分析草案 07 引入的 0-RTT 握手机制的安全性, 因为 0-RTT 并不能转换为对称密钥协议.

与此同时, Jager 等人[46]对草案 07 进行了分析, 指出尽管握手协议已不支持 PKCS#1 v1.5 填充方式, 但是仍然可以在某些场景下被攻击者利用 PKCS#1 v1.5 弱点来攻击 TLS 1.3 和 QUIC 协议, 比如在非前向安全性的 RSA 公钥加密模式中进行服务器身份验证.

2016 年, Li 等人[47]对草案 10 中的握手协议进行了密码学分析, 同样基于 BR 模型提出了一个可分析 0-RTT 机制的多级 MSKE 安全分析模型. Li 等人证明了该模型可以涵盖不同版本 TLS 握手机制之间的各种组合交互, 验证了 1-RTT 握手机制能提供会话密钥、身份认证、前向安全性, 但 0-RTT 并不能提供前向安全保证. 草案 10 中的握手协议符合多次握手安全的概念, 且多阶段安全模型普遍适用于具有不同会话机制的通信协议.

Fischlin 等人[48]和 Dowling 等人[49]同样对草案 10 进行了安全分析. Fischlin 等人基于 BR 模型证明了草案中的完整握手提供了会话保密和身份验证安全. Dowling 等人延续并扩展了之前关于草案 05 和 DH-based 的分析[45], 证明了草案 10 中基于 (EC)DHE 的完整握手提供了会话密钥和前向安全, 但基于 PSK 和 PSK-DHE 的 0-RTT 握手只实现了密钥安全, 无法保证前向安全性.

Krawczyk 等人[50]基于 CK 模型提出了一种身份认证编译器, 能将单向认证 (unilaterally authenticated, UA) 密钥交换协议升级到互相认证 (mutually-authenticated, MA) 密钥交换协议, 同时证明了一大类 UA 协议可以由此工具编译从而完成升级转换. 该工具主要针对草案 13 中的客户端身份认证机制. Krawczyk 等人提出的方法支持模块化方式对认证机制进行分析, 在“功能安全”概念下引入了一个广义的密钥交换模型.

2017 年, Lan 等人[51]对草案 18 进行了分析, 基于 Dowling 等人[45]提出的多阶段安全分析模型证明了草案中的 1-RTT 握手即使在 TLS 协议不同版本交互时, 也能提供会话安全和身份认证安全. Bhargavan 等人[52]则基于 ProVerif 符号模型和 CryptoVerif 计算模型对草案 18 中的 1-RTT 完整握手、0-RTT 握手及记录协议的安全性进行了证明.

(2) 正式标准分析阶段

2019 年, Chen 等人[53]基于 QACCE 模型[54]对 TLS 1.3 和 QUIC 协议进行了深入对比分析, 证明它们在通道安全、身份认证以及 IP 欺骗防护方面提供了很好的安全保证, 但 0-RTT 机制都存在前向安全和重放攻击问题.

2021 年, Dowling 等人[55]进一步扩展了之前的 MSKE 模型[45]和安全分析[7,49], 重新分析了 TLS 1.3 正式标准中的 1-RTT 完整握手及 0-RTT 机制的安全性. Diemert 等人[56]基于 MSKE 模型对 TLS 1.3 正式标准进行了分析, 并给出了严格的安全证明. 但是, Diemert 等人只分析了 TLS 1.3 标准中的握手协议核心部分, 即基于 (EC)DHE 的 1-RTT 完整握手, 并未对 0-RTT 握手机制进行分析.

2022 年, Bhargavan 等人[57]基于 ProVerif 对 TLS 1.3 隐私性进行了分析和证明. Brzuska 等人[58]基于状态分离[59]对 TLS 1.3 进行了全面的安全分析和完整证明, 包括 1-RTT 握手和 0-RTT 握手.

2023 年 Fischlin[60]提出了一种更隐蔽的 TLS 1.3 协议变体 sTeaLS, 并进行了完整的安全分析和证明. 通常情况下, 在 TLS 握手过程中, 客户端和服务器通过公开的方式协商密钥. sTeaLS 在 TLS 握手过程中引入了一种新的机制, 使得密钥交换过程更加隐蔽, 不易被第三方观察者发现或破解. 这种技术可以增加 TLS 通信的安全性, 能够有效防止密钥交换信息被恶意监听.

3.3.2 0-RTT 安全分析

0-RTT 机制最早在 2015 年 7 月发布的草案 07 中被引入, 其安全分析是握手协议安全分析中的关键问题. 因此, 有不少研究者单独针对 0-RTT 机制进行安全分析, 并针对前向安全问题和重放攻击提出了理论优化方案.

(1) 0-RTT 机制安全分析

2016 年, Krawczyk 等人[61]针对 0-RTT 机制进行了改进, 提出了一种新的密钥交换协议 OPTLS, 给出了协议设计原理并基于 CK 模型进行了密码学分析. OPTLS 协议放弃了以 RSA 为中心的传统设计思路, 而是以 DH 和椭圆曲线为主要加密基础, 不仅保证了 TLS 1.3 的 0-RTT 要求, 同时使得前向安全性成为强制性要求. TLS 1.3 草案 09 随后采用了 OPTLS 框架的 4 种工作模式和密钥导出方法, 包括 1-RTT (半静态)、1-RTT、基于 PSK 的 0-RTT 和基于 (EC)DHE 的 0-RTT. 由于存在安全风险, 基于 (EC)DHE 的 0-RTT 在草案 13 中被移除. Cremers 等人[62]基于 Tamarin 符号模型对草案 10 中的 0-RTT 握手进行了建模和分析, 证明了单向和双向身份认证情况下都满足密钥交换的安全要求.

2017 年, Cremers 等人[63]进一步对草案 21 进行了安全分析, 同样基于 Tamarin 工具证明了草案中的 1-RTT 握手及 0-RTT 握手都符合会话安全和前向安全要求. Fischlin 等人[7]指出 TLS 1.3 中的 0-RTT 机制存在重放攻击风险. Fischlin 等人分析了草案 12 中基于 (EC)DHE 的 0-RTT 机制和草案 14 中基于 PSK 的 0-RTT 机制的会话密钥保密性, 扩展了之前的多阶段安全分析模型[45,49], 同时阐明重放攻击下 0-RTT 安全性的限制.

2021 年, Drucker 等人[64]发现 TLS 1.3 基于 PSK 的 0-RTT 机制存在 Selfie 反射攻击. Selfie 攻击利用了 TLS 1.3 不强制要求对服务器和客户端进行显式身份验证这一事实, 并利用它来破坏协议的相互身份验证属性. Selfie 攻击是 TLS 1.3 标准正式发布后被发现的第 1 个攻击. Lu 等人[65]基于 Scyther 工具对 TLS 1.3 中的 0-RTT 握手机制进行了形式化分析, 发现存在 KCI 攻击和重放攻击, 并提出了一种优化机制提高了 0-RTT 数据安全性.

2022 年, Davis 等人[66]基于 MSKE 模型严格证明了 TLS 1.3 中的 PSK 握手安全性.

(2) 0-RTT 机制安全优化

针对 0-RTT 前向安全和重放攻击问题, 近年来学术界研究了一些缓解方案[8,67−70] (见表 5), 通过可穿刺密钥封装来实现前向安全保证, 并基于布隆过滤器进一步优化密钥穿刺效率, 但在实际应用方面仍然面临性能挑战.

2017 年, Günther 等人[8]研究了草案 18 的 0-RTT 机制, 提出了一种新的密钥交换协议, 即利用时间同步和 PKEM 可穿刺密钥封装机制来构建一个 0-RTT 密钥协商协议, 但是只讨论了协议架构而并未涉及具体算法. Günther 等人指出解决 0-RTT 密钥协商协议前向安全和重放攻击问题的关键并不是修改协议, 而是修改服务端处理和存储密钥的方式. 因此 Günther 等人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, 服务端可以通过密钥穿刺来“清空”密

钥, 将密钥长度减少到时间间隔的对数因子, 从而理论上解决了密钥长度线性增长的问题. 由于 Günther 等人提出的 PKEM 方案成本太高而无法在实践中使用, 张兴隆等人[67]将细粒度时间段内前向安全与具体网络协议结合起来, 对 TLS 1.3 中 Early data 的加密过程和传输过程进行了改进.

表 5　0-RTT 缓解方案对比

文献	核心机制	前向安全性	抗重放攻击能力	部署前提	性能开销
[8]	可穿刺密钥封装 (PKEM)	高	强	需修改TLS 1.3协议以支持穿刺加密, 并维护时钟同步和穿刺状态	穿刺操作需要更新密钥状态, 计算和存储开销高
[67]	细粒度时间优化的PKEM	高	强	需修改TLS 1.3协议以支持穿刺加密, 并维护时钟同步和穿刺状态	存储开销较大, 计算开销因具体加密算法实现而异
[68]	布隆过滤器加密 (BFKEM)	高	中	需修改TLS 1.3协议, 并维护布隆过滤器状态	计算和存储开销比PKEM低
[69]	可穿刺伪随机函数 (PPRF)	高	强	无需修改TLS 1.3协议, 仅需升级票证签发与验证逻辑, 并维护穿刺状态	基于RSA的PPRF方案性能开销较大, 基于对称密码的PPRF方案开销较小
[70]	基于时间的布隆过滤器 (TB-BFKEM)	高	中	需修改TLS 1.3协议, 并维护时钟同步和布隆过滤器状态	计算和存储开销较小

2018 年, Derler 等人[68]提出了基于布隆过滤器 (Bloom filter) 的 BFKEM 优化方案, 增加了密钥穿刺效率. 2019 年, Aviram 等人[69]进一步提出了基于可穿刺伪随机函数 PPRF 的优化方案.

上述研究方案虽然理论上可行, 但是都并未进行实际的性能测试. 2020 年, Dallmeier 等人[71]首次将基于布隆过滤器的 BFKEM 优化方案应用在 QUIC 协议中并进行了测试. 测试结果显示虽然该方案会带来大量的计算开销, 但在对 CPU 计算和内存开销不敏感的场景下, 该方案依然具有应用价值.

2023 年, Göth 等人[70]对上述 0-RTT 优化方案进行了系统分析, 提出了一种性能和安全适中的方案, 即基于时间的 BFKEM. 与上述方案对比, 该方案不仅获得了不错的计算效率, 同时也拥有更小的密钥空间.

3.3.3　记录协议分析

尽管研究重点集中在握手协议和 0-RTT 上, 但业界对 TLS 记录协议的安全分析也给予了持续关注. 在 2015–2018 年间, 部分研究者深入探讨了 TLS 1.3 之前版本中记录协议的攻击方法及其相应的防御策略, 构建了 TLS 1.3 记录层的安全分析模型并验证了 AES-GCM 和 ChaCha20-Poly1305 算法的安全性.

2015 年, Levillain 等人[72]对 TLS 1.3 之前版本的记录协议进行了系统研究, 对现有的各种攻击方法和防护措施进行了深入安全分析, 提出了针对这些攻击方法的通用防护对策. Levillain 等人首先列举了 TLS 1.0–TLS 1.2 记录协议的攻击方法, 包括针对 CBC 加密模式的 BEAST 攻击[73]、Lucky13 攻击[22]、POODLE 攻击[24]和针对压缩加密的 CRIME 攻击[74]、BREACH 攻击[75]、TIME 攻击[76], 以及 RC4 biases 攻击[21], 并对攻击方法和防御措施进行了详细的对比分析; 然后指出这些防御措施在实际应用中的效果并不理想, 原因在于它们仅针对特定类型的攻击, 且有时这些措施之间存在相互矛盾的情况; 最后提出了一种基于掩码屏蔽的通用防护模型.

2015 年, Badertscher 等人[77]首次对 TLS 1.3 草案 08 的记录协议进行安全分析, 提出了增强安全通道的分析模型, 并证明了记录协议中的认证加密方案满足该模型.

2016 年, Bellare 等人[78]对 TLS 1.3 记录协议进行分析, 证明了 AES-GCM 算法的多用户安全性, 并提出了一种更安全的认证加密方案.

2017 年, Delignat-Lavaud 等人[79]首次对 TLS 1.3 记录协议进行密码学分析, 构建了记录层的安全模型并进行证明. Delignat-Lavaud 等人计算了 AES-GCM 和 ChaCha20-Poly1305 算法的具体安全范围及边界, 实现了第一个可验证的 TLS 1.3 记录协议, 通过将记录层的实现代码插入到现有的 TLS 库中, 验证了与草案 14 和草案 18 的 TLS 库可以进行正常通信.

2018 年, Patton 等人[80]进一步针对草案 23 的记录协议进行分析, 指出草案并未实现密文流完整意义上的安全性, 并提出了改进意见.

3.4 后量子安全

传统加密算法如 RSA 和椭圆曲线通常依赖大数分解或离散对数问题, 这些问题对于经典计算机难以解决, 但对于量子计算机则容易破解. 随着后量子密码标准发布, 量子加密和 TLS 后量子安全也引起了业界普遍关注. 研究者在后量子握手协议设计、后量子迁移和性能优化等方面取得了一些进展, 推动 TLS 1.3 后量子研究从理论设计走向实践部署.

2016 年, Google 在 CECPQ1 (combined elliptic-curve and post-quantum 1) 实验中将后量子密钥交换算法 newhope1024 集成到 TLS 1.2 协议中. 2016 年, Stebila 等人[81]提出开放量子安全 (open quantum safe, OQS) 项目, 提供了一组后量子密码学算法的开源实现.

2019 年, Google 和 Cloudflare 在 CECPQ2 中将更高效的密钥交换算法 ntruhrss701 集成到 TLS 1.3 协议中. 随后, 学术界也对 TLS 协议后量子安全展开了分析.

2020 年, Schwabe 等人[82]提出 TLS 1.3 后量子握手协议 KEMTLS, 通过使用密钥封装机制来进行服务器身份验证, 大大提高了握手效率. 同年, Sikeridis 等人[83]分析对比了 NIST 标准中的后量子加密算法, 并集成到 TLS 1.3 库中进行大规模测试和性能评估. Sikeridis 等人验证了 Dilithium 和 Falcon 后量子签名算法都可在 TLS 协议中实际应用, 并且 Falcon 算法更适合 WEB 服务器场景.

2022 年, Bernstein 等人[84]将 sntrup761 后量子密钥交换算法集成到 OpenSSL 库中并提出 OpenSSLNTRU, 验证了该算法可以提升 TLS 1.3 密钥交换效率.

2023 年, Garcia 等人[85]首次提出并实现了将经典密码、后量子密码和量子密钥分发三者融合的 TLS 1.3 协议, 通过实验验证了 TLS 握手时间增加 68%, 为高安全需求场景提供了一个强有力的候选方案.

2024 年, Campos 等人[86]对高安全参数下的后量子密钥交换协议 CSIDH 进行了首次全面深入的实用性评估, 并将其集成到 TLS 1.3 协议中, 验证了其握手延迟过高的致命缺陷.

2025 年, Chen 等人[87]提出了一种通过密文扩展实现紧致安全规约的新型后量子 TLS 1.3 方案, 证明了在引入轻微开销的情况下, 基于 IND-CPA 安全 KEM 的握手协议可获得近乎最优的理论安全性.

3.5 对比分析

为了对上述研究进行系统对比, 本文将这些文献所研究的 TLS 协议层次、发表的会议期刊和时间、分析的协议版本和内容进行了归纳和总结, 见表 6.

表 6 TLS 1.3 协议机制安全分析相关研究对比

协议层次	分析类型	文献	发表时间	发表会议/期刊	分析协议版本	主要研究内容
握手协议	握手安全分析	[45]	2015	CCS	草案05	基于MSKE模型证明1-RTT握手安全性
		[46]	2015	CCS	草案07	指出握手协议PKCS#1 v1.5填充方式存在安全缺陷
		[47]	2016	S&P	草案10	基于MSKE模型证明1-RTT握手安全性
		[48]	2016	S&P	草案10	基于BR模型证明1-RTT握手安全性
		[49]	2016	IACR eprint	草案10	基于文献[45]证明1-RTT握手导出密钥的前向安全性
		[50]	2016	CCS	草案13	基于CK模型提出密钥交换协议身份认证编译器
		[51]	2017	TDSC	草案18	基于文献[45]证明1-RTT握手协议安全性
		[52]	2017	S&P	草案18	使用ProVerif/CryptoVerif证明握手/记录协议安全性
		[53]	2019	ESORICS	RFC 8446	基于QAACE模型对TLS 1.3/QUIC进行安全分析对比
		[55]	2021	JOC	RFC 8446	基于文献[7,45,49]重新分析1-RTT/0-RTT握手安全性
		[56]	2021	JOC	RFC 8446	基于MSKE模型严格证明了1-RTT握手安全性
		[57]	2022	CCS	RFC 8446	使用ProVerif证明1-RTT/0-RTT握手协议隐私安全
		[58]	2022	ASIACRYPT	RFC 8446	基于状态分离完整证明1-RTT/0-RTT握手安全性
		[60]	2023	CCS	RFC 8446	提出更隐蔽的TLS 1.3协议变体sTeaLS并进行了证明

表 6 TLS 1.3 协议机制安全分析相关研究对比 (续)

协议层次	分析类型	文献	发表时间	发表会议/期刊	分析协议版本	主要研究内容
握手协议	0-RTT安全分析	[61]	2016	EuroS&P	草案07	针对0-RTT机制进行改进并提出了OPTLS
		[62]	2016	S&P	草案10	使用Tamarin证明0-RTT握手的身份认证安全性
		[7]	2017	EuroS&P	草案12/14	基于文献[45,49]证明0-RTT存在重放攻击风险
		[63]	2017	CCS	草案21	使用Tamarin证明1-RTT/0-RTT握手安全性
		[64]	2021	JOC	RFC 8446	发现0-RTT握手机制存在Selfie反射攻击
		[65]	2021	软件学报	RFC 8446	使用Scyther证明0-RTT存在KCI攻击和重放攻击
		[66]	2022	Eurocrypt	RFC 8446	基于MSKE模型证明0-RTT握手机制安全性
	0-RTT安全优化	[8]	2017	Eurocrypt	草案18	提出基于可穿刺密钥封装的0-RTT优化方案
		[67]	2017	网络与信息安全学报	草案18	提出细粒度时间0-RTT优化方案保证早期数据安全性
		[68]	2018	Eurocrypt	NA	提出基于布隆过滤器的0-RTT优化方案
		[69]	2019	Eurocrypt	NA	提出基于可穿刺伪随机函数的0-RTT优化方案
		[71]	2020	CANS	NA	对基于布隆过滤器的0-RTT优化方案进行实际测试
		[70]	2023	CCSW	NA	提出安全和性能权衡的0-RTT优化方案
	后量子安全	[82]	2020	CCS	NA	提出TLS 1.3后量子握手协议KEMTLS
		[83]	2020	NDSS	NA	对NIST后量子加密算法进行TLS 1.3集成测试
		[84]	2022	UsenixSEC	NA	提出TLS 1.3后量子密钥交换算法OpenSSLNTRU
		[85]	2023	CAMAD	NA	提出TLS 1.3后量子密码和量子密钥分发融合
		[86]	2024	IACR	NA	对后量子密钥交换协议CSIDH进行TLS 1.3集成测试
		[87]	2025	IACR	NA	提出基于IND-CPA安全KEM的TLS 1.3握手协议
记录协议	安全分析	[77]	2015	ProvSec	草案08	对记录协议进行了安全分析
		[78]	2016	CRYPTO	NA	证明了AES-GCM算法的多用户安全性
		[79]	2017	S&P	草案14/18	证明并实现了可验证安全的TLS 1.3记录协议
		[80]	2018	CCS	草案23	对记录协议进行了密码学分析和改进

注: NA表示该文献中并未具体说明分析的TLS协议版本

从表 6 中可以看出, 在 TLS 协议草案发布阶段, 研究者主要围绕草案中的握手协议某一部分功能特性展开密码学分析和严格数学证明, 特别是握手协议中的 1-RTT、0-RTT 以及 PSK 机制的安全性是否满足密码学安全要求; 在 TLS 协议 RFC 标准发布后, 研究者主要针对前期已有的分析和未覆盖的协议特性进行更完备的分析和证明, 如隐私安全性和多用户安全性分析. 整体来看, 早期的安全分析工作为推进 TLS 1.3 安全奠定了重要基础, 目前 TLS 1.3 核心机制的安全性已经非常健全, 但随着协议的成熟和部署的普及, 当前的研究重点已转向 0-RTT 安全和后量子安全等更具挑战性的前沿问题. 虽然研究者针对 0-RTT 和后量子提出了一些优化方案, 如基于可穿刺密钥封装、布隆过滤器、可穿刺伪随机函数等机制来提供一定的前向安全和抗重放攻击, 以提升 0-RTT 数据传输安全性, 但由于性能开销过大离实际应用仍然还有很大差距.

3.6 小 结

TLS 1.3 安全分析主要集中在对 1-RTT/0-RTT 握手机制的深入探讨. 目前, 研究者已经基于多阶段密钥交换安全模型严格证明了基于 (EC)DHE 密钥交换的 1-RTT 握手安全性, 并验证了早期版本中静态 RSA 和 DH 密钥协商算法以及 RSA 算法 PKCS#1 v1.5 填充方式存在的安全缺陷. 同时, 研究者也证实了 0-RTT 握手存在以下安全风险, 包括前向安全、重放攻击、KCI 攻击和 Selfie 反射攻击. 针对前两个风险, 研究者虽提出了一系列理论优化方案, 如通过可穿刺伪随机函数、布隆过滤器、细粒度时间控制等机制来保证一定前向安全性和抵抗重放攻击, 但仍然存在很大性能挑战. 针对记录协议, 研究者证明了 AES-GCM 算法的多用户安全性, 实现了可验证安全. 此外, 研究者提出了一些后量子握手协议, 显著提升了 TLS 协议的后量子安全性和握手效率. 这些工作共同推动了 TLS 1.3 的安全发展, 为互联网通信的安全提供了坚实的理论和技术基础. 除了上述在 TLS 1.3 协议层内的安全分

析与攻击研究外, 近年来学术界开始将视角扩展到 TLS 与底层网络协议的跨层交互上. TLS 1.3 的安全模型通常建立在 TCP 提供的有序和可靠的数据流服务之上, 但 TCP/IP 协议栈与 TLS 加密协议在抽象层级和行为假设上的不一致, 可能引入新的攻击面. 这类跨层交互安全问题, 已成为 TLS 1.3 整体安全态势中一个不容忽视的组成部分, 本文将在第 6 节的未来展望中进一步阐述.

4 TLS 1.3 协议软件安全

在 2018 年之前, TLS 1.3 安全研究大多集中在对协议顶层设计的安全性分析和数学证明, 并非具体的软件实现. 然而, 尽管 TLS 1.3 协议规范从理论机制上提供了足够的安全保障, 但如果在软件具体实现过程中出现漏洞, 也可能导致 TLS 协议的安全保证失效. 因此, 在 TLS 1.3 正式标准发布后, 国内外研究人员围绕不规范和不正确的软件实现展开深入研究, 通过借助程序分析方法或软件测试技术来检测协议实现中的安全漏洞和缺陷, 以降低 TLS 协议被攻击的风险.

4.1 TLS 软件实现

TLS 软件实现是确保互联网和其他网络通信中数据交换安全的关键组件, 它基于 TLS 协议标准实现, 提供了一套机制和 API 接口来保护客户端和服务器之间的信息传输免受窃听、篡改和伪造. 随着网络安全的不断演变, TLS 协议标准和实现代码也在持续进化.

4.1.1 TLS 软件代码库

目前, 业界存在许多开源和商业的 TLS 协议软件, 它们广泛应用于各种软件和服务中, 以支持安全的网络通信. 以下是最流行的 5 款 TLS 协议软件.

(1) OpenSSL

OpenSSL 是业界最知名的开源 TLS 实现之一, 支持广泛的密码套件和协议版本, 包括 TLS 1.3. OpenSSL 因其灵活性和强大的功能特性而受到开发者的欢迎, 但也因其复杂的 API 和历史安全漏洞 (如 Heartbleed[30]) 而备受争议. OpenSSL 对 TLS 1.3 的支持非常全面, 从 1.1.1 版本开始正式支持 TLS 1.3. 它不仅实现了 TLS 1.3 的所有强制性特性, 还提供了对多种可选特性的支持, 如 0-RTT 握手机制, 这可以显著减少建立 TLS 连接所需的时间.

(2) BoringSSL

BoringSSL 是 Google 公司基于 OpenSSL 库开发的一个新分支, 旨在简化和优化 TLS 协议实现. BoringSSL 对 API 进行了深度清理, 移除了一些过时的功能, 并加强了对 TLS 1.3 标准的支持. BoringSSL 对 TLS 1.3 的支持同样很全面. BoringSSL 在设计上更加注重性能和安全性, 如通过移除不必要的功能来减少攻击面.

(3) NSS (network security services)

NSS 是 Mozilla 公司开发的一套开源安全库, 支持 TLS 1.3 等多种安全协议. NSS 可跨平台使用, 提供了丰富的 API 来处理证书管理、加密操作等任务. NSS 对 TLS 1.3 的支持较为完善, 它不仅实现了 TLS 1.3 的基本要求, 还在不断更新以适应新的安全需求和技术发展. Mozilla 的 Firefox 浏览器就是使用 NSS 库来支持 TLS 1.3.

(4) GnuTLS

GnuTLS 是另一个自由软件库, 专注于提供安全的网络通信功能. GnuTLS 同样支持 TLS 1.3, 并且具有良好的文档和支持社区. GnuTLS 在 TLS 1.3 的支持方面也做得不错, 它提供了对最新协议版本的全面支持, 包括对后量子密码学的研究和实验性支持.

(5) mbedTLS

mbedTLS 是一款由 ARM 公司开发和维护的轻量级开源 TLS 库, 特别适合资源受限的环境, 如嵌入式系统. mbedTLS 也实现了 TLS 1.3 标准, 同时保持了小体积和高性能的特点. 虽然 mbedTLS 是一个轻量级库, 但它对 TLS 1.3 的支持也非常完善. 它实现了 TLS 1.3 的关键特性, 确保即使是在资源有限的设备上也能实现安全通信.

4.1.2 TLS 代码库及 TLS 1.3 版本支持对比

尽管上述的这些 TLS 软件代码库各有特色, 但它们都致力于提供对 TLS 1.3 版本的良好支持, 以满足不同应

用场景下的安全需求. 开发者在选择合适的 TLS 库时需要根据项目的具体要求来决定, 比如性能需求、系统兼容性、安全性考虑以及开发者的熟悉程度等因素.

表 7 对业界流行的 TLS 代码库相关信息及支持的 TLS 1.3 特性进行了简要对比. 整体来看, OpenSSL 和 BoringSSL 是当前最流行的 TLS 库, 不仅拥有稳定的社区和开发维护, 还有非常友好的 Apache 开源许可, 且提供了对操作系统和加密协议 (TLS、DTLS 和 QUIC 协议) 的广泛支持. 从实现语言方面看, C 语言是当前最主要的 TLS 库实现语言, 而且复杂的密码学操作原语用 C 语言实现可以获得更好的性能. 在 TLS 1.3 特性支持方面, 大多数 TLS 库都支持 0-RTT 和 PSK 机制, 这给研究人员分析和测试 0-RTT 安全性带来了便利. mbedTLS、CycloneSSL、wolfSSL 库为嵌入式环境下的开发和测试提供了很好的支持, 如果研究者针对嵌入式设备及工业互联网设备进行研究, 可以选择这些库作为研究目标.

表 7 TLS 代码库对比

代码库	实现语言	开发厂商	开源许可	操作系统支持	协议支持	TLS 1.3特性支持
OpenSSL	C	OpenSSL	Apache	广泛支持	TLS/DTLS/QUIC	0-RTT/PSK
BoringSSL	C	Google	Apache	广泛支持	TLS/DTLS/QUIC	0-RTT/PSK
NSS	C	Mozilla	MPL	Windows/Linux/Mac	TLS/DTLS	0-RTT/PSK
GnuTLS	C	GNU	LGPL	Windows/Linux/Mac	TLS/DTLS	0-RTT/PSK
mbedTLS	C	ARM	Apache	嵌入式操作系统	TLS/DTLS	0-RTT/PSK
CycloneSSL	C	Oryx Embedded	GPL	嵌入式操作系统	TLS/DTLS	0-RTT/PSK
wolfSSL	C	wolfSSL	GPL	嵌入式操作系统	TLS/DTLS	0-RTT/PSK
RebexTLS	C#	Rebex	商用	Windows	TLS	PSK
Botan	C++	独立开发者	BSD	Windows/Linux/Mac	TLS/DTLS	0-RTT/PSK
fizz	C++	Facebook	BSD	Windows/Linux/Mac	TLS	0-RTT/PSK
Tris	Go	Cloudflare	BSD	Windows/Linux/Mac	TLS	0-RTT/PSK
JSSE/JDK	Java	Oracle	OTN	Windows/Linux/Mac	TLS/DTLS	PSK
tlslite-ng	Python	tlsfuzzer	BSD/LGPL	Windows/Linux/Mac	TLS	0-RTT/PSK
tttls1.3	Ruby	独立开发者	MIT	Windows/Linux/Mac	TLS	0-RTT/PSK
SwiftTLS	Swift	独立开发者	MIT	Mac	TLS	0-RTT

4.2 基于程序分析的 TLS 软件缺陷检测

程序分析技术可以用来帮助研究人员发现 TLS 协议软件或应用程序中的潜在错误和漏洞, 常用的包括污点分析、插桩、符号执行等分析方法, 其原理如图 9 所示.

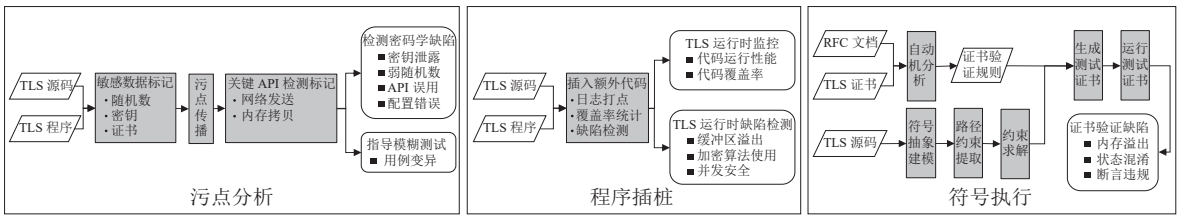


图 9 基于程序分析的 TLS 缺陷检测基本原理

污点分析是一种用于检测数据流安全性漏洞的技术. 它通过静态分析或动态追踪敏感数据 (即 Source 点) 到关键 API (即 Sink 点) 使用的全过程, 以确定是否有导致不安全行为的数据流动. 污点分析基本步骤包含 Source/Sink 定义和污点传播. 在 TLS 上下文中, 污点分析可以用来检查密钥材料、证书和其他敏感信息是否可能泄露给未授权的实体. 例如, 它可以检测出是否有不安全的 API 调用或者不当的数据处理可能导致密钥暴露. 此外, 污点分析还可以指导模糊测试进行针对性的测试用例变异.

程序插桩是指在源代码或者二进制级别插入额外代码 (如 log 函数), 以便监控程序在动态运行时的行为. 在 TLS 协议软件安全检测中, 插桩可以帮助识别协议实现代码中的加密算法使用是否正确, 或者是否存在缓冲区溢

出等运行时错误. 通过动态分析, 插桩能够捕捉到静态分析无法发现的问题, 特别是在多线程环境中复杂的并发条件下. 此外, 程序插桩也用于模糊测试中监控和收集反馈信息, 如运行性能和代码、路径覆盖率等.

符号执行是一种辅助程序分析技术, 它通过模拟程序路径的执行而不需要具体值. 在执行过程中, 程序状态被视为一组数学公式或抽象符号, 这些符号描述了程序变量间的关系. 程序的每条路径可用这些符号来表示其路径约束. 因此, 判断各路径上的可达性问题, 就可转换为路径约束的可满足性问题. 符号执行有助于发现 TLS 程序边界条件下和证书相关漏洞, 如内存溢出、状态混淆、断言违规, 这些漏洞难以通过传统测试方法发现.

近年来, 针对 TLS 1.3 代码库中实现缺陷的研究取得了显著进展, 特别是在密码程序分析 (cryptographic program analysis, CPA) 和证书验证缺陷检测方面. 研究者基于程序分析技术和自动化测试手段提高 TLS 软件缺陷检测能力, 并成功在 OpenSSL 库中发现了多个严重安全问题.

4.2.1 密码学缺陷检测

密码协议通常被期望是可证明安全的. 然而, 由于各种实现缺陷, 这种安全保证在实践中往往不足. 而加密实现中的逻辑缺陷, 往往更难以被发现.

2021 年, Rahaman 等人[88]提出了 CPA 的概念, 通过使用程序分析技术在编译时检测这些实现缺陷. CPA 的核心思想是密码实现中的许多缺陷都可以映射到元属性的违反. Rahaman 等人将实现密码属性所必需的程序属性称为元属性, 这些元属性的违反可以在编译时被识别出来, 因此可以用于检测逻辑错误. Rahaman 等人研究了现有的关于密码实现缺陷的文献, 并推导出了 25 个相应的元属性. 为了实例化 CPA 的抽象范式, Rahaman 等人基于 DFA 开发了一种规范语言, 并表明大多数元属性都可以用该语言来表达. 最后实现了 TAINTCRYPT 检测工具, 通过静态污点分析在编译时识别 C/C++密码库中的元属性违规. TAINTCRYPT 在 5 款使用 OpenSSL 库的应用 (包括 httpd/curl 等) 中检测出 10 个安全违规问题.

4.2.2 证书验证缺陷检测

主机名验证是 TLS 连接过程中证书校验的关键组成部分, 它通过检查服务端的主机名与 X.509 证书中所包含的主机名称是否一致, 来验证远程服务端的身份. 由于存在许多特性和边缘情况 (如通配符、IP 地址、国际域名等), 主机名验证变得尤为复杂和具有挑战性. 研究者提出了基于自动机学习和差分测试的缺陷检测方法, 前者自动从证书模板中学习验证规则, 后者从 RFC 标准中提取规则并通过动态符号执行生成证书测试用例.

2017 年, Sivakorn 等人[89]基于自动机学习算法提出了一种新型黑盒测试框架 HVLearn, 用于分析 TLS 协议软件在握手过程中的主机名验证是否符合 RFC 协议规范. HVLearn 测试了 9 个 TLS 软件及应用, 并发现了 8 个 RFC 不合规问题. HVLearn 利用多个证书模板 (将通用名称 CN 设置为特定模式的证书) 来测试相应规范中的不同规则. 对于每个证书模板, HVLearn 使用自动机学习算法来推断一个 DFA 自动机, 该自动机描述了与给定证书的 CN 名匹配的所有主机名的集合. 一旦为证书模板推断出模型, HVLearn 就会通过查找与其他实现推断出的模型之间的差异或根据规范中派生出的基于正则表达式的规则来检查模型中的错误.

2018 年, Chen 等人[90]和 Tian 等人[91]基于差分测试实现了证书校验检测工具 RFCcert. RFCcert 首先自动从协议 RFC 标准中提取证书验证规则, 然后通过动态符号执行技术生成证书测试用例, 最后用这些用例对 TLS 库进行测试以揭示潜在的证书校验漏洞或缺陷. RFCcert 测试了 6 款 TLS 库, 发现 9 个安全问题, 包括 1 个新的 CVE 漏洞.

2023 年, Nie 等人[92]提出了基于代码覆盖率引导的证书校验模糊测试方法, 测试了 5 款 TLS 库并发现了 11 个新的安全问题.

2023 年, Chen 等人[93]进一步改进了之前的研究[90], 提出了基于搜索的差分测试方法. 该方法无需从 RFC 标准中提取证书验证规则, 并能区分和检测证书解析和证书验证过程中的缺陷. Chen 等人测试了 2 款 TLS 库, 发现了 25 个安全问题.

4.3 基于模糊测试的 TLS 软件缺陷检测

模糊测试是一种软件缺陷检测和漏洞挖掘技术, 通过对目标程序注入大量的随机字符序列作为输入, 来检测

程序是否会出现非预期结果. 目前, 模糊测试已经成为漏洞挖掘的一种重要方法, 在 2019 年 Google Project Zero[94] 的报告中, 通过模糊测试发现的漏洞数量比例高达 37%.

在程序安全分析领域中, 模糊测试技术一直都是热点研究, 近年来 4 大安全顶会上也都发表了 TLS 协议模糊测试相关的论文. 相关团队一方面研究如何提高模糊测试的状态空间覆盖深度, 提出了面向协议状态的反馈机制; 另一方面研究如何拓宽模糊测试的应用范围和提高挖掘效率, 提出了基于交互式流量分析的黑盒测试方法. 同时, 为了进一步提升模糊测试的缺陷检测能力, 研究人员结合诸如符号执行、污点分析以及自动机等程序分析理论和技术, 提升了模糊测试在执行效率、覆盖率等评估指标上的表现, 进而获得了更好的检测效果.

4.3.1 TLS 协议软件模糊测试

传统的模糊测试通常都是面向无状态系统, 如图像处理库或者简单的命令行程序. 这些程序通常只接收单个输入, 其测试指标常以覆盖更多的代码覆盖率或节点覆盖率为目标. 随着安全需求的增长, 有状态系统特别是网络协议逐渐受到测试人员的关注. 有状态系统是指将一系列消息作为输入并产生输出的系统. 与无状态系统不同的是, 有状态系统的每个输入都可能导致系统内部状态发生巨大变化.

TLS 协议模糊测试比无状态系统更难, 体现在两个方面. 首先, 因为内部状态变化增加了需要探索的状态空间, 模糊测试器可能很难达到更深的状态. 其次, 由于反馈机制的无效或低效, 以及协议状态空间探索技术不足, 传统的模糊测试技术在测试 TLS 协议时面临着巨大挑战.

图 10 是 TLS 协议软件模糊测试的基本过程, 包含预处理、输入构造、输入选择、测试评估、结果分析这 5 个处理流程. 预处理是非核心流程, 其分别用于测试前的铺垫准备和测试后的分析总结. 输入构造、输入选择和测试评估属于核心测试流程, 用于生成高质量的测试用例并对目标程序进行测试和评估.

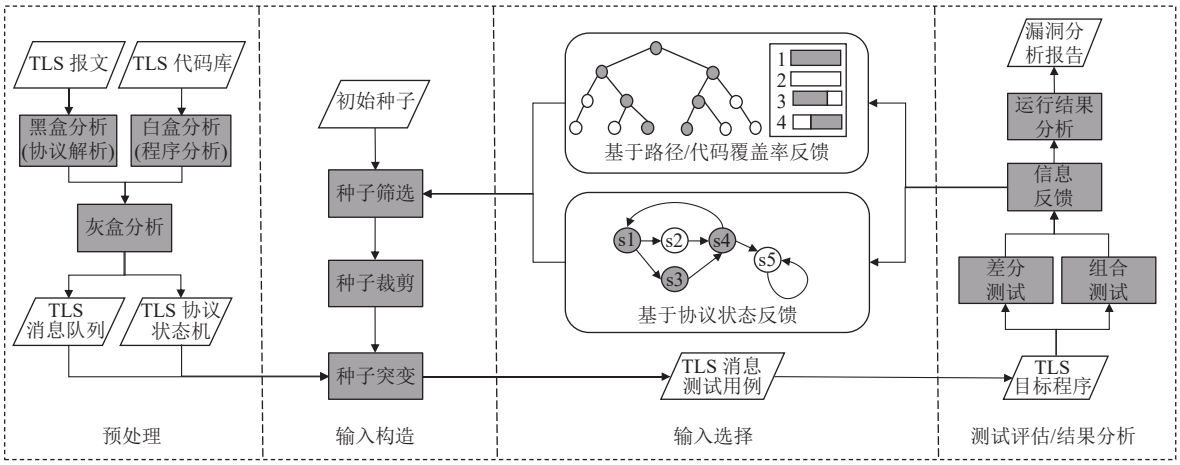


图 10 TLS 协议软件模糊测试基本过程

4.3.2 TLS 软件预处理

在测试 TLS 软件前, 首先需要对 TLS 程序进行预处理. 其目的是搜集 TLS 程序的相关信息, 获取 TLS 协议状态机模型并制定模糊测试策略, 以及为监控 TLS 程序在实际测试中的运行状态做必要准备, 以便在后续阶段对测试用例进行优化和调整策略. 通常, 该阶段的监控和分析是通过插桩、符号执行或污点分析等程序分析技术来实现.

通常, 业界根据预处理阶段对 TLS 程序的分析和了解程度, 将其分为白盒、黑盒和灰盒测试. 在白盒测试中, 预处理阶段会分析 TLS 软件的全部源代码, 黑盒测试通常只分析 TLS 软件二进制程序的输入数据格式 (如 PCAP 数据报文), 而灰盒测试会分析部分源代码及二进制输入数据. 在黑盒测试中, 研究人员常使用状态机学习算法来获取 TLS 软件的 FSM 状态机模型. 状态机学习算法能够假设协议的初始状态机模型, 根据目标系统的输入和响应, 不断修正模型, 使其与目标系统的差距越来越小.

4.3.3 TLS 软件输入构造

在预处理完成后, 第 2 阶段是构造 TLS 程序的输入用例. 在该阶段, 首先获取一定数量的初始种子. 种子可以由算法自动生成或由用户指定; 随后确定种子的能量分配策略、种子的优先级以及种子的裁剪和突变策略; 最后依据这些种子获得大量的输入用例, 即 TLS 协议消息.

2023 年, Luo 等人[95]提出了一种面向数据包序列的黑盒模糊测试工具 Bleem, 通过新的动态反馈机制和系统状态跟踪图来引导模糊测试生成消息序列, 从而大大提升了 TLS 软件输入用例的质量. 传统的协议软件测试工具只专注于单个数据包生成, 而 Bleem 会在序列级别上生成多个数据包, 从而遍历更多的协议状态空间. Bleem 通过动态分析系统输出序列来提供有效的反馈机制, 通过及时涵盖状态空间跟踪来引导模糊测试. Bleem 采用交互式流量信息生成协议逻辑感知的数据包序列, 因此可以黑盒测试大多数通用协议而无需源码. Bleem 测试了 15 个 TLS 和 QUIC 协议库, 结果表明 Bleem 有更高的分支覆盖率 (最高提高了 174.93%), 发现了 15 个 CVE 漏洞, 其中包括 10 个新的 CVE 漏洞.

4.3.4 TLS 软件输入选择

第 3 阶段是对用例进行筛选. 其目的是过滤无效或者重复数据, 提高模糊测试效率. 为此, 研究人员开发了多种选择策略, 其中最常用的一种算法策略是寻找能最大化特定覆盖率 (如代码覆盖率或路径覆盖率) 的最小集合, 又称为 miniset. 在筛选过程中, 算法逐步将无法扩大覆盖率的用例从集合中删除.

为了更高效地遍历 TLS 协议状态空间, 研究人员提出了面向协议状态和基于 DY 模型来指引输入构造, 从而通过变异或协议语义识别来生成更高质量的消息序列, 提升了 TLS 软件缺陷检测效果.

(1) 面向协议状态的缺陷检测

2015 年, de Ruiter 等人[96]首次提出了面向 TLS 协议状态的黑盒模糊测试, 通过使用状态机学习技术从 TLS 库中推断协议状态机, 然后检查推断状态以寻找程序中存在的逻辑缺陷. de Ruiter 等人测试了 14 个 TLS 库, 在 3 款 TLS 库 (GnuTLS、Java 安全套接字扩展和 OpenSSL) 中均发现了安全漏洞.

2016 年, Somorovsky[97]提出了 TLS 模糊测试框架 TLS-Attacker, 用于评估 TLS 库的安全性. TLS-Attacker 允许安全模糊测试人员使用简单的接口创建自定义 TLS 消息流和修改消息内容, 以测试 TLS 库的行为. 基于该框架, Somorovsky 提出了一个两阶段的模糊方法来评估 TLS 服务端行为. 该方法可以自动搜索加密失败和内存溢出漏洞的情况, 经过测试它在 3 个广泛使用的 TLS 库 (包括 OpenSSL、Botan 和 MatrixSSL) 中发现了 7 个字节填充和溢出问题, 包括 2 个新的 CVE 漏洞.

近年来, 越来越多的协议开始依赖 UDP 协议. 与 TCP 协议相比, UDP 提供了诸如简单性和低延迟等性能优势. 这推动了其在 VoIP、隧道技术、物联网和 QUIC 协议中的应用. 为了保护这些场景中的敏感数据传输, DTLS 协议[98]被提出, 以作为 TLS 协议在 UDP 上的变体. DTLS 的主要挑战是支持 UDP 的无状态和不可靠传输. 这迫使协议设计者做出影响 DTLS 复杂性的选择, 并加入在众多 TLS 分析中无需考虑的特性.

2020 年, Fiterau-Brostean 等人[99]首次提出面向 DTLS 协议的状态模糊测试工具 DTLS-Fuzzer[100]. DTLS-Fuzzer 对 TLS-Attacker 测试框架进行了扩展, 增加了对 DTLS 协议的支持, 以适应底层 UDP 层的无状态和不可靠特性. Fiterau-Brostean 等人使用 DTLS-Fuzzer 学习了 13 个 DTLS 库的状态机模型, 发现了 4 个安全问题, 其中包括 1 个历史 CVE 漏洞和 1 个新的身份验证绕过 CVE 漏洞.

灰盒模糊测试已广泛应用于无状态协议并取得了巨大成功. 然而, 在对能够记住和存储交互细节的有状态网络协议程序进行模糊测试过程中, 大多数最先进的灰盒测试工具也存在速度慢和状态深度覆盖浅的问题.

2022 年, Ba 等人[101]提出面向有状态协议的灰盒模糊测试. Ba 等人指出某些错误只能在特定状态下暴露出来, 因此模糊测试器需要提供一系列特定事件作为输入来触发该错误, 并将这些错误称为“有状态”错误. 测试有状态协议的关键挑战是在没有明确协议规范的情况下如何覆盖状态空间. Ba 等人认为协议软件中使用的状态变量通常出现在枚举类型变量中, 这些变量的值 (状态名称) 来自命名常量, 并通过人工分析 Top-50 开源协议库证实了该猜测. 通过对这些状态变量进行自动识别, 并在模糊测试期间跟踪分配给它们的值序列, 以生成已探索状态空间的“映射图”. 最终实验测试了 8 个音视频传输和网络传输相关的开源协议库 (包括 OpenSSL 和 mbedTLS 库), 发现了

12 个安全问题, 其中包括 8 个新的 CVE 漏洞.

(2) 基于 DY 模型引导的缺陷检测

基于 DY 模型的形式化验证方法, 虽然擅长发现协议逻辑缺陷, 但仅适用于抽象规范模型. 针对网络协议软件完全自动化的形式化验证仍然存在巨大挑战.

针对此问题, 2024 年 Ammann 等人[102]提出了基于 DY 模型引导的模糊测试, 可以检测加密协议在设计及其实现中的逻辑攻击. 其主要思想是将 DY 模型攻击者的抽象 DY 执行集视为可能的测试用例, 并使用一种基于变异的模糊测试器来遍历这个集合. DY 模糊测试器将每个抽象执行具体化, 以便在待测试程序上进行测试. 这种方法使得能够以更结构化和与安全相关的级别对表示为正式术语的消息 (例如, 解密消息并使用不同的密钥重新加密它) 进行推理, 从而发现协议软件中的逻辑缺陷. Ammann 等人实现了 DY 协议模糊测试器 tlspuffin, 并测试了 OpenSSL、LibreSSL 和 wolfSSL 库, 最终发现了 7 个 CVE 漏洞, 其中 4 个是新的 CVE 漏洞.

4.3.5 TLS 软件测试评估

筛选用例后, 便可以用这些输入测试用例对 TLS 程序进行测试评估. 该阶段的主要工作是设计合适的测试方法、测试指标和选择安全检测器.

研究人员提出了基于差分测试和组合测试的测试方法, 提升模糊测试效率. 合适的指标可以从侧面反映模糊测试执行的质量, 这是制定模糊测试执行策略的重要依据. 常用的指标包括两类, 一类是覆盖率指标, 如代码覆盖率、路径覆盖率、TLS 协议状态空间覆盖率等; 另一类是时间指标, 如执行时间、发现漏洞平均时间等. 经典的模糊测试工具, 如 AFL[103]、libFuzzer[104]都是基于覆盖率引导. 另外, 选取合适的安全检测器也对模糊测试质量至关重要. 典型的检测器包括以下几类, 内存错误类的用于检测缓冲区溢出、越界读写等内存相关漏洞, 输入验证类的用于检测 SQL 注入、XSS 注入等 Web 漏洞, 未定义行为类的用于检测带符号整数溢出等漏洞, 语义差异用于检测信息泄漏等漏洞.

在基于反馈的协议软件模糊测试中, 考虑到协议软件具有高度规范性的特点, 即使遵循相同规范, 不同开发者也可能创建出多种实现方式. 因此, 研究人员提出了基于差分测试和组合测试的测试方法, 来进一步提升 TLS 软件测试效率. 差分测试有助于辨别不同实现的细粒度差异, 从而发现潜在漏洞或不一致性.

2017 年, Walz 等人[105]提出了基于差分测试的 TLS 握手协议黑盒测试方法, 可用于生成大量有效且多样化的 TLS 握手消息. 相比其他传统的模糊测试工具如 TLS-Attacker[97]、NEZHA[106]、tlsfuzzer[107]和 scapy-SSL_tls[108], 该方法生成的握手消息能够引发更多的响应差异, 且生成的 TLS 握手消息拥有更高的代码覆盖率. Walz 等人测试了 OpenSSL、BoringSSL、wolfSSL、mbedTLS 和 MatrixSSL 库, 发现了 16 个安全问题.

2022 年, Maehren 等人[109]提出了基于组合测试的 TLS 测试工具 TLS-Anvil. Maehren 等人认为 TLS 实现缺陷主要来源于 TLS 规范的复杂性, 允许指数级多的可能参数组合. 组合测试 (combinational testing, CT) 是一种控制这种复杂性的技术, 但由于参数之间的语义依赖关系, 很难将其应用于 TLS 库的安全检测中, 主要挑战是测试预言机问题, 即确定给定测试输入的软件观测行为是否正确. TLS-Anvil 可以高效且系统地测试参数值组合, 并通过动态提取基于 TLS 特定参数值交互的特定于实现的输入参数模型 (IPM) 来克服预言机问题. TLS-Anvil 测试了 13 个 TLS 库, 发现了 23 个安全风险, 包括 wolfSSL 和 MatrixSSL 的两个新 CVE 漏洞.

2024 年, Zhao 等人[110]提出了一种基于差分测试的模糊测试框架 TLS-DeepDiffer, 专注于检测 TLS 协议状态中的逻辑问题, 并能深度遍历协议状态空间. Zhao 等人测试了 27 个 TLS 库, 发现了 24 个安全问题, 包括 4 个历史 CVE 漏洞和 1 个新的 CVE 漏洞.

4.4 TLS 软件安全攻击

TLS 1.3 协议机制安全在草案阶段得到了研究人员的充分分析和证明, 不安全机制已经被移除. 因此, 现有针对 TLS 1.3 协议的攻击大多是针对共存的低版本 TLS 协议实施降级攻击和跨协议攻击, 或针对不安全的协议配置和实现方式 (如 0-RTT 机制未实施防重放攻击措施) 进行攻击. 下面简单介绍这些攻击方式.

(1) 0-RTT 重放攻击

0-RTT 早期数据面临重放攻击风险[7], 为了降低此风险, TLS 协议标准建议只使用 0-RTT 发送非机密数据, 且

这些数据操作是幂等的 (即数据可以安全的多次执行而不改变结果状态). 否则, 服务端需要添加额外的防重放攻击措施, 如在 0-RTT 数据中加入序列号或时间戳来验证数据有效性.

(2) PSK 密钥泄露攻击

PSK 机制允许客户端和服务器通过带外方式预先协商一个密钥, 然后在 TLS 握手中使用它来进行身份验证和加密. 尽管 PSK 提供了性能上的优势, 但如果不正确实现或配置, 也可能引入潜在的安全风险. 如 PSK 密钥本身强度不够可能被暴力破解, 或者客户端与多个服务器共享相同的 PSK 密钥, 以及结合使用 PSK 和证书认证可能导致身份验证机制之间的混淆. 为此, TLS 协议标准建议不要同时使用外部 PSK 认证与证书认证, 除非有明确的扩展机制允许并定义了如何安全地结合这两种认证方式. 本文在第 3.3.2 节中介绍的 Selfie 攻击[64]是 TLS 1.3 标准正式发布后被发现的第 1 个针对 PSK 的攻击方式.

(3) 侧信道攻击

侧信道攻击是指通过分析加密操作产生的副作用, 如执行时间、功耗模式或电磁泄漏等来获取密钥及其他敏感信息. 虽然 TLS 1.3 引入了 AEAD 身份认证加密、统一的错误处理和快速终止有问题的握手连接, 来简化开发者编写抗侧信道攻击代码的过程和减少某些类型的侧信道攻击的可能性, 但并不能完全防止该攻击.

(4) 签名算法攻击

此外, 还有一些研究者针对签名算法进行了攻击尝试.

2023 年, Mus 等人[111]提出了一种针对 RSA 和 (EC)DSA 签名的新型攻击 Jolt. Jolt 攻击首先在签名生成过程中注入错误获得错误签名, 再通过签名验证原语来纠正错误签名, 并在此过程中推断出签名密钥. Mus 等人测试了 5 款 TLS 库及加密库, 验证了 Jolt 攻击可行性.

2023 年, Sullivan 等人[112]针对 RSA 和 (EC)DSA 签名算法进行了大规模分析, 使用被动和主动网络测量和分析了数千万台主机生成的 27 亿个签名. Sullivan 等人验证了使用 PKCS#1 v1.5 填充方式的 RSA 签名算法的脆弱性, 并基于硬件故障注入技术恢复出了 RSA 签名密钥.

(5) 加密流量分析

随着 QUIC 和 TLS 1.3 协议不断发展, 越来越多的应用程序使用 TLS 协议加密传输数据, 给传统流量分析技术带来很大挑战. 近期, 研究者利用深度学习模型从 TLS 加密流量中提取丰富特征来实现加密流量检测和分类.

虽然现有的深度学习模型在加密流量中取得了不错的分类效果, 但在不同的网络环境中表现出明显的性能下降. 2023 年, Xie 等人[113]提出了一种加密流量分类方法 Rosetta. Rosetta 通过 TCP 感知流量增强机制和自我监督学习来理解隐式 TCP 语义, 从而提高了在不同环境下对 TLS 加密流量的分类效果.

2023 年, Niere 等人[114]提出利用 TLS 1.3 记录分片来规避 HTTPS 流量检测, 从而绕过防火墙拦截.

2024 年, Xue 等人[115]将封装的 TLS 握手作为指纹来检测混淆的代理流量. Xue 等人深入分析了现有的各种形式的代理及隧道工具, 发现嵌套的加密协议最终都会产生一个特定指纹, 即封装的 TLS 握手, 并且可以在加密流量中被识别出来. 基于这个发现, Xu 等人构建了一个相似性分类器, 能够从加密流量流中分离出这种特有指纹, 从而有效地检测出加密代理.

4.5 TLS 软件性能优化

由于 TLS 1.3 协议中的握手及身份验证大量使用椭圆曲线加密算法, 这些算法往往非常耗时. 于是研究者围绕椭圆曲线加速进行研究, 提出了基于硬件指令集的优化方案.

2024 年, Zhang 等人[116]基于 Intel 处理器 AVX 指令集优化 ENG25519 算法, 最大限度发挥指令并行能力来加速椭圆曲线算法, 如有限域和标量乘法, 并基于 OpenSSL 库进行了测试. 基准测试验证了该算法可以将 TLS 1.3 握手速率提高 25%–35%, DNS over TLS (DoT) 查询的峰值服务器吞吐量提高 24%–41%.

4.6 对比分析

针对上述 TLS 1.3 软件缺陷检测相关研究, 本文从测试类型、文献发表时间和会议期刊、检测目标、检测技术、检测代码库数量和检测效果等方面对这些工作进行了综合对比和分析, 如表 8 所示.

表 8 TLS 1.3 协议软件缺陷检测相关研究对比

测试类型	文献	发表时间	发表会议/期刊	检测目标	检测技术	检测数量	协议版本		发现问题			
							1.2	1.3	I	NI	C	NC
白盒	[97]	2016	CCS	密码学攻击/内存溢出	静态污点分析	3	√	×	7	7	2	2
	[88]	2021	TDSC	密码学逻辑错误	静态污点分析	5	√	×	10	10	0	0
黑盒	[96]	2015	UsenixSEC	状态转换异常	自动机	14	√	×	3	3	0	0
	[89]	2017	S&P	证书验证缺陷	自动机	9	√	×	8	8	0	0
	[105]	2017	TDSC	握手异常	差分测试	5	√	×	16	16	0	0
	[90]	2018	ICSE	证书验证缺陷	差分测试/符号执行	6	√	×	9	9	1	1
	[99]	2020	UsenixSEC	状态转换异常	自动机	13	—	—	4	4	2	1
	[109]	2022	UsenixSEC	测试断言	组合测试	13	√	√	23	23	2	2
	[92]	2023	TOSEM	证书验证缺陷	差分测试/覆盖率引导	5	√	√	11	11	0	0
	[93]	2023	ISSTA	证书验证缺陷	差分测试	2	√	√	25	25	0	0
	[95]	2023	UsenixSEC	状态转换异常	自动机/面向消息序列	15	√	√	0	0	15	10
	[110]	2024	SANER	状态转换/逻辑错误	差分测试	27	√	×	24	24	5	1
灰盒	[101]	2022	UsenixSEC	状态转换异常	自动机	8	√	√	12	12	8	8
	[102]	2024	S&P	逻辑错误	自动机/DY模型引导	3	√	√	0	0	7	4

注: —表示该文献研究DTLS协议; I表示安全问题数量; NI表示新问题数量; C表示CVE漏洞数量; NC表示新漏洞数量

从表 8 中可以看到, 研究人员在 2015–2024 年对 TLS 协议软件缺陷展开了大量研究, 提出了很多有效的检测方法并获得了非常不错的检测效果. 状态转换异常、证书验证缺陷是研究人员的首要检测目标, 也是 TLS 软件代码实现时最常见的安全问题.

在状态转换异常检测中, 研究人员主要通过自动机学习技术来推断 TLS 协议的状态机模型, 并结合模糊测试技术挖掘 TLS 协议软件中的状态转换错误.

在证书验证缺陷检测中, 研究人员基于自动机学习方法来自动生成畸形的证书测试用例, 并结合差分测试技术来发现证书校验问题. 在这些研究中, 大部分都支持 TLS 1.2 代码库检测, 2020 年之后的工作开始支持检测 TLS 1.3 代码库.

为了客观评估上述工作的实际检测效果, 本文将这些文献所发现的问题数量进行了简单的统计. 本文将问题分为两大类: 第 1 类为可能引起安全隐患或影响协议交互功能的安全问题, 包含了 security issue 和 security bug 这一类问题; 第 2 类为可被攻击者利用进行实际攻击的安全漏洞, 即 CVE 漏洞. 同时, 为了区分历史 (重复) 问题和新问题, 本文在统计时也进行了区分. 从表 8 中可以看出, Luo 等人[95]提出的面向消息序列的模糊测试方法和 Chen 等人[93]提出的差分测试方法获得了较好的安全检测效果, 值得研究人员重点关注和借鉴.

4.7 小 结

综上所述, 为了保障 TLS 软件实现的安全性, 研究人员综合程序分析、模糊测试和组合测试等技术来识别 TLS 软件库中的潜在错误和漏洞, 提出了一些有效方法并挖掘到了多个高危漏洞. 在基于程序分析的缺陷检测中, 研究者利用污点分析、符号执行、自动机学习来检测 TLS 软件库中的密码学缺陷和证书验证缺陷. 在基于模糊测试的安全检测中, 研究者一方面提出了面向协议状态和消息序列的黑盒模糊测试技术方法, 不仅优化了模糊测试过程中的反馈机制, 也有效提升了协议状态空间覆盖和应用范围; 另一方面提出了基于差分测试和 DY 模型引导的测试技术, 提升了测试的针对性和有效性, 从而可以更高效地发现深层次的逻辑安全问题.

5 TLS 1.3 协议应用配置安全

协议应用配置同样是 TLS 1.3 协议安全研究的重要组成部分, 主要关注 TLS 协议在实际部署和应用配置过程中的安全问题等. 服务端应用主要涉及是否使用了不安全的协议版本、加密算法以及会话票据安全等. 客户端应

用包含密码学 API 误用和自定义证书验证等场景. 前者是客户端应用程序开发过程中的不当使用. 后者是指某些操作系统如安卓允许应用程序自定义证书验证过程, 而不安全的证书验证容易使应用程序遭受中间人攻击、证书伪造等安全攻击, 使得用户敏感数据未授权访问和泄露.

5.1 服务端应用配置安全

5.1.1 协议配置安全

服务器不正确地配置 TLS 1.3 可能会削弱其提供的安全性. 例如, 服务器使用了弱加密算法或者启用了不安全的协议版本或功能, 包括 TLS 旧版本及版本降级的支持, 这都可能为攻击者提供可乘之机. 为此, 业界提供了一些检测工具来判定服务器配置是否安全.

SSLtest (SSL server test)[117]、SSLyze[118]、testSSL[119]和 TLS-Scanner[120]是工业界著名的 TLS 配置安全扫描工具, 不仅拥有非常全面的 TLS 协议扫描能力, 还能提供详尽的扫描报告. 首先, 它们可以扫描服务器支持的 TLS 协议版本和加密套件, 并分析哪些是弱加密或不推荐使用的; 其次, 还可以进行 TLS 协议漏洞检测, 包括已知的 TLS 协议漏洞如 Heartbleed、POODLE、Logjam、DROWN 等. 此外, 还支持证书链扫描, 以验证服务器提供的 X.509 证书链的有效性, 包括检查证书是否过期、是否有正确的主机名匹配、是否被撤销等.

此外, 学术界也提出了一些检测方法. 2019 年, Merget 等人[121]针对 TLS 协议 CBC 加密模式填充攻击首次实现了大规模扫描工具 TLS-Crawler. 2023 年, Sosnowski 等人[122]提出了 TLS 协议扫描和指纹识别工具 DissecTLS. DissecTLS 不仅可以扫描 TLS 协议配置参数, 还可以对其进行指纹建模并动态创建扫描探测, 以实现对恶意 C&C 服务器的指纹检测. 2024 年, Sosnowski 等人[123]基于主动测量技术对 DissecTLS 进行了进一步优化, 提出 EFACTLS. EFACTLS 通过使用经验优化策略, 充分挖掘并利用每次 TLS 握手中获得的信息, 最大限度地降低 TLS 测量成本.

5.1.2 会话票据安全

会话票据机制通过在后续连接中重用之前的加密参数而提升了连接效率. 然而, 协议标准只定义了其基本工作原理, 而并未说明如何具体实现. 服务器实现该机制时所采用的数据格式、加密算法和密钥管理都会直接影响 TLS 协议的实际安全性. 为此, 研究人员针对此问题进行了研究和验证.

2023 年, Hebrok 等人[124]针对 TLS 1.2 和 TLS 1.3 的会话票据具体实现进行了深入研究, 通过对 9 款 TLS 代码库和 3 个 Web 服务器软件中的会话票据数据格式和加密算法进行系统分析, 发现一些代码库存在弱安全实现, 导致攻击者能解密 TLS 会话. Hebrok 等人基于 TLS-Scanner 和 TLS-Crawler 工具对 TLS 服务器进行了大规模扫描, 发现 Amazon 网站服务器中的一个实现缺陷, 该缺陷允许攻击者解密至少两千台服务器的 TLS 流量.

2024 年, Radoy 等人[125]针对会话票据是否会受到分区预言机攻击[126]进行了研究. 分区预言机攻击使用 AEAD 认证加密算法 (如 AES-GCM) 中的 Carter-Wegman MAC 属性来恢复密钥. Radoy 等人测试了 6 款使用 AES-GCM 或 ChaCha20-Poly1305 加密算法的 TLS 代码库, 发现 4 款 TLS 代码库容易受到预言机攻击.

5.2 客户端应用配置安全

5.2.1 应用程序密码学 API 误用检测

TLS 加密是所有安全系统不可或缺的重要组成部分. 然而, 由于各种加密 API 的错误使用方式, 使得应用程序所期望的 TLS 安全保证在实践中往往无法实现, 严重威胁着应用安全. 因此, 研究人员借助污点分析技术来检测应用程序中的密码学 API 误用.

2019 年, Rahaman 等人[127]针对 Java 语言提出了基于静态污点分析的密码学 API 误用检测工具 CryptoGuard, 用于检测 Java 程序中的密钥暴露、可预测的随机数以及脆弱的证书验证等行为. Rahaman 等人首先分析了 16 种典型的密码学 API 误用和对应的分析方法; 然后通过前向和后向程序切片来识别和提取 Java 程序中的密码学相关代码; 最后通过流敏感、上下文敏感和字段敏感的污点分析实现了一组 API 误用检测算法, 大大降低了误报. Rahaman 等人对 46 个大型 Apache 项目和 6 181 个 Android 应用进行了测试, 发现了 1 277 个误用行为.

以往的许多研究都针对 Java、C/C++和 Python 语言中的密码学 API 误用情况, 但 Go 语言内的类似问题仍未被挖掘. 2022 年, Li 等人[128]首次针对 Go 语言设计并实现了检测工具 CryptoGo. CryptoGo 利用污点分析技术来检

测大规模 Go 语言项目中的 12 种密码学误用情况. CryptoGo 对 GitHub 上的 120 个开源 Go 语言项目进行了检测, 发现 83.33% 的 Go 语言加密项目至少存在一处密码学误用.

为了评估 API 误用检测工具的效果, 2022 年 Ami 等人[129]提出了 MASC 测试框架. 该框架使用变异测试对 9 个经典的密码学误用检测工具进行了系统测试和评估. Ami 等人对以往论文中的 105 种密码学 API 误用行为进行了梳理, 归为 9 大类; 然后对这些误用行为进行了系统建模和抽象描述, 通过对其中的加密运算符进行变异产生了两万多个变异体; 最后用检测工具对变异体进行测试, 发现了 19 个新的漏洞.

2024 年, Chen 等人[130]对密码学 API 误用检测进行了深入研究, 针对以往论文中所研究的 API 误用行为以及检测工具规则、安全模型和具体实现进行了详细的比较和系统分析. Chen 等人分析和解释了这些工具的检测结果与实际漏洞之间的差距, 以及为改进检测工具的精度和可用性提供了可能的方向. Chen 等人指出, 以前工作的小规模检查存在一些盲点, 导致其检测规则、模型和实现中的问题被忽视, 进而导致了误报和漏报.

5.2.2 应用程序验证过程检测

加密协议中的常见验证过程如消息签名、MAC 认证或椭圆曲线点的验证, 对于协议的整体安全性至关重要. 然而, 开发者在具体实现时往往容易省略这些验证步骤而导致安全风险. 针对这一类安全问题, 研究者提出了一些检测方法. 2020 年, Wazan 等人[131]对业界的防火墙产品证书验证进行了研究, 发现所有 TLS 流量拦截防火墙都忽视了证书验证, 如服务器证书是否被吊销, 这给 WEB 用户带来了一定风险. 2023 年, Fischlin 等人[132]提出了一种可验证的验证步骤检测方法, 通过确认码机制来自动检测椭圆曲线中的签名、MAC 或参数验证的实现代码中是否意外忽略或跳过了某些验证步骤, 从而有效帮助开发者规避这一类实现缺陷.

5.2.3 应用程序自定义证书验证检测

现代操作系统如安卓系统, 允许开发人员根据不同安全需求定制证书验证过程. 但是不安全的自定义实现, 往往让应用程序更容易遭受中间人攻击. 为了缓解这些问题, Google 推出了网络安全配置 (network security configuration, NSC). NSC 是一种基于配置文件的方法, 用来帮助开发人员提高自定义证书验证逻辑的安全性.

2021 年, Oltrogge 等人[133]基于静态分析技术对 Google Play 应用市场上的 133 万安卓应用进行了系统分析, 发现使用 NSC 配置的安卓应用有 9 921 个, 其中 88.87% 的应用由于 NSC 配置不当反而降低了安全性, 只有 0.67% 的应用正确实现了证书验证逻辑.

2024 年, Pourali 等人[134]同样针对安卓应用中的证书验证问题进行了深入研究, 发现大多数应用都存在证书验证劫持的行为, 即使用不正确的验证逻辑或者完全忽略证书验证错误来覆盖全局的默认验证函数, 从而影响整个应用的 TLS 连接. Pourali 等人结合静态和动态程序分析实现了 Marvin 工具, 对国内外安卓应用市场中的 7 826 个应用进行了大规模检测, 结果显示 55.3% 的国内应用和 6.4% 的国外应用都存在不安全的证书验证过程.

5.3 对比分析

针对上述的 TLS 1.3 应用配置安全相关工作, 本文从文献发表时间、发表会议期刊或在线网址、研究目标、分析技术、TLS 1.3 版本支持对这些工作进行了汇总, 如表 9 所示.

从表 9 中可以看到, 在协议配置安全扫描工具中, SSL Labs 提供的 SSLtest 工具虽然拥有高效的扫描能力, 但是由于不开源, 因此只适合研究人员进行在线扫描. 而 SSLyze、testSSL、TLS-Scanner、TLS-Crawler 这 4 款开源工具可进行大规模扫描和统计分析, TLS-Scanner 扫描框架基于 Java 代码实现, 提供了较好的代码集成和二次开发能力, 适合研究人员针对 TLS 协议新漏洞进行扫描实现. 与上述远程动态扫描工具不同, DissecTLS 和 EFACTLS 使用了 TLS 指纹识别技术, 只需发送少量的请求就可以检测 TLS 协议配置, 因此检测效率更高.

在协议应用安全方面, 研究人员主要基于动静态程序分析和可验证安全技术来检测应用程序在使用 TLS 协议时存在的安全风险, 包括密码学 API 误用及证书验证相关缺陷.

总体而言, 协议配置扫描和自定义证书验证已成为当前 TLS 协议应用配置安全方面的研究重心, 亟需研究人员在检测技术上持续优化. DissecTLS 因其可扩展性与高效率, 已成为协议配置安全研究的主流选择. Marvin 首次系统性揭示了 TLS 在移动端应用中普遍存在的证书验证劫持问题, 凸显了应用层 TLS 实现风险的严峻性.

表 9　TLS 1.3 协议配置及应用安全相关研究对比

文献	发表时间	发表会议/期刊	开源工具	研究目标	分析技术	协议版本	
						1.2	1.3
[117]	2009	—	SSLtest	协议配置/漏洞/证书链	远程动态扫描	√	√
[118]	2014	—	SSLyze	协议配置/漏洞/证书链	远程动态扫描	√	√
[119]	2014	—	testSSL	协议配置/漏洞/证书链	远程动态扫描	√	√
[120]	2017	—	TLS-Scanner	协议配置/漏洞/证书链	远程动态扫描	√	√
[121]	2019	UsenixSEC	TLS-Crawler	协议配置 (CBC填充攻击)	远程动态扫描	√	√
[122]	2023	PAM	DissecTLS	协议配置	指纹识别	√	√
[123]	2024	TNSM	EFACTLS	协议配置	指纹识别	√	√
[124]	2023	UsenixSEC	NA	会话票据	人工分析/远程动态扫描	√	√
[125]	2024	ESORICS	NA	会话票据 (分区预言机攻击)	人工分析/远程动态扫描	√	√
[127]	2019	CCS	CryptoGuard	密码学API误用	人工分析/静态污点分析	√	×
[128]	2022	ACSAC	CryptoGo	密码学API误用	人工分析/静态污点分析	√	√
[129]	2022	S&P	MASC	密码学API误用	人工分析/静态污点分析	√	×
[130]	2024	NDSS	artifacts	密码学API误用	人工分析	√	√
[133]	2021	UsenixSEC	NA	自定义证书验证	人工分析/静态程序分析	√	√
[134]	2024	UsenixSEC	Marvin	自定义证书验证	动静态程序分析	√	√
[131]	2020	TDSC	NA	验证过程	人工分析	√	√
[132]	2023	CCS	NA	验证过程	可验证安全	√	√

注: —表示该文献仅为开源工具, 未发表在学术会议或期刊; NA表示该文献工具未提供开源工具

5.4 小　结

本节介绍了 TLS 1.3 协议配置及应用过程中的安全问题, 并对相关研究进行了梳理和分析. 在协议配置安全方面, 业界提出了一些扫描工具, 工业界的工具侧重于全面的配置扫描和漏洞能力, 因此更适用于大规模扫描和分析; 而学术界的工具则侧重于更低的测量成本和精细化的特定漏洞扫描, 因此更适合研究者进行方法创新研究. 在协议应用方面, 研究者针对 3 种应用场景进行研究, 包括应用程序中的密码学 API 误用和验证过程, 以及安卓系统中的自定义证书验证. 在 TLS 1.3 攻击方面, 研究者提出了针对 RSA、(EC)DSA 签名算法和 PSK 机制的攻击方式. 此外, 研究者在 TLS 1.3 性能优化和后量子安全方面也进行了一些前沿研究, 测试了不同后量子加密算法在 TLS 软件库中的实际性能表现, 为 TLS 1.3 后量子安全特性的发展奠定了基础.

6 总结与展望

本文系统梳理了 TLS 1.3 协议安全相关研究, 从协议机制安全、软件实现安全、应用配置安全这 3 个方面对现有研究进行了分析对比. 尽管 TLS 1.3 协议规范机制已经非常健全, 但在某些 0-RTT 机制使用场景下仍存在前向安全风险和重放攻击风险, 且 TLS 1.3 协议软件和应用程序使用过程中还存在很多实现缺陷和配置问题. 另外, 根据 SSLLabs 机构最新调查统计[4], 仍有 29.9% 的网站不支持 TLS 1.3 或存在低版本 TLS. 因此, TLS 1.3 协议仍然面临很大安全挑战, 未来针对 TLS 1.3 协议的攻击, 会转化为针对软件实现及配置应用方面的攻击, 以及协议跨层交互安全和基于量子计算机的攻击. 未来的研究方向可以重点关注以下几个方向.

(1) 0-RTT 安全

TLS 1.3 引入的 0-RTT 机制带来了潜在的安全风险, 如重放攻击和前向安全问题, 因为早期数据可以在不同的会话中被重复使用. 现有的研究虽然提出了一些安全解决方案, 但是实际性能离应用还有很大差距. 因此, 如何进一步降低 0-RTT 安全风险仍然是后续很重要的一个研究方向, 包括但不限于以下几个方面. 首先, 改进 0-RTT 重放保护机制, 研究更有效的方法来防止 0-RTT 数据的重放; 其次, 研究更有效的 0-RTT 前向安全保护机制, 确保应用程序传输 0-RTT 数据时能保证前向安全性且拥有可接受的性能损耗.

(2) TLS 软件缺陷检测

尽管 TLS 1.3 在设计上已经非常安全, 但在软件实现过程中仍然可能存在漏洞或错误配置. 这些缺陷可能源于代码质量、第三方软件库的使用不当或是对协议细节理解不足. 现有基于模糊测试的研究, 虽然已经在多个 TLS 库中发现了一些安全问题和 CVE 漏洞, 但是仍然存在一些不足, 如多类型漏洞的检测能力不足、测试效率较低、TLS 库覆盖面不够等. 为了进一步提高 TLS 软件安全性, 基于模糊测试的高效缺陷检测是未来关键的研究方向, 在状态空间覆盖深度、反馈机制、测试效率以及 TLS 库覆盖面等方面亟待突破.

(3) 证书验证缺陷检测

证书验证是 TLS 握手过程中至关重要的一步, 任何疏忽都可能导致中间人攻击或其他形式的身份冒充. 虽然 TLS 1.3 强化了一些验证步骤, 但在实践中仍需面对复杂的证书管理系统以及新兴威胁. 目前, 有一些研究针对 TLS 库及应用程序中的证书验证缺陷进行了研究, 提出了一些检测方法, 但仍然存在改进空间. 后续可以针对性地加强证书验证研究, 提高证书验证缺陷检测能力, 如覆盖证书状态 (撤销等) 的检测和实时监控与响应, 构建能够即时检测和应对证书滥用和撤销等行为的证书监测系统.

(4) 协议跨层交互安全

在跨层交互安全方面, TLS 协议安全高度依赖于底层 TCP/IP 协议栈, 但二者设计上的隔离构成了新的攻击面. 现有研究表明, 即使 TLS 1.3 在协议层提供了完善的加密保护, 攻击者仍可通过对底层协议的非路径攻击[135]间接破坏 TLS 会话的可用性与上下文完整性. 未来研究需重点关注 TLS over TCP 的连接劫持与可用性攻击. 如攻击者可基于 WiFi 网络加密帧长度[136]等侧信道信息推断 TCP 序列号, 进而实施精准的 TCP 连接重置或数据注入. 尽管 TLS 记录层协议会通过完整性校验丢弃被篡改的数据, 但这一过程将导致 TLS 连接性能下降甚至中断, 形成拒绝服务攻击. 此外, 攻击者还可利用 ICMP 错误消息[137]或 ICMP 重定向报文[138]误导主机路由或触发连接状态异常, 从而破坏 TLS 连接稳定性.

(5) 协议攻击及后量子安全

随着量子计算机的发展, 现有公钥密码体系面临日益严峻的安全挑战. 尽管 TLS 1.3 在协议设计时已为后量子密码的引入预留了扩展空间, 但目前全面迁移至具备抗量子攻击能力的方案仍面临诸多未解难题. 在协议攻击及后量子安全方面, 后续研究应从协议与算法两个维度共同推进, 重点关注以下方向. 首先, 深化协议层面攻击分析. 针对 0-RTT 机制 (包括 PSK) 和签名算法存在的安全风险, 进一步探索基于硬件故障注入或侧信道攻击手段对协议握手或加密过程的影响, 并设计相应防护机制. 其次, 优化混合加密协议设计. 当前主流策略是采用经典算法与后量子加密算法并行执行的混合模式, 以兼顾 TLS 1.3 协议安全性与向后兼容. 然而, 这种叠加式设计增加了握手延迟与带宽开销. 未来研究应探索更高效的组合方式, 如基于单轮次多算法密钥交换的紧凑型协议结构, 或利用密钥封装复用机制减少通信轮次. 再者, 推动标准化与互操作性实践, 积极参与 NIST 等国际标准组织的标准制定流程, 确保新算法的选择和实施符合全球共识, 推动后量子加密算法在主流 TLS 库中的规范化集成. 最后, 提升算法性能与协议运行效率. 后量子加密算法在计算或通信开销上显著高于传统算法, 对移动端与低功耗设备构成挑战. 研究应聚焦于轻量化实现, 包括算法层面的软件实现优化、硬件加速, 以及协议层面的密钥协商机制优化.

References

[1] W3techs. Default protocol https vs. QUIC usage statistics. 2024. https://w3techs.com/technologies/comparison/ce-httpsdefault, ce-quic

[2] Rescorla E. HTTP over TLS. RFC 2818, 2000. [doi: 10.17487/RFC2818]

[3] Iyengar J, Thomson M. QUIC: A UDP-based multiplexed and secure transport. RFC 9000, 2021. [doi: 10.17487/RFC9000]

[4] SSL Labs. SSL pulse. 2024. https://www.ssllabs.com/ssl-pulse/

[5] Dierks T, Rescorla E. The transport layer security (TLS) protocol version 1.2. RFC 5246, 2008. [doi: 10.17487/RFC5246]

[6] Rescorla E. The transport layer security (TLS) protocol version 1.3. RFC 8446, 2018. [doi: 10.17487/RFC8446]

[7] Fischlin M, Günther F. Replay attacks on zero round-trip time: The case of the TLS 1.3 handshake candidates. In: Proc. of the 2017 IEEE European Symp. on Security and Privacy. Paris: IEEE, 2017. 60–75. [doi: 10.1109/EuroSP.2017.18]

[8] Günther F, Hale B, Jager T, Lauer S. 0-RTT key exchange with full forward secrecy. In: Proc. of the 36th Annual Int'l Conf. on the

Theory and Applications of Cryptographic Techniques. Paris: Springer, 2017. 519–548. [doi: 10.1007/978-3-319-56617-7_18]

[9] OpenSSL. Vulnerabilities. 2024. https://www.openssl.org/news/vulnerabilities.html

[10] Ronen E, Gillham R, Genkin D, Shamir A, Wong D, Yarom Y. The 9 lives of Bleichenbacher's CAT: New cache attacks on TLS implementations. In: Proc. of the 2019 IEEE Symp. on Security and Privacy. San Francisco: IEEE, 2019. 435–452. [doi: 10.1109/SP.2019.00062]

[11] Alashwali ES, Rasmussen K. What's in a downgrade? A taxonomy of downgrade attacks in the TLS protocol and application protocols using TLS. In: Proc. of the 14th Int'l Conf. on Security and Privacy in Communication Networks. Singapore: Springer, 2018. 468–487. [doi: 10.1007/978-3-030-01704-0_27]

[12] Giesen F, Kohlar F, Stebila D. On the security of TLS renegotiation. In: Proc. of the 2013 ACM SIGSAC Conf. on Computer and Communications Security. Berlin: ACM, 2013. 387–398. [doi: 10.1145/2508859.2516694]

[13] Wei JL, Duan HX, Wan T. A survey of security deficiencies in design and implementation of HTTPS/TLS. Journal of Cyber Security, 2018, 3(2): 1–15 (in Chinese with English abstract). [doi: 10.19363/j.cnki.cn10-1380/tn.2018.03.01]

[14] Zhang XL, Cheng QF, Ma JF. Advance in TLS 1.3 protocol studies. Journal of Wuhan University (Natural Science Edition), 2018, 64(6): 471–484 (in Chinese with English abstract). [doi: 10.14188/j.1671-8836.2018.06.001]

[15] de Carné de Carnavalet X, van Oorschot PC. A survey and analysis of TLS interception mechanisms and motivations: Exploring how end-to-end TLS is made "end-to-me" for Web traffic. ACM Computing Surveys, 2023, 55(13s): 269. [doi: 10.1145/3580522]

[16] Daniele C, Andarzian SB, Poll E. Fuzzers for stateful systems: Survey and research directions. ACM Computing Surveys, 2024, 56(9): 222. [doi: 10.1145/3648468]

[17] Yu B, Su JS, Yang Q, Huang JX, Sheng ZS, Liu RH, Lu JJ, Liang C, Chen C, Zhao L. Survey on vulnerability mining techniques of network protocol software. Ruan Jian Xue Bao/Journal of Software, 2024, 35(2): 872–898 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/6942.htm [doi: 10.13328/j.cnki.jos.006942]

[18] Swierzy B, Boes F, Pohl T, Bungartz C, Meier M. SoK: Automated software testing for TLS libraries. In: Proc. of the 19th Int'l Conf. on Availability, Reliability and Security. Vienna: ACM, 2024. 54. [doi: 10.1145/3664476.3670871]

[19] Dierks T, Allen C. The TLS protocol version 1.0. RFC 2246, 1999. [doi: 10.17487/RFC2246]

[20] Dierks T, Rescorla E. The transport layer security (TLS) protocol version 1.1. RFC 4346, 2006. [doi: 10.17487/RFC4346]

[21] AlFardan NJ, Bernstein DJ, Paterson KG, Poettering B, Schuldt JCN. On the security of RC4 in TLS. In: Proc. of the 22nd USENIX Security Symp. Washington: USENIX Association, 2013. 305−320.

[22] AlFardan NJ, Paterson KG. Lucky thirteen: Breaking the TLS and DTLS record protocols. In: Proc. of the 2013 IEEE Symp. on Security and Privacy. Berkeley: IEEE, 2013. 526−540. [doi: 10.1109/SP.2013.42]

[23] Ronen E, Paterson KG, Shamir A. Pseudo constant time implementations of TLS are only pseudo secure. In: Proc. of the 2018 ACM SIGSAC Conf. on Computer and Communications Security. Toronto: ACM, 2018. 1397–1414. [doi: 10.1145/3243734.3243775]

[24] Möller B, Duong T, Kotowicz K. This POODLE bites: Exploiting the SSL 3.0 fallback. 2014. http://cdn1.vox-cdn.com/uploads/chorus_asset/file/2354994/ssl-poodle.0.pdf

[25] Adrian D, Bhargavan K, Durumeric Z, Gaudry P, Green M, Halderman JA, Heninger N, Springall D, Thomé E, Valenta L, VanderSloot B, Wustrow E, Zanella-Béguelink S, Zimmermann P. Imperfect forward secrecy: How Diffie-Hellman fails in practice. In: Proc. of the 22nd ACM SIGSAC Conf. on Computer and Communications Security. Denver: ACM, 2015. 5−17. [doi: 10.1145/2810103.2813707]

[26] Bhargavan K, Lavaud AD, Fournet C, Pironti A, Strub PY. Triple handshakes and cookie cutters: Breaking and fixing authentication over TLS. In: Proc. of the 2014 IEEE Symp. on Security and Privacy. Berkeley: IEEE, 2014. 98−113. [doi: 10.1109/SP.2014.14]

[27] Merget R, Brinkmann M, Aviram N, Somorovsky J, Mittmann J, Schwenk J. Raccoon attack: Finding and exploiting most-significant-bit-oracles in TLS-DH(E). In: Proc. of the 30th USENIX Security Symp. USENIX Association, 2021. 213–230.

[28] Aviram N, Schinzel S, Somorovsky J, Heninger N, Dankel M, Steube J, Valenta L, Adrian D, Halderman JA, Dukhovni V, Käsper E, Cohney S, Engels S, Paar C, Shavitt Y. DROWN: Breaking TLS using SSLv2. In: Proc. of the 25th USENIX Security Symp. Austin: USENIX Association, 2016. 689–706.

[29] Brinkmann M, Dresen C, Merget R, Poddebniak D, Müller J, Somorovsky J, Schwenk J, Schinzel S. ALPACA: Application layer protocol confusion-analyzing and mitigating cracks in TLS authentication. In: Proc. of the 30th USENIX Security Symp. USENIX Association, 2021. 4293–4310.

[30] Synopsys Inc. The Heartbleed bug. 2014. https://heartbleed.com/

[31] Beurdouche B, Bhargavan K, Delignat-Lavaud A, Fournet C, Kohlweiss M, Pironti A, Strub PY, Zinzindohoue JK. A messy state of the union: Taming the composite state machines of TLS. In: Proc. of the 2015 IEEE Symp. on Security and Privacy. San Jose: IEEE, 2015. 535−552. [doi: 10.1109/SP.2015.39]

[32] Canetti R, Krawczyk H. Universally composable notions of key exchange and secure channels. In: Proc. of the 2002 Int'l Conf. on the Theory and Applications of Cryptographic Techniques. Amsterdam: Springer, 2002. 337–351. [doi: 10.1007/3-540-46035-7_22]

[33] Cremers C, Feltz M. Beyond eCK: Perfect forward secrecy under actor compromise and ephemeral-key reveal. In: Proc. of the 17th European Symp. on Research in Computer Security. Pisa: Springer, 2012. 734–751. [doi: 10.1007/978-3-642-33167-1_42]

[34] Dolev D, Even S, Karp RM. On the security of ping-pong protocols. Information and Control, 1982, 55(1–3): 57–68. [doi: 10.1016/S0019-9958(82)90401-6]

[35] Armando A, Basin D, Boichut Y, Chevalier Y, Compagna L, Cuellar J, Drielsma PH, Heám PC, Kouchnarenko O, Mantovani J, Mödersheim S, von Oheimb D, Rusinowitch M, Santiago J, Turuani M, Viganò L, Vigneron L. The AVISPA tool for the automated validation of Internet security protocols and applications. In: Proc. of the 17th Int'l Conf. on Computer Aided Verification. Edinburgh: Springer, 2005. 281–285. [doi: 10.1007/11513988_27]

[36] Blanchet B. An efficient cryptographic protocol verifier based on prolog rules. In: Proc. of the 14th IEEE Computer Security Foundations Workshop. Cape Breton: IEEE, 2001. 82–96. [doi: 10.1109/CSFW.2001.930138]

[37] Escobar S, Meadows C, Meseguer J. Maude-NPA: Cryptographic protocol analysis modulo equational properties. In: Foundations of Security Analysis and Design V. Berlin: Springer, 2007. 1–50. [doi: 10.1007/978-3-642-03829-7_1]

[38] Cremers CJF. The scyther tool: Verification, falsification, and analysis of security protocols: Tool paper. In: Proc. of the 20th Int'l Conf. on Computer Aided Verification. Princeton: Springer, 2008. 414–418. [doi: 10.1007/978-3-540-70545-1_38]

[39] Meier S, Schmidt B, Cremers C, Basin D. The Tamarin prover for the symbolic analysis of security protocols. In: Proc. of the 25th Int'l Conf. on Computer Aided Verification. Saint Petersburg: Springer, 2013. 696–701. [doi: 10.1007/978-3-642-39799-8_48]

[40] Blanchet B. CryptoVerif: Computationally sound mechanized prover for cryptographic protocols. 2007. https://bblanche.gitlabpages.inria.fr/talks/Dagstuhl07.pdf

[41] Bellare M, Rogaway P. Entity authentication and key distribution. In: Proc. of the 13th Annual Int'l Cryptology Conf. Santa Barbara: Springer, 1993. 232–249. [doi: 10.1007/3-540-48329-2_21]

[42] LaMacchia B, Lauter K, Mityagin A. Stronger security of authenticated key exchange. In: Proc. of the 1st Int'l Conf. on Provable Security. Wollongong: Springer, 2007. 1–16. [doi: 10.1007/978-3-540-75670-5_1]

[43] 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. Scottsdale: ACM, 2014. 1193–1204. [doi: 10.1145/2660267.2660308]

[44] Jager T, Kohlar F, Schäge S, Schwenk J. On the security of TLS-DHE in the standard model. In: Proc. of the 32nd Annual Int'l Cryptology Conf. Santa Barbara: Springer, 2012. 273–293. [doi: 10.1007/978-3-642-32009-5_17]

[45] Dowling B, Fischlin M, Günther F, Stebila D. A cryptographic analysis of the TLS 1.3 handshake protocol candidates. In: Proc. of the 22nd ACM SIGSAC Conf. on Computer and Communications Security. Denver: ACM, 2015. 1197–1210. [doi: 10.1145/2810103.2813653]

[46] 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 22nd ACM SIGSAC Conf. on Computer and Communications Security. Denver: ACM, 2015. 1185–1196. [doi: 10.1145/2810103.2813657]

[47] Li XY, Xu J, Zhang ZF, Feng DG, Hu HG. Multiple handshakes security of TLS 1.3 candidates. In: Proc. of the 2016 IEEE Symp. on Security and Privacy. San Jose: IEEE, 2016. 486–505. [doi: 10.1109/SP.2016.36]

[48] Fischlin M, Günther F, Schmidt B, Warinschi B. Key confirmation in key exchange: A formal treatment and implications for TLS 1.3. In: Proc. of the 2016 IEEE Symp. on Security and Privacy. San Jose: IEEE, 2016. 452–469. [doi: 10.1109/SP.2016.34]

[49] Dowling B, Fischlin M, Günther F, Stebila D. A cryptographic analysis of the TLS 1.3 draft-10 full and pre-shared key handshake protocol. 2016. https://eprint.iacr.org/2016/081

[50] Krawczyk H. A unilateral-to-mutual authentication compiler for key exchange (with applications to client authentication in TLS 1.3). In: Proc. of the 2016 ACM SIGSAC Conf. on Computer and Communications Security. Vienna: ACM, 2016. 1438–1450. [doi: 10.1145/2976749.2978325]

[51] Lan X, Xu J, Zhang ZF, Zhu WT. Investigating the multi-ciphersuite and backwards-compatibility security of the upcoming TLS 1.3. IEEE Trans. on Dependable and Secure Computing, 2019, 16(2): 272–286. [doi: 10.1109/TDSC.2017.2685382]

[52] 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. San Jose: IEEE, 2017. 483–502. [doi: 10.1109/SP.2017.26]

[53] Chen S, Jero S, Jagielski M, Boldyreva A, Nita-Rotaru C. Secure communication channel establishment: TLS 1.3 (over TCP fast open) vs. QUIC. In: Proc. of the 24th European Symp. on Research in Computer Security. Luxembourg: Springer, 2019. 404–426. [doi: 10.1007/978-3-030-29959-0_20]

[54] Lychev R, Jero S, Boldyreva A, Nita-Rotaru C. How secure and quick is QUIC? Provable security and performance analyses. In: Proc. of the 2015 IEEE Symp. on Security and Privacy. San Jose: IEEE, 2015. 214–231. [doi: 10.1109/SP.2015.21]

[55] Dowling B, Fischlin M, Günther F, Stebila D. A cryptographic analysis of the TLS 1.3 handshake protocol. Journal of Cryptology, 2021, 34(4): 37. [doi: 10.1007/s00145-021-09384-1]

[56] Diemert D, Jager T. On the tight security of TLS 1.3: Theoretically sound cryptographic parameters for real-world deployments. Journal of Cryptology, 2021, 34(3): 30. [doi: 10.1007/s00145-021-09388-x]

[57] Bhargavan K, Cheval V, Wood C. A symbolic analysis of privacy for TLS 1.3 with encrypted client hello. In: Proc. of the 2022 ACM SIGSAC Conf. on Computer and Communications Security. Los Angeles: ACM, 2022. 365–379. [doi: 10.1145/3548606.3559360]

[58] Brzuska C, Delignat-Lavaud A, Egger C, Fournet C, Kohbrok K, Kohlweiss M. Key-schedule security for the TLS 1.3 standard. In: Proc. of the 28th Int'l Conf. on the Theory and Application of Cryptology and Information Security. Taipei: Springer, 2022. 621–650. [doi: 10.1007/978-3-031-22963-3_21]

[59] Brzuska C, Delignat-Lavaud A, Fournet C, Kohbrok K, Kohlweiss M. State separation for code-based game-playing proofs. In: Proc. of the 24th Int'l Conf. on the Theory and Application of Cryptology and Information Security. Brisbane: Springer, 2018. 222–249. [doi: 10.1007/978-3-030-03332-3_9]

[60] Fischlin M. Stealth Key exchange and confined access to the record protocol data in TLS 1.3. In: Proc. of the 2023 ACM SIGSAC Conf. on Computer and Communications Security. Copenhagen: ACM, 2023. 2901–2914. [doi: 10.1145/3576915.3623099]

[61] Krawczyk H, Wee H. The OPTLS protocol and TLS 1.3. In: Proc. of the 2016 IEEE European Symp. on Security and Privacy. Saarbruecken: IEEE, 2016. 81–96. [doi: 10.1109/EuroSP.2016.18]

[62] Cremers C, Horvat M, Scott S, van der Merwe T. Automated analysis and verification of TLS 1.3: 0-RTT, resumption and delayed authentication. In: Proc. of the 2016 IEEE Symp. on Security and Privacy. San Jose: IEEE, 2016. 470–485. [doi: 10.1109/SP.2016.35]

[63] Cremers C, Horvat M, Hoyland J, Scott S, Van Der Merwe T. A comprehensive symbolic analysis of TLS 1.3. In: Proc. of the 2017 ACM SIGSAC Conf. on Computer and Communications Security. Dallas: ACM, 2017. 1773–1788. [doi: 10.1145/3133956.3134063]

[64] Drucker N, Gueron S. Selfie: Reflections on TLS 1.3 with PSK. Journal of Cryptology, 2021, 34(3): 27. [doi: 10.1007/s00145-021-09387-y]

[65] 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 with English abstract). http://www.jos.org.cn/1000-9825/5973.htm [doi: 10.13328/j.cnki.jos.005973]

[66] Davis H, Diemert D, Günther F, Jager t. On the concrete security of TLS 1.3 PSK mode. In: Proc. of the 41st Annual Int'l Conf. on the Theory and Applications of Cryptographic Techniques. Trondheim: Springer, 2022. 876–906. [doi: 10.1007/978-3-031-07085-3_30]

[67] Zhang XL, Cheng QF, Ma JF. Protocol to enhance the security of Early data in TLS 1.3. Chinese Journal of Network and Information Security, 2017, 3(12): 8–16 (in Chinese with English abstract). [doi: 10.11959/j.issn.2096-109x.2017.00224]

[68] Derler D, Jager T, Slamanig D, Striecks C. Bloom filter encryption and applications to efficient forward-secret 0-RTT key exchange. In: Proc. of the 37th Annual Int'l Conf. on the Theory and Applications of Cryptographic Techniques. Tel Aviv: Springer, 2018. 425–455. [doi: 10.1007/978-3-319-78372-7_14]

[69] Aviram N, Gellert K, Jager T. Session resumption protocols and efficient forward security for TLS 1.3 0-RTT. In: Proc. of the 38th Annual Int'l Conf. on the Theory and Applications of Cryptographic Techniques. Darmstadt: Springer, 2019. 117–150. [doi: 10.1007/978-3-030-17656-3_5]

[70] Göth C, Ramacher S, Slamanig D, Striecks C, Tairi E, Zikulnig A. Optimizing 0-RTT key exchange with full forward security. In: Proc. of the 2023 Cloud Computing Security Workshop. Copenhagen: ACM, 2023. 55–68. [doi: 10.1145/3605763.3625246]

[71] Dallmeier F, Drees JP, Gellert K, Handirk T, Jager T, Klauke J, Nachtigall S, Renzelmann T, Wolf R. Forward-secure 0-RTT goes live: Implementation and performance analysis in QUIC. In: Proc. of the 19th Int'l Conf. on Cryptology and Network Security. Vienna: Springer, 2020. 211–231. [doi: 10.1007/978-3-030-65411-5_11]

[72] Levillain O, Gourdin B, Debar H. TLS record protocol: Security analysis and defense-in-depth countermeasures for HTTPS. In: Proc. of the 10th ACM Symp. on Information, Computer and Communications Security. Singapore: ACM, 2015. 225–236. [doi: 10.1145/2714576.2714592]

[73] Duong T, Rizzo J. Here come the ⊕ ninjas. 2011. https://nerdoholic.org/uploads/dergln/beast_part2/ssl_jun21.pdf

[74] Schwenk J. Guide to Internet Cryptography: Security Protocols and Real-world Attack Implications. Cham: Springer, 2022. 267–328. [doi: 10.1007/978-3-031-19439-9]

[75] Prado A, Harris N, Gluck Y. SSL, gone in 30 seconds—A BREACH beyond CRIME. 2013. https://media.blackhat.com/us-13/US-13-Prado-SSL-Gone-in-30-seconds-A-BREACH-beyond-CRIME-Slides.pdf

[76] Be'ery T. A perfect CRIME? TIME will tell. 2013. https://media.blackhat.com/eu-13/briefings/Beery/bh-eu-13-a-perfect-crime-beery-slides.pdf

[77] Badertscher C, Matt C, Maurer U, Rogaway P, Tackmann B. Augmented secure channels and the goal of the TLS 1.3 record layer. In: Proc. of the 9th Int'l Conf. on Provable Security. Kanazawa: Springer, 2015. 85–104. [doi: 10.1007/978-3-319-26059-4_5]

[78] Bellare M, Tackmann B. The multi-user security of authenticated encryption: AES-GCM in TLS 1.3. In: Proc. of the 36th Annual Int'l Cryptology Conf. Santa Barbara: ACM, 2016. 247–276. [doi: 10.1007/978-3-662-53018-4_10]

[79] Delignat-Lavaud A, Fournet C, Kohlweiss M, Protzenko J, Rastogi A, Swamy N, Zanella-Beguelin S, Bhargavan K, Pan JY, Zinzindohoue JK. Implementing and proving the TLS 1.3 record layer. In: Proc. of the 2017 IEEE Symp. on Security and Privacy. San Jose: IEEE, 2017. 463–482. [doi: 10.1109/SP.2017.58]

[80] Patton C, Shrimpton T. Partially specified channels: The TLS 1.3 record layer without elision. In: Proc. of the 2018 ACM SIGSAC Conf. on Computer and Communications Security. Toronto: ACM, 2018. 1415–1428. [doi: 10.1145/3243734.3243789]

[81] Stebila D, Mosca M. Post-quantum key exchange for the Internet and the open quantum safe project. In: Proc. of the 23rd Int'l Conf. on Selected Areas in Cryptography. St. John's: Springer, 2017. 14–37. [doi: 10.1007/978-3-319-69453-5_2]

[82] Schwabe P, Stebila D, Wiggers T. Post-quantum TLS without handshake signatures. In: Proc. of the 2020 ACM SIGSAC Conf. on Computer and Communications Security. ACM, 2020. 1461–1480. [doi: 10.1145/3372297.3423350]

[83] Sikeridis D, Kampanakis P, Devetsikiotis M. Post-quantum authentication in TLS 1.3: A performance study. In: Proc. of the 2020 Network and Distributed System Security Symp. San Diego: Internet Society, 2020. [doi: 10.14722/ndss.2020.24203]

[84] Bernstein DJ, Brumley BB, Chen MS, Tuveri N. OpenSSLNTRU: Faster post-quantum TLS key exchange. In: Proc. of the 31st USENIX Security Symp. Boston: USENIX Association, 2022. 845–862.

[85] Garcia CR, Aguilera AC, Olmos JJV, Monroy IT, Rommel S. Quantum-resistant TLS 1.3: A hybrid solution combining classical, quantum and post-quantum cryptography. In: Proc. of the 28th IEEE Int'l Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD). Edinburgh: IEEE, 2023. 246–251. [doi: 10.1109/CAMAD59638.2023.10478407]

[86] Campos F, Chávez-Saab J, Chi-Domínguez JJ, Meyer M, Reijnders K, Rodríguez-Henríquez F, Schwabe P, Wiggers T. Optimizations and practicality of high-security CSIDH. IACR Communications in Cryptology, 2024, 1(1): 1–26. [doi: 10.62056/anjbksdja]

[87] Chen JR, Zhou BM, Chen RM, Jiang HD, Wang Y, Huang XY, Zhao YL, Yung MT. Post-quantum TLS 1.3 handshake from CPA-secure KEMs with tighter reductions. 2025. https://eprint.iacr.org/2025/1748

[88] Rahaman S, Cai HP, Chowdhury O, Yao DF. From theory to code: Identifying logical flaws in cryptographic implementations in C/C++. IEEE Trans. on Dependable and Secure Computing, 2022, 19(6): 3790–3803. [doi: 10.1109/TDSC.2021.3108031]

[89] Sivakorn S, Argyros G, Pei KX, Keromytis AD, Jana S. HVLearn: Automated black-box analysis of hostname verification in SSL/TLS implementations. In: Proc. of the 2017 IEEE Symp. on Security and Privacy. San Jose: IEEE, 2017. 521–538. [doi: 10.1109/SP.2017.46]

[90] Chen C, Tian C, Duan ZH, Zhao L. RFC-directed differential testing of certificate validation in SSL/TLS implementations. In: Proc. of the 40th Int'l Conf. on Software Engineering. Gothenburg: ACM, 2018. 859–870. [doi: 10.1145/3180155.3180226]

[91] Tian C, Chen C, Duan ZH, Zhao L. Differential testing of certificate validation in SSL/TLS implementations: An RFC-guided approach. ACM Trans. on Software Engineering and Methodology, 2019, 28(4): 24. [doi: 10.1145/3355048]

[92] Nie PB, Wan CC, Zhu JY, Lin ZY, Chen YT, Su ZD. Coverage-directed differential testing of X.509 certificate validation in SSL/TLS implementations. ACM Trans. on Software Engineering and Methodology, 2023, 32(1): 3. [doi: 10.1145/3510416]

[93] Chen C, Ren PH, Duan ZH, Tian C, Lu X, Yu B. SBDT: Search-based differential testing of certificate parsers in SSL/TLS implementations. In: Proc. of the 32nd ACM SIGSOFT Int'l Symp. on Software Testing and Analysis. Seattle: ACM, 2023. 967–979. [doi: 10.1145/3597926.3598110]

[94] Hawkes B. Project zero five years of 'make 0day hard'. 2020. https://i.blackhat.com/USA-19/Thursday/us-19-Hawkes-Project-Zero-Five-Years-Of-Make-0day-Hard.pdf

[95] Luo ZX, Yu JZ, Zuo FL, Liu JZ, Jiang Y, Chen T, Roychoudhury A, Sun JG. Bleem: Packet sequence oriented fuzzing for protocol implementations. In: Proc. of the 32nd USENIX Security Symp. Anaheim: USENIX Association, 2023. 4481–4498.

[96] De Ruiter J, Poll E. Protocol state fuzzing of TLS implementations. In: Proc. of the 24th USENIX Security Symp. Washington: USENIX Association, 2015. 193–206.

[97] Somorovsky J. Systematic fuzzing and testing of TLS libraries. In: Proc. of the 2016 ACM SIGSAC Conf. on Computer and Communications Security. Vienna: ACM, 2016. 1492–1504. [doi: 10.1145/2976749.2978411]

[98] Rescorla E, Tschofenig H, Modadugu N. The datagram transport layer security (DTLS) protocol version 1.3. RFC 9147, 2022. [doi: 10.17487/RFC9147]

[99] Fiterău-Broştean P, Jonsson B, Merget R, de Ruiter J, Sagonas K, Somorovsky J. Analysis of DTLS implementations using protocol

state fuzzing. In: Proc. of the 29th USENIX Security Symp. USENIX Association, 2020. 2523–2540.

[100] DTLS-Fuzzer. 2020. https://github.com/assist-project/dtls-fuzzer/

[101] Ba JS, Böhme M, Mirzamomen Z, Roychoudhury A. Stateful greybox fuzzing. In: Proc. of the 31st USENIX Security Symp. Boston: USENIX Association, 2022. 3255–3272.

[102] Ammann M, Hirschi L, Kremer S. DY fuzzing: Formal Dolev-Yao models meet cryptographic protocol fuzz testing. In: Proc. of the 2024 IEEE Symp. on Security and Privacy. San Francisco: IEEE, 2024. 1481–1499. [doi: 10.1109/SP54263.2024.00096]

[103] Michał Z. American fuzzy lop (AFL). 2013. https://lcamtuf.coredump.cx/afl/

[104] libFuzzer. libFuzzer—A library for coverage-guided fuzz testing. 2017. https://llvm.org/docs/LibFuzzer.html

[105] Walz A, Sikora A. Exploiting dissent: Towards fuzzing-based differential black-box testing of TLS implementations. IEEE Trans. on Dependable and Secure Computing, 2020, 17(2): 278–291. [doi: 10.1109/TDSC.2017.2763947]

[106] Petsios T, Tang A, Stolfo S, Keromytis AD, Jana S. NEZHA: Efficient domain-independent differential testing. In: Proc. of the 2017 IEEE Symp. on Security and Privacy. San Jose: IEEE, 2017. 615–632. [doi: 10.1109/SP.2017.27]

[107] Kario H. Testing TLS. 2015. https://github.com/tlsfuzzer/tlsfuzzer

[108] Moneger A. Penetration testing custom TLS stacks. 2016. https://github.com/tintinweb/scapy-ssl_tls

[109] Maehren M, Nieting P, Hebrok S, Merget R, Somorovsky J, Schwenk J. TLS-Anvil: Adapting combinatorial testing for TLS libraries. In: Proc. of the 31st USENIX Security Symp. Boston: USENIX Association, 2022. 215–232.

[110] Zhao Z, Song XP, Zhong QY, Zeng YP, Hu CY, Guo SQ. TLS-DeepDiffer: Message tuples-based deep differential fuzzing for TLS protocol implementations. In: Proc. of the 2024 IEEE Int'l Conf. on Software Analysis, Evolution and Reengineering. Rovaniemi: IEEE, 2024. 918–928. [doi: 10.1109/SANER60148.2024.00100]

[111] Mus K, Doröz Y, Tol MC, Rahman K, Sunar B. Jolt: Recovering TLS signing keys via rowhammer faults. In: Proc. of the 2023 IEEE Symp. on Security and Privacy. San Francisco: IEEE, 2023. 1719–1736. [doi: 10.1109/SP46215.2023.10179450]

[112] Sullivan GA, Sippe J, Heninger N, Wustrow E. Open to a fault: On the passive compromise of TLS keys via transient errors. In: Proc. of the 31st USENIX Security Symp. Boston: USENIX Association, 2022. 233–250.

[113] Xie RJ, Cao JH, Dong EH, Xu MW, Sun K, Li Q, Shen LC, Zhang MH. Rosetta: Enabling robust TLS encrypted traffic classification in diverse network environments with TCP-aware traffic augmentation. In: Proc. of the 32nd USENIX Conf. on Security Symp. Anaheim: USENIX Association, 2023. 625–642.

[114] Niere N, Hebrok S, Somorovsky J, Merget R. Poster: Circumventing the GFW with TLS record fragmentation. In: Proc. of the 2023 ACM SIGSAC Conf. on Computer and Communications Security. Copenhagen: ACM, 2023. 3528–3530. [doi: 10.1145/3576915.3624372]

[115] Xue DW, Kallitsis M, Houmansadr A, Ensafi R. Fingerprinting obfuscated proxy traffic with encapsulated TLS handshakes. In: Proc. of the 33rd USENIX Security Symp. Philadelphia: USENIX Association, 2024. 2689–2706.

[116] Zhang JP, Huang JH, Zhao LR, Chen DL, Koç CK. ENG25519: Faster TLS 1.3 handshake using optimized X25519 and Ed25519. In: Proc. of the 33rd USENIX Security Symp. Philadelphia: USENIX Association, 2024. 6381–6398.

[117] SSL Labs. SSL server test. 2009. https://www.ssllabs.com/ssltest/

[118] Diquet A. SSLyze. 2014. https://github.com/nabla-c0d3/sslyze

[119] Wetter D. Testing TLS/SSL encryption. 2014. https://testssl.sh/

[120] TLS-Scanner. 2017. https://github.com/tls-attacker/TLS-Scanner

[121] Merget R, Somorovsky J, Aviram N, Young C, Fliegenschmidt J, Schwenk J, Shavitt Y. Scalable scanning and automatic classification of TLS padding oracle vulnerabilities. In: Proc. of the 28th USENIX Security Symp. Santa Clara: USENIX Association, 2019. 1029–1046.

[122] Sosnowski M, Zirngibl J, Sattler P, Carle G. DissecTLS: A scalable active scanner for TLS server configurations, capabilities, and TLS fingerprinting. In: Proc. of the 24th Int'l Conf. on Passive and Active Measurement. Springer, 2023. 110–126. [doi: 10.1007/978-3-031-28486-1_6]

[123] Sosnowski M, Zirngibl J, Sattler P, Carle G, Grohnfeldt C, Russo M, Sgandurra D. EFACTLS: Effective active TLS fingerprinting for large-scale server deployment characterization. IEEE Trans. on Network and Service Management, 2024, 21(3): 2582–2595. [doi: 10.1109/TNSM.2024.3364526]

[124] Hebrok S, Nachtigall S, Maehren M, Erinola N, Merget R, Somorovsky J, Schwenk J. We really need to talk about session tickets: A large-scale analysis of cryptographic dangers with TLS session tickets. In: Proc. of the 32nd USENIX Security Symp. Anaheim: USENIX Association, 2023. 4877–4894.

[125] Radoy M, Hebrok S, Somorovsky J. In search of partitioning oracle attacks against TLS session tickets. In: Proc. of the 29th European Symp. on Research in Computer Security. Bydgoszcz: Springer, 2024. 320–340. [doi: 10.1007/978-3-031-70896-1_16]

[126] Len J, Grubbs P, Ristenpart T. Partitioning oracle attacks. In: Proc. of the 30th USENIX Security Symp. USENIX Association, 2021. 195–212.

[127] Rahaman S, Xiao Y, Afrose S, Shaon F, Tian K, Frantz M, Kantarcioglu M, Yao DF. CryptoGuard: High precision detection of cryptographic vulnerabilities in massive-sized Java projects. In: Proc. of the 2019 ACM SIGSAC Conf. on Computer and Communications Security. London: ACM, 2019. 2455–2472. [doi: 10.1145/3319535.3345659]

[128] Li WQ, Jia SJ, Liu LM, Zheng FY, Ma Y, Lin JQ. CryptoGo: Automatic detection of Go cryptographic API misuses. In: Proc. of the 38th Annual Computer Security Applications Conf. Austin: ACM, 2022. 318–331. [doi: 10.1145/3564625.3567989]

[129] Ami AS, Cooper N, Kafle K, Moran K, Poshyvanyk D, Nadkarni A. Why crypto-detectors fail: A systematic evaluation of cryptographic misuse detection techniques. In: Proc. of the 2022 IEEE Symp. on Security and Privacy. San Francisco: IEEE, 2022. 614–631. [doi: 10.1109/SP46214.2022.9833582]

[130] Chen YK, Liu YB, Wu KL, Le DV, Chau SY. Towards precise reporting of cryptographic misuses. In: Proc. of the Network and Distributed System Security Symp. San Diego: Internet Society, 2024. [doi: 10.14722/ndss.2024.241032]

[131] Wazan AS, Laborde R, Chadwick DW, Venant R, Benzekri A, Billoir E, Alfandi O. On the validation of Web X.509 certificates by TLS interception products. IEEE Trans. on Dependable and Secure Computing, 2022, 19(1): 227–242. [doi: 10.1109/TDSC.2020.3000595]

[132] Fischlin M, Günther F. Verifiable verification in cryptographic protocols. In: Proc. of the 2023 ACM SIGSAC Conf. on Computer and Communications Security. Copenhagen: ACM, 2023. 3239–3253. [doi: 10.1145/3576915.3623151]

[133] Oltrogge M, Huaman N, Amft S, Acar Y, Backes M, Fahl S. Why eve and mallory still love Android: Revisiting TLS (In)Security in Android applications. In: Proc. of the 30th USENIX Security Symp. USENIX Association, 2021. 4347–4364.

[134] Pourali S, Yu XF, Zhao LY, Mannan M, Youssef A. Racing for TLS certificate validation: A hijacker's guide to the Android TLS galaxy. In: Proc. of the 33rd USENIX Security Symp. Philadelphia: USENIX Association, 2024. 683–700.

[135] Feng XW, Li Q, Sun K, Xu K, Wu JP. Exploiting cross-layer vulnerabilities: Off-path attacks on the TCP/IP protocol suite. Communications of the ACM, 2025, 68(3): 48–59. [doi: 10.1145/3689819]

[136] Wang ZQ, Feng XW, Li Q, Sun K, Yang YX, Li MY, Du GQ, Xu K, Wu JP. Off-path TCP hijacking in Wi-Fi networks: A packet-size side channel attack. In: Proc. of the 2025 Network and Distributed System Security Symp. San Diego: Internet Society, 2025. [doi: 10.14722/ndss.2025.230305]

[137] Feng XW, Fu CP, Li Q, Sun K, Xu K. Off-path TCP exploits of the mixed IPID assignment. In: Proc. of the 2020 ACM SIGSAC Conf. on Computer and Communications Security. ACM, 2020. 1323–1335. [doi: 10.1145/3372297.3417884]

[138] Feng XW, Li Q, Sun K, Qian ZY, Zhao G, Kuang XH, Fu CP, Xu K. Off-path network traffic manipulation via revitalized ICMP redirect attacks. In: Proc. of the 31st USENIX Security Symp. Boston: USENIX Association, 2022. 2619–2636.

附中文参考文献

[13] 韦俊琳, 段海新, 万涛. HTTPS/TLS 协议设计和实现中的安全缺陷综述. 信息安全学报, 2018, 3(2): 1–15. [doi: 10.19363/j.cnki.cn10-1380/tn.2018.03.01]

[14] 张兴隆, 程庆丰, 马建峰. TLS 1.3 协议研究进展. 武汉大学学报 (理学版), 2018, 64(6): 471–484. [doi: 10.14188/j.1671-8836.2018.06.001]

[17] 喻波, 苏金树, 杨强, 黄见欣, 盛周石, 刘润昊, 卢建君, 梁晨, 陈晨, 赵磊. 网络协议软件漏洞挖掘技术综述. 软件学报, 2024, 35(2): 872–898. http://www.jos.org.cn/1000-9825/6942.htm [doi: 10.13328/j.cnki.jos.006942]

[65] 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 [doi: 10.13328/j.cnki.jos.005973]

[67] 张兴隆, 程庆丰, 马建峰. 增强 TLS 1.3 中 Early data 安全性的协议. 网络与信息安全学报, 2017, 3(12): 8–16. [doi: 10.11959/j.issn.2096-109x.2017.00224]

作者简介

柏军洋, 博士, 工程师, 主要研究领域为网络安全, 应用安全, 传输安全.

张宇, 博士, 教授, 博士生导师, CCF 高级会员, 主要研究领域为互联网基础设施安全, 网络拓扑测量, 未来网络体系.

张宾, 博士, 高级工程师, 博士生导师, 主要研究领域为网络安全, 网络管理, 数据分析.

张伟哲, 博士, 教授, 博士生导师, CCF 杰出会员, 主要研究领域为信息安全, 系统结构.