TLS1.3 协议更新发展及其攻击与防御研究

沈若愚[1] 卢盛祺[2] 赵运磊[1]

[1](复旦大学软件学院 上海 201203)

[2](上海财经大学信息管理与工程学院 上海 200433)

摘 要 SSL/TLS(Secure Sockets Layer /Transport Layer Security)协议旨在为网络通信提供安全的信道,为通信双方提供认证、机密性和完整性。由于协议的复杂及其设计和实现上的漏洞导致许多安全隐患,新版本 TLS 1.3的制定引起信息安全学术界和产业界广泛的关注。概述 TLS1.3 的协议结构。在此基础上,对 TLS1.3 几个革新性的改变:密钥编排表、PSK 和 0-RTT 进行了系统性地分析与梳理。对近 10 年协议受到的攻击按照协议的层次分类进行概述,提炼出每种攻击的原理以及 TLS1.3 针对这些攻击作出的应对措施。对 TLS 协议的未来发展作出预测并提出建议。

关键词 TLS1.3 SSL/TLS 攻击 0-RTT PSK 密钥生成表

中图分类号 TP3 **文献标识码** A DOI:10.3969/j.issn.1000-386x.2017.11.049

THE DEVELOPMENTS OF TLS1.3 AND ITS ATTACK AND DEFENSE

Shen Ruoyu[1] Lu Shengqi[2] Zhao Yunlei[1]

[1](*School of Software, Fudan University, Shanghai 201203, China*)

[2](*School of Information Management and Engineering, Shanghai University of Finace and Economics, Shanghai 200433, China*)

Abstract Secure Sockets Layer/Transport Layer Security (SSL/TLS) is intended to provide a secure channel for network communications, providing authentication, confidentiality and integrity. Due to the complexity and loopholes in the design and implementation of protocol leading to many security risks, the development of the new version of TLS1.3 caused widespread concern in the information security academia and industry. We outlined the protocol structure of TLS1.3. On this basis, several innovative changes of TLS1.3 were systematically analysed and combed, such as key schedule, PSK and 0-RTT. We reviewed the attacks received by the protocols for the last 10 years, and extracted the principle of each attack and TLS1.3 response to these attacks. And we made some predictions about the future development of TLS and make some recommendations.

Keywords TLS1.3 SSL/TLS attack 0-RTT PSK Key schedule

0 引 言

SSL/TLS 协议是应用范围非常广泛的密码协议之一,其主要的目的是为网络中通信的双方建立一个安全的通信信道。而这个信道需要为通信双方提供认证、机密性和完整性。该协议也是实际应用中的最复杂的密码协议之一,拥有高度的复杂性,版本众多,扩展、变体、工作模式、参数算法协商复杂多变。

由于 SSL/TLS 本身的复杂性,版本更新的缓慢以及实现维护过程的不重视,协议的设计与实现过程存在许多漏洞。此外,因其保护数据安全的特殊性使得协议存在很大的攻击价值,针对 SSL/TLS 出现的攻击与日剧增:降级攻击[1]、重协商攻击[2]、Lucky Thirteen 攻击[3]、POODLE 攻击[4]、BEAST 攻击[5]、Heartbleed 攻击[6]、时间差分攻击[7]、因代码实现问题产生的攻

收稿日期:2017-02-20。国家自然科学基金项目(61472084);上海市科委项目(16DZ1100200)。沈若愚,硕士生,主研领域:密码与信息安全。卢盛祺,博士生。赵运磊,教授。

击[8]等。这些攻击造成的危害与影响面的广泛为当下的网络安全敲响一记警钟。而 TLS1.2 版本的发布距今已有 8 年,从近 10 年针对协议的攻击数量的增长来看,亟需重新制定 TLS 协议的 1.3 版本。TLS1.3 协议的更新与发展过程值得信息安全学术界和产业界的广泛关注。

1　SSL/TLS 协议简介

SSL 起源于 Netscape, SSL3.0[9] 于 1996 年发布,对后续 TLS 的发展有着基本的指导作用。其确定了协议目标,整体的层次结构(主要由记录协议和握手协议组成)以及消息流的顺序。目前的最高版本为 TLS1.2[10]。由于之前版本过于陈旧,存在许多的漏洞,协议遭受的攻击越来越多,严重危及互联网安全。互联网工程任务组开始筹划制定新的版本 TLS1.3[11]。

1.1　SSL3.0-TLS1.3 发展历程

SSL/TLS 的更新发展过程如表 1 所示。协议的具体内容,详见参考文献[9-12]。RFC(Request For Comments)文档是由互联网工程任务组发布的一系列备忘录,是用以记录互联网规范、协议及过程等的标准文件。

表 1　SSL/TLS 历史版本

TLS 协议版本	RFC 文档	时间
SSL3.0[9]	RFC6101	1996
TLS1.0[10](SSL3.1)	RFC2246	1999
TLS1.1[11]	RFC4346	2006
TLS1.2[12]	RFC5246	2008
TLS1.3	2014 年 4 月开始制定,现未正式发布	

1.2　TLS1.3 协议结构

TLS1.3 由三部分组成,握手协议、警告协议和记录协议。如图 1 灰色方框所示。其在 TCP/IP 协议中介于应用层协议和可靠的传输层协议之间,且独立于应用层协议,因此可以置于很多不同的协议之下,如 HTTP、FTP、SMTP、XMTP 等。

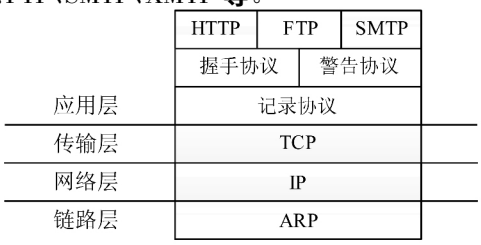


图 1　TLS 协议所处位置

当客户端和服务器端双方第一次建立连接时可通过握手协议协商协议版本,选择密码算法,认证通信双方,协商算法所需参数,且能防篡改。握手协议完成后,记录协议用握手协议协商好的算法和参数对消息流进行分块加密。

1.2.1　握手协议

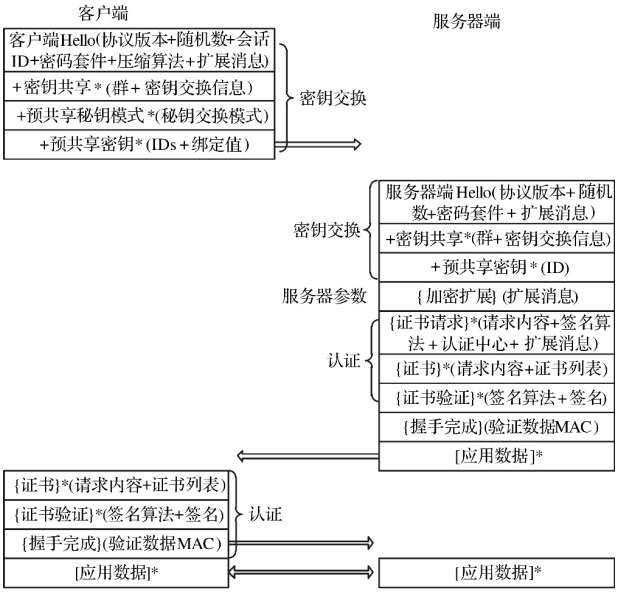
如图 2 所示,握手协议有三个阶段:

1) 密钥交换: 选择协议版本和密码算法,协商算法所需参数,为明文传输。

2) 服务器参数: 建立其他的握手协议参数,如是否需要认证客户端,消息由握手层流密钥(handshake traffic secret)加密传输。

3) 认证: 认证通信双方(客户端认证可选),并保证握手消息的完整性,消息由握手层流密钥加密传输。

这三个阶段完成后,便可进行应用层数据的传输,应用层数据由流密钥(traffic secret)加密后传输。



注: +:上一消息的扩展消息; *:可选发送;
{}:用握手层流密钥加密; Ⅰ:用流密钥加密

图 2　TLS 握手协议消息流

1.2.2　记录协议

记录协议位于握手协议下层,发送方从高层接受任意长度的非空数据,对其进行合并或分块处理,然后利用带有辅助数据的认证加密 AEAD(authenticated encryption with associated data)进行加密传输。接收方接收数据后对其进行解密和验证,重组后再传送给高层用户。记录协议得到要发送的消息之后,进行如图 3 所示的两个步骤的处理。

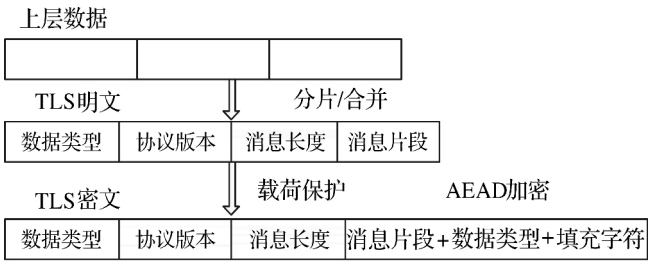


图3 记录协议流程

分片: 把上层数据分片或合并成易于处理的数据分组,大小不超过 2^{14} 字节。记录协议的数据类型有三种: 握手消息,应用数据,警告消息。警告消息的级别有两种: 一种是预警错误,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; 一种是致命错误,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。注意: 对记录层消息进行分片处理时不能对警告消息进行分块处理。此外,握手消息和警告消息的消息长度不能为 0,应用数据的消息长度可以为 0,用来防范针对流量分析的攻击。

载荷保护: 将明文数据通过 AEAD 认证加密算法加密为密文,密文数据的长度略大于明文数据长度。明文数据结构由消息片段,数据类型和任意长度的填充 0 值这三个部分组成,作为 AEAD 的一个输入值计算得到加密内容。TLS 密文数据结构包括四个部分: 数据类型、协议版本、消息长度和加密内容。其中数据类型字段一直都被设置为应用数据,真实的数据类型可以从明文的数据类型获取,该字段是为了兼容之前版本而存在。协议版本与明文消息的协议版本功能一模一样,一直被设置为 0x0301(TLS1.0),真实的协议版本可以从客户端和服务器端的 Hello 消息获取。因此,TLS 密文中的数据类型和协议版本这两个字段在后续版本中可能会被除去。

0 值填充主要是为发送方隐藏真实数据长度而设计。接收方解密密文后,从后往前读取数据,直到遇到非 0 的值,即明文的数据类型这一字段。

2 TLS1.3 主要改变

TLS1.3 从 2014 年 4 月 17 日起开始更新,至今更新到 TLS1.3 Draft18 [13] 版本。本节将深入梳理并讨论 TLS1.3 相对于 TLS1.2 的几个革新性的改变。

2.1 握手层消息结构的改变

从图 2 握手协议的消息流可以看出,相较 TLS1.2,握手协议的结构发生了较大改变: 移除“改变密码规范”和“密钥交换”消息; 为通信双方新增了几个 Hello 扩展消息且新增“请求重新握手”消息; “握手完成”消息对握手层的所有消息进行哈希,进一步保证了消息的完整性。握手协议从之前的四个交互步骤简化为三个步骤。

2.2 算法的移除与更新

计算机的软硬件各方面性能都得到了极大的提升,计算速度以惊人的速度在增长,攻击方法也一直在涌现。因此,之前认为安全的算法于近期及将来都不能确保数据安全,亟需选择可靠性、安全性更高的算法来替换易被攻击的算法。TLS1.3 移除的算法主要有: DH、RC4、记录层压缩算法、重协商、一些不安全的利用率不高的椭圆曲线算法,以及哈希算法(如 MD5、SHA-1、SHA-224 等)。新引入的算法有: AEAD 算法,以及从 RFC4492 (ECC Cipher Suites for TLS) 中引入 CFRG 曲线和签名算法等。

2.3 密钥生成表

TLS 握手协议协商生成一个或多个密钥,如图 4 所示,以这些密钥作为输入通过哈希密钥推导提取函数和哈希密钥推导扩展函数推导出多个记录层所需的密钥。哈希密钥推导提取函数按照箭头方向从上接收参数做加盐操作,从右边接收因特网密钥管理参数。密钥扩展函数有三个输入值: 密钥、标签和握手消息。按照图 4 箭头方向所示从上端接收参数作为第一个密钥参数。

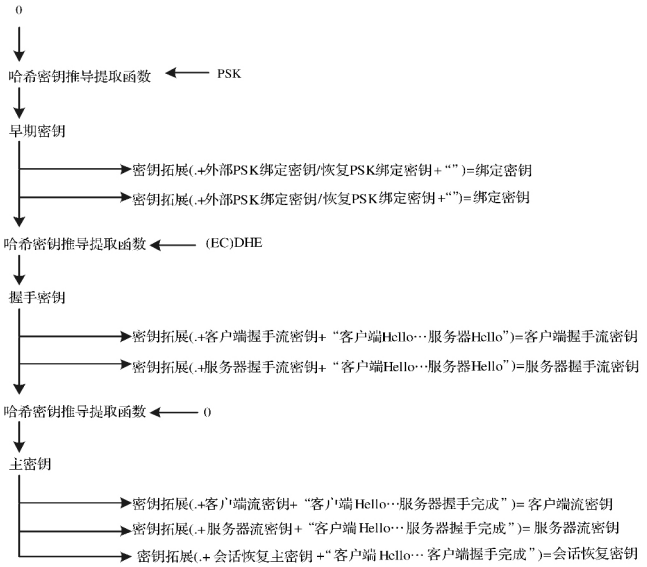


图4 密钥生成表

由于不同的密码方案存在不同的安全性限制,需要为不同的密码算法设置各自的密钥更新时间限度。可结合握手层的“密钥更新”消息告知通信双方及时更新密钥。

2.4 PSK

PSK 预共享密钥交换模式主要是为了简化频繁通信的双方握手协议的认证过程: 如图 5 所示,共享有效

“会话票据”(见 2.4.1)的通信双方不用发送“证书”和“证书认证”消息。第一次握手时,客户端和服务器端建立通信,共享会话票据。连接断开后,客户端再次与服务器建立通信时可以通过 PSK 进行认证,只有真实的通信双方才能解密会话票据得到票据,知道相应的会话恢复密钥,后续才能导出加密应用数据的流密钥。如图 6 所示,后续握手阶段,客户端通过“预共享密钥模式”消息告诉服务器其选择的密钥交换模式,通过“预共享密钥”告知服务器 PSK IDs,这两个消息必须发送。客户端可以选择给服务器发送“密钥共享”消息,让服务器选择是否接受会话恢复,若不接受,可回退进行完整的握手协议。

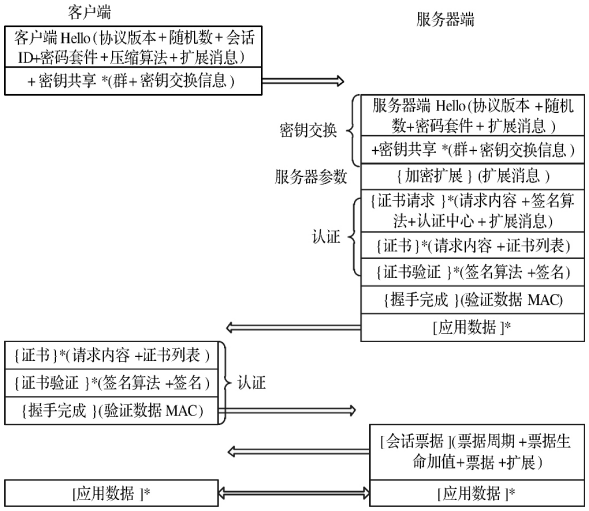
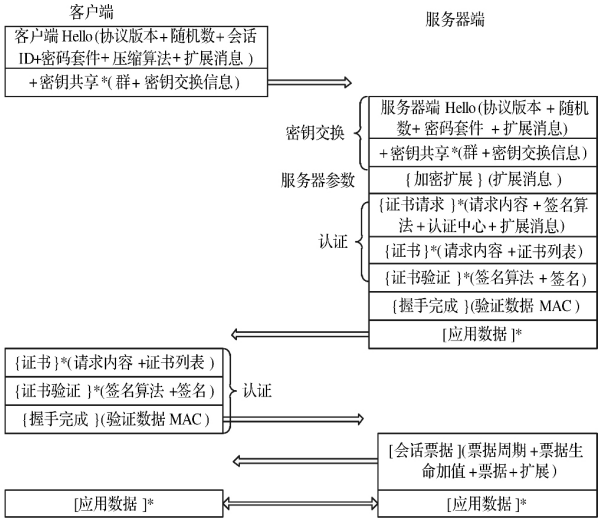


图 5　PSK 消息流(第一次握手)

如果使用 PSK 模式,客户端必须发送“预共享密钥模式”(见 2.4.2)消息。预共享密钥交互模式有两种:(EC)DHE-PSK 和纯 PSK 模式。前一种模式在减少数据传输时延的同时还可以保证前向安全,而后一种模式则不能保证。



注: +:上一消息的扩展消息; *:可选发送;
{}:用握手流密钥加密; ‖:用流密钥加密

图 6　PSK 消息流(第二次握手)

2.4.1　会话票据

在服务器给客户端发送“握手完成”消息以后的任意时间里,服务器都可以给客户端发送“会话票据”消息(如图 5 灰色方框)。会话票据绑定了票据和会话恢复密钥。会话票据(图 7)消息包含 4 个字段。后续握手协议中,客户端可以在“预共享密钥”消息中将密钥名字(也即 PSK ID)发送给服务器。客户端不能重复使用一个票据多次,且优先使用最新的票据。

为了加强安全性,为票据设置了一个 32 比特的生命周期,最长时长为 7 天。票据生命加值是一个随机生成的 32 比特的数值。由于会话票据是加密传输,而预共享秘钥是明文传输,客户端可利用这个加值来混淆 PSK ID 的真实票据时长。票据(图 8)由服务器创建,其本身一个用于数据库查询的关键字或者是一串密文,只有服务器自身才能查询或解密 PSK ID 恢复之前的会话状态。包含四个字段,密钥名字也即“预共享密钥”消息中的 PSK ID,方便服务器根据 ID 快速地找到自己签发过的票据; 加密状态字段主要用来存储第一次握手的状态信息; MAC 值防止票据被攻击者篡改; IV 用作计算 MAC 值和加密状态的初始向量输入。文献[14]提供了票据的构成方案和安全性分析。

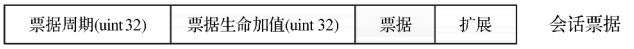


图 7　会话票据消息结构

图 8　票据消息结构

2.4.2　预共享密钥

“预共享密钥”扩展消息(图 9)用来发送 PSK 的 ID 值。客户端给服务器端发送一个 IDs 序列,服务器端从中选取一个 ID 返回给客户端,如果没有合适的值,则发送一个警告消息。如果与“早期数据”消息一起发送,将同时开启 0-RTT 交互模式。

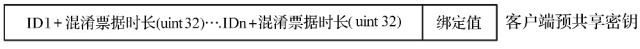


图 9　预共享秘钥消息结构

IDs(客户端):客户端提供给服务器端进行选择的一列 ID 值。

ID(服务器端):服务器端从客户端提供的 IDs 中选取的一个 ID 值。

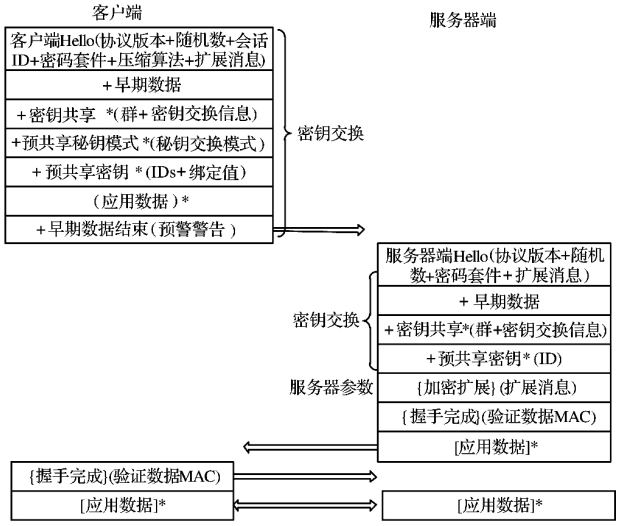
混淆票据时长: 客户端收到会话票据的时间加上会话票据的票据生命加值字段。由于预共享密钥是明文传输,所以这一策略隐藏了真实的票据时长。

绑定值: 绑定值利用 HMAC 算法建立了一个 PSK 和当前的握手消息的绑定。在服务器接收客户端 PSK 认证之前,首先要验证这个绑定值。如果这个值不存在或者验证失败,服务器必须终止会话,这个认证过程

主要是为了防止中间人攻击。握手消息从“客户端 Hello 消息”开始到“预共享密钥的”ID 字段为止，但不包含绑定值本身。例如：客户端发送一个 Hello1 消息，绑定值的 HMAC 计算将只包含这个 Hello1 消息；如果客户端发送一个 Hello1 消息，服务器回了一个“请求重新握手”消息，客户端再发送一个 Hello2 消息，绑定值的 HMAC 计算将包含：Hello1 + 请求重新握手 + Hello2。

2.5 0-RTT

TLS1.3 支持“0-RTT(RoundTripTime)”模式，客户端在发送“客户端 Hello”等消息时可一并发送应用层的数据，因此减少了握手延迟。0-RTT 模式必须和 PSK 模式一起使用，0-RTT 数据由 PSK 导出的早期流密钥加密传输。0-RTT 的消息交互过程如图 10 所示。



注：+：上一个消息的扩展消息；*：消息可选发送；()：用客户端早期流密钥加密；{}：用握手层流密钥加密；Ⅰ：用流密钥加密

图 10 0-RTT 握手协议消息流

这种模式没有完整版的 TLS 握手协议安全：1）0-RTT 数据不能保证前向安全，因为是用 PSK 导出的早期流密钥加密；2）不能保证不受重放攻击，除非服务器采取协议外的防范措施。因此，各方针对 0-RTT 的讨论较为激烈。反对方认为：1）0-RTT 存在上述两个威胁，影响协议的安全性；2）1-RTT 模式已经能满足要求，没必要引入隐患；3）协议的实现者不一定精通密码学知识，盲目相信标准文档的安全性，且为了追求效率使用 0-RTT，忽略了需要采取协议外的安全策略这一需求。支持方则坚持：1）可以为了效率，对安全性做出适当让步，且谷歌公司已经在使用这一方案且效果良好；2）不宜直接反对并废除这一机制，应该给通信双方选择的权利；3）可以把 0-RTT 模式作为一个扩展文档，与 TLS1.3 标准文档区分，需要使用的时候即可作为扩展包添加使用。

当客户端选择 PSK 模式进行认证的时候，客户端在发送“客户端 Hello”消息时必须发送“早期数据”、“预共享密钥模式”和“预共享密钥”消息。客户端可以一直发送 0-RTT 应用数据直到收到服务器的“握手完成”消息，这时客户端需要发送“早期数据结束”预警警告。服务器收到客户端的“早期数据”消息以后有三种表现形式：

1）忽略这个消息，不做任何应答，也即回归正常的 1-RTT 握手。

2）发送“请求重新握手”消息，让客户端重新发送 Hello 消息，该消息将不再包含“早期数据”消息。

3）返回与之相应的“早期数据”扩展消息，接受 0-RTT 的早期数据。

3 SSL/TLS 攻击以及 TLS1.3 防御措施

今年来，随着互联网的使用越加广泛和方便，越来越多的数据和信息通过互联网进行传输。与此同时针对 SSL/TLS 协议的攻击方法与日俱增，安全问题也逐渐浮出水面。本节主要从协议的两层结构以及代码实现这三个方面对近几年出现的攻击进行分类概述，并讨论 TLS1.3 针对这些攻击做出的改进方式。

3.1 针对握手层的攻击

1）降级攻击

降级攻击包括针对协议版本和针对密码中可用加密方式的回滚攻击。由于 TLS1.2 及之前的握手协议中通信双方以明文传输的方式协商密码算法，中间人攻击者可能通过冒充通信的某一方，诱使另一方修改密码算法列表，将协议版本或加密算法降低到不安全的却仍被双方都支持的版本，使得双方通信时使用安全性较低的协议版本和较弱的加密算法。

2）重协商攻击

其攻击原理主要是因为通信双方的身份不是和安全信道绑定的，因此攻击者伪装成客户端与服务器进行握手；建立了通信信道之后再将原客户端要发送的握手信息接入这个安全信道中，完成再次握手，而通信双方并不知情。

3）防御措施

针对降级攻击，TLS1.3 移除了许多不安全的算法，并且在服务器 Hello 消息的随机数字段中内嵌一个降级攻击保护机制。另外，在握手结束消息中对整个握手协议消息流内容进行哈希计算出消息认证码，防止消息被中间人篡改。针对重协商攻击，TLS1.3 移除了重协商这一功能。

3.2　针对记录层的攻击

1) Lucky Thirteen 攻击

其攻击原理是 HMAC 验证计算时,去掉填充信息之后消息的长度不同会导致计算时间上的区别。攻击者对密文进行针对性的修改之后,根据 HMAC 验证的处理时间长短可以获取关于密文的信息。

2) POODLE 攻击

其攻击原理主要是利用 SSL3.0 协议中 CBC 加密算法零值填充无规律且填充字符的完整性在解密时不能进行验证的缺陷。

3) BEAST 攻击

主要利用 CBC 加密模式的链式结构,即除了第一条记录之外,之后每条记录进行加密时的初始向量都是前一条记录的最后一个加密块。攻击者通过某种方式够获取密文一个块的信息,然后逐块破解一段密文。

4) 防御措施

TLS1.3 采用的 AEAD 算法可以避免 Lucky Thirteen 攻击。TLS1.3 不允许将版本回退到 SSL3.0,且将密文分组链接(CBC)加密模式更换为计数器模式(CTR),避免了后两种攻击。

3.3　代码实现方面的攻击

在实现 TLS 协议的时候,应用程序接口设计得较差,且由于标准文档存在部分令人混淆的设置和选项,开发者通常会误用、误解相关参数和选项的使用以及返回值,造成程序实现错误。

1) Heartbleed 攻击

主要是程序员在调用 memcpy( ) 函数时没有对缓冲区边界进行检查而造成的信息泄露。

2) 防御措施

TLS1.3 在附录 B 中增加了许多代码实现方面的注意事项,如: 伪随机数生成器的选取,握手协议某些字段需要留意的特殊取值,如何预防计时攻击等。TLS1.3 主要涉及协议标准的设计,代码的实现主要还是依靠工程师,双方都需引起重视。比如,为 TLS 库提供模糊测试和敌手测试,设计一个简洁稳定的错误汇报机制等。

3.4　其他攻击手段

1) 时间差分攻击

针对对称算法和非对称算法都有计时攻击。其本质是通过观察不同数据的加解密时间差实现密钥破解。

2) 公钥证书验证缺失[15]

另一个常见的攻击来源是公钥证书及其信任链验证的缺失(尤其是基于移动终端的 SSL/TTL 实现)。

3) 防御措施

TLS1.3 采用 AEAD 算法,该算法会对待加密的明文进行填充,隐藏真实数据长度,混淆了加解密过程与时间的关联性。此外,TLS1.3 丰富了证书链表这一部分的说明。

4　TLS1.3 发展趋势

TLS1.3 目前一直处于更新状态,还有许多工作未完成,专家学者正从多方面对协议的制定进行激烈的讨论: 小到算法的选取、参数长度的范围确定,大到新机制的引入、协议安全性证明。

纵观当前讨论热点可总结出以下几个发展趋势: 1) 密码算法的选取、移除与更新; 2) 密码算法密钥的长度以及相关参数范围的设置(最大值,最小值); 3) 0-RTT 和 PSK 模式的进一步优化; 4) TLS 协议扩展文件的更新; 5) 商讨、解决 TLS 1.3 draft 文档中的 open issues; 6) 握手层协议安全性证明[16] 以及安全性分析[17] 这一方向的研究较难,但非常重要。

实际应用中,协议的安全与否主要由两个方面决定: 协议的设计与实现。目前针对协议设计的讨论众多,但代码实现的安全性部分的考虑欠缺。本文作者希望互联网工程任务组在制定新的标准的同时可以为协议的代码实现人员提供一份说明文档,详尽说明协议实现过程中可能会遇到的问题。比如,提供一份详细的消息状态图和协议应用程序编程接口。标注标准文档中安全性较弱的部分。除了对协议的结构的安全性分析与证明,也希望相关学者以及互联网开发人员重视协议代码安全实现这一方面的工作。

5　结　语

近年来,互联网安全性问题频发,SSL/TLS 的安全性研究引起广泛关注。本文从跟进 TLS 协议新版本的制定出发,对 TLS1.3 几个革新性的改变: 密钥编排表、PSK 和 0-RTT 进行了分析与梳理,详细地描绘了协议的结构以及握手层消息的交互过程。同时对近十年协议受到的攻击按照协议的层次分类进行概述,提炼出每种攻击的原理以及 TLS1.3 针对这些攻击做出的应对措施。最后对 TLS 协议的更新发展趋势进行总结和预测。伴随着 TLS1.3 的制订,预计未来 5 年将迎来 TLS 研究的新机遇期。在未来的研究中将继续关注新的密码协议的设计、安全性建模与分析、安全实现和形式化验证、新型漏洞和攻击发展等。

(下转第 329 页)

[4] Dalvik Execuatble Format [OL]. 2016. https://source.android.com/devices/tech/dalvik/dex-format.html.

[5] Strazzere T. Dex education: Practicing safe dex [M]. Black Hat, USA, 2012.

[6] Strazzere T. Dex education 201: anti-emulation [M]. HITCON, 2013.

[7] Vidas T, Christin N. Evading android runtime analysis via sandbox detection [C]//Proceedings of the 9th ACM symposium on Information, computer and communications security. ACM, 2014: 447-458.

[8] Rowland Yu. Android Packers: Facing the challenges, Building solutions [C]//Proc. of the 24th Virus Bulletin International Conference, 2014.

[9] Strazzere T, Sawyer J. Android Hacker Protection Level 0 DEF CON 22 [Z]. 2014.

[10] Zhang Y, Luo X, Yin H. Dexhunter: toward extracting hidden code from packed android applications [M]//European Symposium on Research in Computer Security. Springer International Publishing, 2015: 293-311.

[11] Yang W B, Zhang Y Y, Li J R, et al. AppSpear: Bytecode Decrypting and DEX Reassembling for Packed Android Malware [M]//Research in Attacks, Intrusions, and Defenses. Springer International Publishing, 2015.

[12] Rasthofer S, Arzt S, Miltenberger M, et al. Harvesting runtime values in android applications that feature anti-analysis techniques [C]//Proceedings of the Annual Symposium on Network and Distributed System Security (NDSS). 2016.

[13] Rastogi V, Chen Y, Enck W. AppsPlayground: automatic security analysis of smartphone applications [C]//Proceedings of the third ACM conference on Data and application security and privacy. ACM, 2013: 209-220.

[14] Monkey Runner [OL]. 2016. https://developer.android.com/studio/test/monkeyrunner/index.html.

[15] Smali and baksmali: Dalvik bytecode toolkit [OL]. 2016. https://github.com/JesusFreke/smali.

[16] Afonso V, Bianchi A, Fratantonio Y, et al. Going native: Using a large-scale analysis of android apps to create a practical native-code sandboxing policy [OL]//Proceedings of the Annual Symposium on Network and Distributed System Security (NDSS), 2016.

(上接第 269 页)

参 考 文 献

[1] Wagner D, Schneier B. Analysis of the ssl 3 [J]. Proceedings of the Second Unix Workshop on Electronic Commerce, 1996, 28(28): 29 - 40.

[2] Giesen F, Kohlar F, Stebila D. On the security of TLS renegotiation [C]//Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security. ACM, 2013: 387 - 398.

[3] Fardan N J A, Paterson K G. Lucky Thirteen: Breaking the TLS and DTLS Record Protocols [C]// IEEE Symposium on Security and Privacy. IEEE Computer Society, 2013: 526 - 540.

[4] Möller B, Duong T, Kotowicz K. This POODLE bites: exploiting the SSL 3.0 fallback [OL]. 2014. https://www.openssl.org/~bodo/ssl-poodle.pdf.

[5] Duong T, Rizzo J. Here Come The ⊕ Ninjas [J]. Unpublished Manuscript, 2011.

[6] 强小辉, 陈波, 陈国凯, 等. OpenSSLHeartBleed 漏洞分析及检测技术研究 [J]. 计算机工程与应用, 2016, 52(9): 88 - 95.

[7] Brumley D, Dan B. Remote timing attacks are practical [J]. Computer Networks, 2005, 48(5): 701 - 716.

[8] Georgiev M, Iyengar S, Jana S, et al. The most dangerous code in the world: validating SSL certificates in non-browser software [C]// Proceedings of the 2012 ACM conference on Computer and communications security. ACM, 2012: 38 - 49.

[9] Freier A, Karlton P, Kocher P. The SSL 3.0 Protocol [Z]. November 1996.

[10] Allen C, Dierks T. The TLS protocol version 1.0 [Z]. 1999.

[11] Dierks T, Rescorla E. The Transport Layer Security (TLS) Protocol Version 1.1 [R]. RFC 4346, DOI 10.17487/RFC4346, April 2006.

[12] Dierks T, Rescorla E. The Transport Layer Security (TLS) Protocol Version 1.2 [R]. RFC 5246, DOI 10.17487/RFC5246, August 2008.

[13] Rescorla E. The Transport Layer Security (TLS) Protocol Version 1.3 [R]. draft-ietf-tls-tls13 - 18, 2016.

[14] Salowey J. Transport Layer Security (TLS) Session Resumption without Server-Side State [Z]. Transport, 2006.

[15] Clark J, Oorschot P C V. SoK: SSL and HTTPS: Revisiting Past Challenges and Evaluating Certificate Trust Model Enhancements [C]// Security and Privacy. IEEE, 2013: 511 - 525.

[16] Bhargavan K, Fournet C, Kohlweiss M, et al. Proving the TLS Handshake Secure (as it is) [J]. Lecture Notes in Computer Science, 2014, 8617: 235 - 255.

[17] Dowling B, Fischlin M, Nther F, et al. A Cryptographic Analysis of the TLS 1.3 Handshake Protocol Candidates [C]// The, ACM Sigsac Conference. ACM, 2015: 1197 - 1210.