

## 转 SSL/TLS协商过程详解

2017年02月16日 16:06:35

阅读数: 2618

本文大部分整理自网络，相关文章请见文后参考。

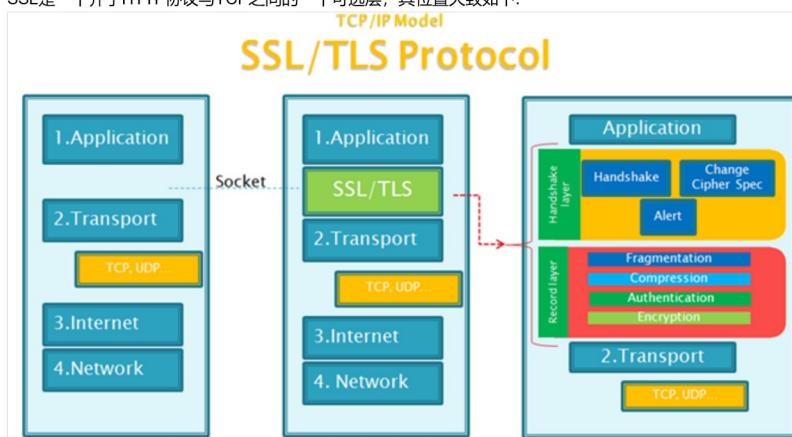
关于证书授权中心CA以及数字证书等概念，请移步 [OpenSSL 与 SSL 数字证书概念贴](#)，如果你想快速自建CA然后签发数字证书，请移步 [基于OpenSSL自建CA和颁发SSL证书](#)。

SSL/TLS作为一种互联网安全加密技术，原理较为复杂，枯燥而无味，我也是试图理解之后重新整理，尽量做到层次清晰。正文开始。

## 1. SSL/TLS概览

### 1.1 整体结构

SSL是一个介于HTTP协议与TCP之间的一个可选层，其位置大致如下：



- **SSL:** (Secure Socket Layer, 安全套接字层)，为Netscape所研发，用以保障在Internet上数据传输之安全，利用数据加密(Encryption)技术，可确保数据在网络上之传输过程中不会被截取。当前版本为3.0。它已被广泛地用于Web浏览器与服务器之间的身份认证和加密数据传输。

SSL协议位于TCP/IP协议与各种应用层协议之间，为数据通讯提供安全支持。SSL协议可分为两层：SSL记录协议 (SSL Record Protocol)：它建立在可靠的传输协议（如TCP）之上，为高层协议提供数据封装、压缩、加密等基本功能的支持。SSL握手协议 (SSL Handshake Protocol)：它建立在SSL记录协议之上，用于在实际的数据传输开始前，通讯双方进行身份认证、协商加密算法、交换加密密钥等。

- **TLS:** (Transport Layer Security, 传输层安全协议)，用于两个应用程序之间提供保密性和数据完整性。

TLS 1.0是IETF (Internet Engineering Task Force, Internet工程任务组) 制定的一种新的协议，它建立在SSL 3.0协议规范之上，是SSL 3.0的后续版本，可以理解为SSL 3.1，它是写入了 [RFC](#) 的。该协议由两层组成：TLS 记录协议 (TLS Record) 和 TLS 握手协议 (TLS Handshake)。较低的层为 TLS 记录协议，位于某个可靠的传输协议（例如 TCP）上面。

SSL/TLS协议提供的服务主要有：

1. 认证用户和服务器，确保数据发送到正确的客户机和服务器；
2. 加密数据以防止数据中途被窃取；
3. 维护数据的完整性，确保数据在传输过程中不被改变。

### 1.2 TLS与SSL的差异

1. 版本号：TLS记录格式与SSL记录格式相同，但版本号的值不同，TLS的版本1.0使用的版本号为SSLv3.1。
2. 报文鉴别码：SSLv3.0和TLS的MAC算法及MAC计算的范围不同。TLS使用了RFC-2104定义的HMAC算法。SSLv3.0使用了相似的算法，两者差别在于SSLv3.0中，填充字节与密钥之间采用的是连接运算，而HMAC算法采用的是异或运算。但是两者的安全程度是相同的。
3. 伪随机函数：TLS使用了称为PRF的伪随机函数来将密钥扩展成数据块，是更安全的方式。
4. 报警代码：TLS支持几乎所有的SSLv3.0报警代码，而且TLS还补充定义了很多报警代码，如解密失败 (decryption\_failed)、记录溢出 (record\_overflow)、未知CA (unknown\_ca)、拒绝访问 (access\_denied) 等。
5. 密文族和客户证书：SSLv3.0和TLS存在少量差别，即TLS不支持Fortezza密钥交换、加密算法和客户证书。

6. certificate\_verify和finished消息：SSLv3.0和TLS在用certificate\_verify和finished消息计算MD5和SHA-1散列码时，计算的输入有少许差别，但安全性相当。
7. 加密计算：TLS与SSLv3.0在计算主密钥（master secret）时采用的方式不同。
8. 填充：用户数据加密之前需要增加的填充字节。在SSL中，填充后的数据长度要达到密文块长度的最小整数倍。而在TLS中，填充后的数据长度可以是密文块长度的任意整数倍（但填充的最大长度为255字节），这种方式可以防止基于对报文长度进行分析的攻击。

#### TLS的主要增强内容

TLS的主要目标是使SSL更安全，并使协议的规范更精确和完善。TLS在SSL v3.0的基础上，提供了以下增强内容：

1. 更安全的MAC算法
2. 更严密的警报
3. “灰色区域”规范的更明确的定义

#### TLS对于安全性的改进

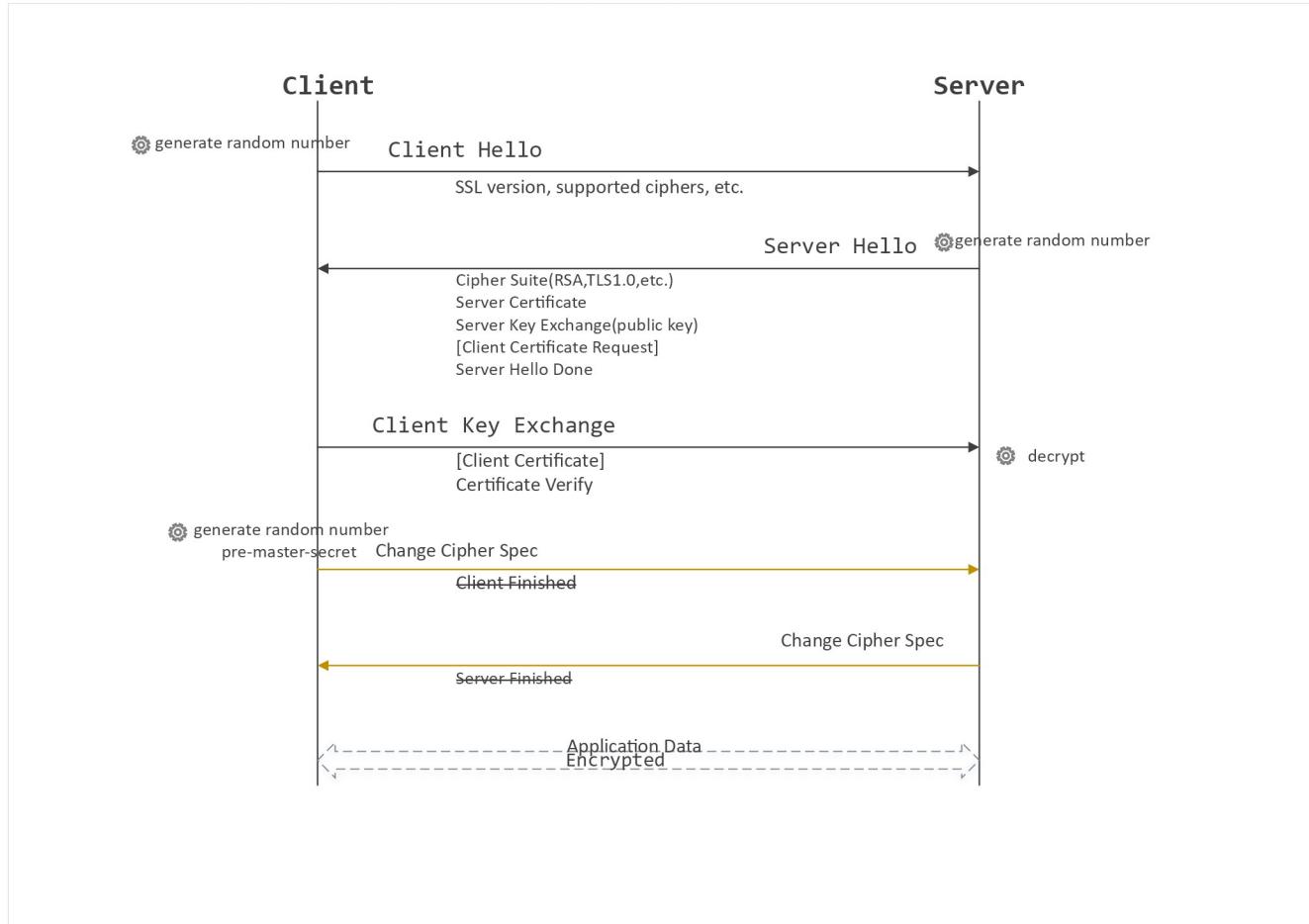
1. 对于消息认证使用密钥散列法：TLS 使用“消息认证代码的密钥散列法”（HMAC），当记录在开放的网络（如因特网）上传送时，该代码确保记录不会被变更。SSLv3.0还提供键控消息认证，但HMAC比SSLv3.0使用的（消息认证代码）MAC功能更安全。
2. 增强的伪随机功能（PRF）：PRF生成密钥数据。在TLS中，HMAC定义PRF。PRF使用两种散列算法保证其安全性。如果任一算法暴露了，只要第二种算法未暴露，则数据仍然是安全的。
3. 改进的已完成消息验证：TLS和SSLv3.0都对两个端点提供已完成的消息，该消息认证交换的消息没有被变更。然而，TLS将此已完成消息基于PRF和HMAC值之上，这也比SSLv3.0更安全。
4. 一致证书处理：与SSLv3.0不同，TLS试图指定必须在TLS之间实现交换的证书类型。
5. 特定警报消息：TLS提供更多的特定和附加警报，以指示任一话端点检测到的问题。TLS还对何时应该发送某些警报进行记录。

## 2. 密钥协商过程——TLS握手

SSL协议分为两部分：Handshake Protocol和Record Protocol。其中Handshake Protocol用来协商密钥，协议的大部分内容就是通信双方如何利用它来安全的协商出一份密钥。Record Protocol则定义了传输的格式。

由于非对称加密的速度比较慢，所以它一般用于密钥交换，双方通过公钥算法协商出一份密钥，然后通过对称加密来通信，当然，为了保证数据的完整性，在加密前要先经过HMAC的处理。

SSL缺省只进行server端的认证，客户端的认证是可选的。以下是其流程图（摘自TLS协议）。



## 2.1 客户端发出请求 (ClientHello)

由于客户端(如浏览器)对一些加解密算法的支持程度不一样，但是在TLS协议传输过程中必须使用同一套加解密算法才能保证数据能够正常的加解密。在TLS握手阶段，客户端首先要告知服务端，自己支持哪些加密算法，所以客户端需要将本地支持的加密套件(Cipher Suite)的列表传送给服务端。除此之外，客户端还要产生一个随机数，这个随机数一方面需要在客户端保存，另一方面需要传送给服务端，客户端的随机数需要跟服务端产生的随机数结合起来产生后面要讲到的 Master Secret。

综上，在这一步，客户端主要向服务器提供以下信息：

1. 支持的协议版本，比如TLS 1.0版
2. 一个客户端生成的随机数，稍后用于生成“对话密钥”
3. 支持的加密方法，比如RSA公钥加密
4. 支持的压缩方法

## 2.2 服务器回应 (ServerHello)

上图中，从Server Hello到Server Done，有些服务端的实现是每条单独发送，有服务端实现是合并到一起发送。Server Hello和Server Done都是只有头没有内容的数据。

服务端在接收到客户端的Client Hello之后，服务端需要将自己的证书发送给客户端。这个证书是对于服务端的一种认证。例如，客户端收到了一个来自于称自己是www.alipay.com的数据，但是如何证明对方是合法的alipay支付宝呢？这就是证书的作用，支付宝的证书可以证明它是alipay，而不是财付通。证书是需要申请，并由专门的数字证书认证机构(CA)通过非常严格的审核之后颁发的电子证书。颁发证书的同时会产生一个私钥和公钥。私钥由服务端自己保存，不可泄漏。公钥则是附带在证书的信息中，可以公开的。证书本身也附带一个证书电子签名，这个签名用来验证证书的完整性和真实性，可以防止证书被串改。另外，证书还有个有效期。

在服务端向客户端发送的证书中没有提供足够的信息（证书公钥）的时候，还可以向客户端发送一个 Server Key Exchange。

此外，对于非常重要的保密数据，服务端还需要对客户端进行验证，以保证数据传给了安全的合法的客户端。服务端可以向客户端发出 Certificate Request 消息，要求客户端发送证书对客户端的合法性进行验证。比如，金融机构往往只允许认证客户连入自己的网络，就会向正式客户提供USB密钥，里面就包含了一张客户端证书。

跟客户端一样，服务端也需要产生一个随机数发送给客户端。客户端和服务端都需要使用这两个随机数来产生Master Secret。

最后服务端会发送一个Server Hello Done消息给客户端，表示Server Hello消息结束了。

综上，在这一步，服务器的回应包含以下内容：

1. 确认使用的加密通信协议版本，比如TLS 1.0版本。如果浏览器与服务器支持的版本不一致，服务器关闭加密通信
2. 一个服务器生成的随机数，稍后用于生成“对话密钥”
3. 确认使用的加密方法，比如RSA公钥加密
4. 服务器证书

## 2.3 客户端回应 (Certificate Verify)

### Client Key Exchange

如果服务端需要对客户端进行验证，在客户端收到服务端的 Server Hello 消息之后，首先需要向服务端发送客户端的证书，让服务端来验证客户端的合法性。

### Certificate Verify

接着，客户端需要对服务端的证书进行检查，如果证书不是可信机构颁布、或者证书中的域名与实际域名不一致、或者证书已经过期，就会向访问者显示一个警告，由其选择是否还要继续通信。如果证书没有问题，客户端就会从服务器证书中取出服务器的公钥。然后，向服务器发送下面三项信息：

1. 一个随机数。该随机数用服务器公钥加密，防止被窃听
2. 编码改变通知，表示随后的信息都将用双方商定的加密方法和密钥发送
3. 客户端握手结束通知，表示客户端的握手阶段已经结束。这一项同时也是前面发送的所有内容的hash值，用来供服务器校验

上面第一项的随机数，是整个握手阶段出现的第三个随机数，它是客户端使用一些加密算法(例如：RSA, Diffie-Hellman)产生一个48个字节的Key，这个Key叫 PreMaster Secret，很多材料上也被称作 PreMaster Key。

### ChangeCipherSpec

ChangeCipherSpec是一个独立的协议，体现在数据包中就是一个字节的数据，用于告知服务端，客户端已经切换到之前协商好的加密套件 (Cipher Suite) 的状态，准备使用之前协商好的加密套件加密数据并传输了。

在ChangeCipherSpec传输完毕之后，客户端会使用之前协商好的加密套件和Session Secret加密一段 Finish 的数据传送给服务端，此数据是为了在正式传输应用数据之前对刚刚握手建立起来的加解密通道进行验证。

## 2.4 服务器的最后回应 (Server Finish)

服务端在接收到客户端传过来的 PreMaster 加密数据之后，使用私钥对这段加密数据进行解密，并对数据进行验证，也会使用跟客户端同样的方式生成 Session Secret，一切准备好了之后，会给客户端发送一个 ChangeCipherSpec，告知客户端已经切换到协商过的加密套件状态，准备使用加密套件和 Session Secret 加密数据了。之后，服务端也会使用 Session Secret 加密一段 Finish 消息发送给客户端，以验证之前通过握手建立起来的加解密通道是否成功。

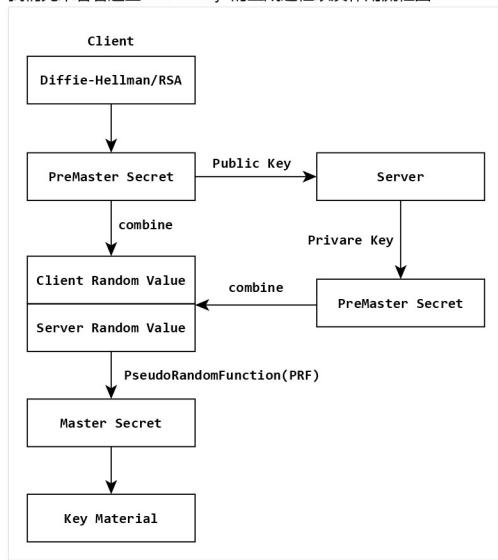
根据之前的握手信息，如果客户端和服务端都能对Finish信息进行正常加解密且消息正确的被验证，则说明握手通道已经建立成功，接下来，双方可以使用上面产生的Session Secret 对数据进行加密传输了。

## 2.5 几个secret

### Secret Keys

上面的分析和讲解主要是为了突出握手的过程，所以PreMaster secret, Master secret, session secret都是一带而过，但是对于Https, SSL/TLS深入的理解和掌握，这些Secret Keys是非常重要的部分。所以，准备把这些Secret Keys抽出来单独分析和讲解。

我们先来看看这些Secret Keys的生成过程以及作用流程图：



### PreMaster secret

PreMaster Secret是在客户端使用RSA或者Diffie-Hellman等加密算法生成的。它将用来跟服务端和客户端在Hello阶段产生的随机数结合在一起生成 Master Secret。在客户端使用服务端的公钥对PreMaster Secret进行加密之后传送给服务端，服务端将使用私钥进行解密得到PreMaster secret。也就是说服务端和客户端都有一份相同的PreMaster secret和随机数。

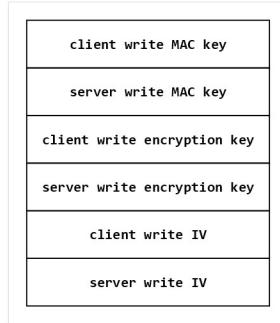
PreMaster secret前两个字节是TLS的版本号，这是一个比较重要的用来核对握手数据的版本号，因为在Client Hello阶段，客户端会发送一份加密套件列表和当前支持的SSL/TLS的版本号给服务端，而且是使用明文传送的，如果握手的数据包被破解之后，攻击者很有可能串改数据包，选择一个安全性较低的加密套件和版本给服务端，从而对数据进行破解。所以，服务端需要对密文中解密出来对的PreMaster版本号跟之前Client Hello阶段的版本号进行对比，如果版本号变低，则说明被串改，则立即停止发送任何消息。

关于PreMaster Secret(Key)的计算请参考 [Https SSL/TLS PreMaster/Master Secret\(Key\)计算](#)。

### Master secret

上面已经提到，由于服务端和客户端都有一份相同的PreMaster secret和随机数，这个随机数将作为后面产生Master secret的种子，结合PreMaster secret，客户端和服务端将计算出同样的Master secret。

Master secret是有系列的hash值组成的，它将作为数据加解密相关的secret的 Key Material 的一部分。Key Material最终解析出来的数据如下：



其中，write MAC key，就是session secret或者说session key。Client write MAC key是客户端发数据的session secret，Server write MAC secret是服务端发送数据的session key。MAC(Message Authentication Code)，是一个数字签名，用来验证数据的完整性，可以检测到数据是否被串改。

关于Session Secret(Key)的计算请参考 [Https SSL/TLS Session Secret\(Key\)计算](#)。

## 2.6 应用数据传输

在所有的握手阶段都完成之后，就可以开始传送应用数据了。应用数据在传输之前，首先要附加MAC secret，然后再对这个数据包使用write encryption key进行加密。在服务端收到密文之后，使用Client write encryption key进行解密，客户端收到服务端的数据之后使用Server write encryption key进行解密，然后使用各自的write MAC key对数据的完整性包括是否被串改进行验证。

## 2.7 总结

SSL客户端（也是TCP的客户端）在TCP链接建立之后，发出一个ClientHello来发起握手，这个消息里面包含了自己可实现的算法列表和其它一些需要的消息，SSL的服务器端会回应一个ServerHello，这里面确定了这次通信所需要的算法，然后发过去自己的证书（里面包含了身份和自己的公钥）。Client在收到这个消息后会生成一个秘密消息，用SSL服务器的公钥加密后传过去，SSL服务器端用自己的私钥解密后，会话密钥协商成功，双方可以用同一份会话密钥来通信了。

## 3. 附：密钥协商的形象化比喻

如果上面的说明不够清晰，这里我们用个形象的比喻，我们假设A与B通信，A是SSL客户端，B是SSL服务器端，加密后的消息放在方括号[]里，以突出明文消息的区别。双方的处理动作的说明用圆括号()括起。

A: 我想和你安全的通话，我这里的对称加密算法有DES,RC5,密钥交换算法有RSA和DH，摘要算法有MD5和SHA。

B: 我们用DES - RSA - SHA这对组合好了。

这是我的证书，里面有我的名字和公钥，你拿去验证一下我的身份（把证书发给A）。

目前没有别的可说的了。

A: （查看证书上B的名字是否无误，并通过手头早已有的CA的证书验证了B的证书的真实性，如果其中一项有误，发出警告并断开连接，这一步保证了B的公钥的真实性）

（产生一份秘密消息，这份秘密消息处理后将用作加密密钥，加密初始化向量(IV)和hmac的密钥。将这份秘密消息-协议中称为per\_master\_secret-用B的公钥加密，封装成称作ClientKeyExchange的消息。由于用了B的公钥，保证了第三方无法窃听）

我生成了一份秘密消息，并用你的公钥加密了，给你（把ClientKeyExchange发给B）

注意，下面我就要用加密的办法给你发消息了！

（将秘密消息进行处理，生成加密密钥，加密初始化向量和hmac的密钥）

[我说完了]

B: （用自己的私钥将ClientKeyExchange中的秘密消息解密出来，然后将秘密消息进行处理，生成加密密钥，加密初始化向量和hmac的密钥，这时双方已经安全的协商出一套加密办法了）

注意，我也要开始用加密的办法给你发消息了！

[我说完了]

A: [我的秘密是...]

B: [其它人不会听到的...]

## 4. SSL安全性

SecurityPortal在2000年底有一份文章《The End of SSL and SSH?》激起了很多的讨论，目前也有一些成熟的工具如dsniff (<http://www.monkey.org/~dugsong/dsniff/>) 可以通过man in the middle攻击来截获https的消息。

从上面的原理可知，SSL的结构是严谨的，问题一般出现在实际不严谨的应用中。常见的攻击就是middle in the middle攻击，它是指在A和B通信的同时，有第三方C处于信道的中间，可以完全听到A与B通信的消息，并可拦截，替换和添加这些消息。

1. SSL可以允许多种密钥交换算法，而有些算法，如DH，没有证书的概念，这样A便无法验证B的公钥和身份的真实性，从而C可以轻易的冒充，用自己的密钥与双方通信，从而窃听到别人谈话的内容。

而为了防止middle in the middle攻击，应该采用有证书的密钥交换算法。

2. 有了证书以后，如果C用自己的证书替换掉原有的证书之后，A的浏览器会弹出一个警告框进行警告，但又有多少人会注意这个警告呢？

3. 由于美国密码出口的限制，IE, netscape等浏览器所支持的加密强度是很弱的，如果只采用浏览器自带的加密功能的话，理论上存在被破解可能。

## 5. 代理

下面探讨一下SSL的代理是怎样工作的

当在浏览器里设置了https的代理，而且里输入了<https://www.example.com>之后，浏览器会与proxy建立tcp链接，然后向其发出这么一段消息：

```
1 | CONNECT server.example.com:443 HTTP/1.1
2 | Host: server.example.com:443
```

然后proxy会向webserver端建立tcp连接，之后，这个代理便完全成了个内容转发装置。浏览器与web server会建立一个安全通道，因此这个安全通道是端到端的，尽管所有的信息流过了proxy，但其内容proxy是无法解密和改动的（当然要由证书的支持，否则这个地方便是个man in the middle攻击的好场所，见上面的安全部分）。

CA证书以及如何使用OpenSSL自签署，见文章[OpenSSL自签署证书](#)。

## 6. 参考

- [Https\(SSL/TLS\)原理详解](#)
- [SSL与TLS的区别以及介绍](#)
- [SSL/TLS协议运行机制的概述](#)
- [SSL/TLS/WTLS原理](#)
- [Transport Layer Security \(TLS\)](#)
- [传输层安全协议](#)
- [Survival guides - TLS/SSL and SSL \(X.509\) Certificates](#)

原文链接地址：<http://seanlook.com/2015/01/07/tls-ssl-understand/>

转自：<https://segmentfault.com/a/1190000002554673>

个人分类：数通知识 openssl