# SSL/TLS协议运行机制的概述

作者： 阮一峰

日期： 2014年2月 5日



互联网的通信安全，建立在SSL/TLS协议之上。

本文简要介绍SSL/TLS协议的运行机制。文章的重点是设计思想和运行过程，不涉及具体的实现细节。如果想了解这方面的内容，请参阅RFC文档。


**一、作用**

不使用SSL/TLS的HTTP通信，就是不加密的通信。所有信息明文传播，带来了三大风险。

(1) **窃听风险**（eavesdropping）：第三方可以获知通信内容。

(2) **篡改风险**（tampering）：第三方可以修改通信内容。

（3） **冒充风险**（pretending）：第三方可以冒充他人身份参与通信。

SSL/TLS协议是为了解决这三大风险而设计的，希望达到：

（1） 所有信息都是**加密传播**，第三方无法窃听。

（2） 具有**校验机制**，一旦被篡改，通信双方会立刻发现。

（3） 配备**身份证书**，防止身份被冒充。

互联网是开放环境，通信双方都是未知身份，这为协议的设计带来了很大的难度。而且，协议还必须能够经受所有匪夷所思的攻击，这使得SSL/TLS协议变得异常复杂。

## 二、历史

互联网加密通信协议的历史，几乎与互联网一样长。

1994年，NetScape公司设计了SSL协议（Secure Sockets Layer）的1.0版，但是未发布。

1995年，NetScape公司发布SSL 2.0版，很快发现有严重漏洞。

1996年，SSL 3.0版问世，得到大规模应用。

1999年，互联网标准化组织ISOC接替NetScape公司，发布了SSL的升级版TLS 1.0版。

2006年和2008年，TLS进行了两次升级，分别为TLS 1.1版和TLS 1.2版。最新的变动是2011年TLS 1.2的修订版。

目前，应用最广泛的是TLS 1.0，接下来是SSL 3.0。但是，主流浏览器都已经实现了TLS 1.2的支持。

TLS 1.0通常被标示为SSL 3.1，TLS 1.1为SSL 3.2，TLS 1.2为SSL 3.3。

## 三、基本的运行过程

SSL/TLS协议的基本思路是采用公钥加密法，也就是说，客户端先向服务器端索要公钥，然后用公钥加密信息，服务器收到密文后，用自己的私钥解密。

但是，这里有两个问题。

**（1）如何保证公钥不被篡改？**

> 解决方法：将公钥放在数字证书中。只要证书是可信的，公钥就是可信的。

**（2）公钥加密计算量太大，如何减少耗用的时间？**

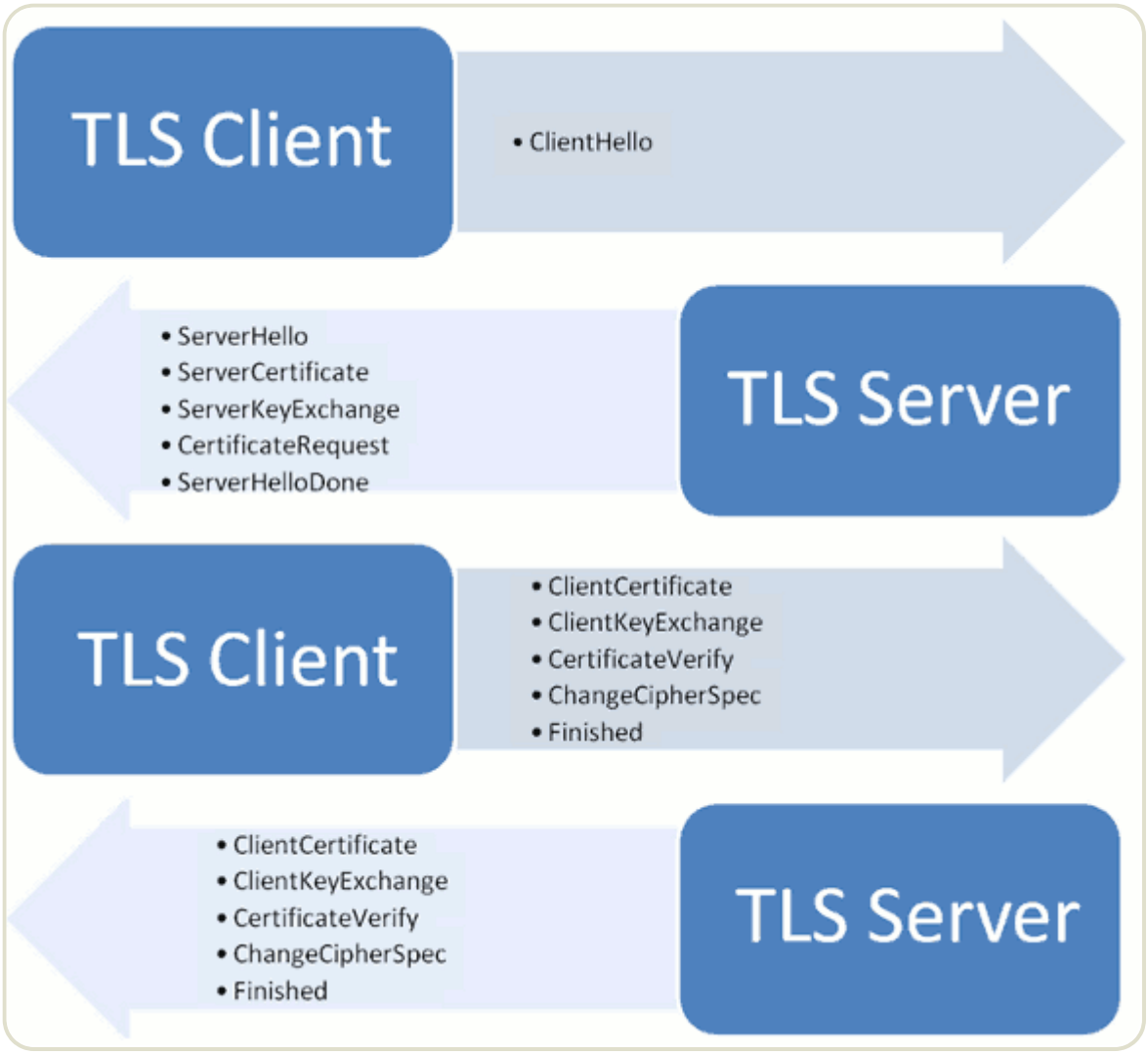
> 解决方法：每一次对话（session），客户端和服务器端都生成一个"对话密钥"（session key），用它来加密信息。由于"对话密钥"是对称加密，所以运算速度非常快，而服务器公钥只用于加密"对话密钥"本身，这样就减少了加密运算的消耗时间。

因此，SSL/TLS协议的基本过程是这样的：

> (1) 客户端向服务器端索要并验证公钥。
>
> (2) 双方协商生成"对话密钥"。
>
> (3) 双方采用"对话密钥"进行加密通信。

上面过程的前两步，又称为"握手阶段"（handshake）。

**四、握手阶段的详细过程**




"握手阶段"涉及四次通信，我们一个个来看。需要注意的是，"握手阶段"的所有通信都是明文的。

### 4.1 客户端发出请求（ClientHello）

首先，客户端（通常是浏览器）先向服务器发出加密通信的请求，这被叫做ClientHello请求。

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。

这里需要注意的是，客户端发送的信息之中不包括服务器的域名。也就是说，理论上服务器只能包含一个网站，否则会分不清应该向客户端提供哪一个网站的数字证书。这就是为什么通常一台服务器只能有一张数字证书的原因。

对于虚拟主机的用户来说，这当然很不方便。2006年，TLS协议加入了一个Server Name Indication扩展，允许客户端向服务器提供它所请求的域名。

### 4.2 服务器回应（SeverHello）

服务器收到客户端请求后，向客户端发出回应，这叫做SeverHello。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。

除了上面这些信息，如果服务器需要确认客户端的身份，就会再包含一项请求，要求客户端提供"客户端证书"。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。

### 4.3 客户端回应

客户端收到服务器回应以后，首先验证服务器证书。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，又称"pre-master key"。有了它以后，客户端和服务器就同时有了三个随机数，接着双方就用事先商定的加密方法，各自生成本次会话所用的同一把"会话密钥"。

至于为什么一定要用三个随机数，来生成"会话密钥"，dog250解释得很好：

> "不管是客户端还是服务器，都需要随机数，这样生成的密钥才不会每次都一样。由于SSL协议中证书是静态的，因此十分有必要引入一种随机因素来保证协商出来的密钥的随机性。
>
> 对于RSA密钥交换算法来说，pre-master-key本身就是一个随机数，再加上hello消息中的随机，三个随机数通过一个密钥导出器最终导出一个对称密钥。
>
> pre master的存在在于SSL协议不信任每个主机都能产生完全随机的随机数，如果随机数不随机，那么pre master secret就有可能被猜出来，那么仅适用pre master secret作为密钥就不合适了，因此必须引入新的随机因素，那么客户端和服务器加上pre master secret三个随机数一同生成的密钥就不容易被猜出了，一个伪随机可能完全不随机，可是是三个伪随机就十分接近随机了，每增加一个自由度，随机性增加的可不是一。"

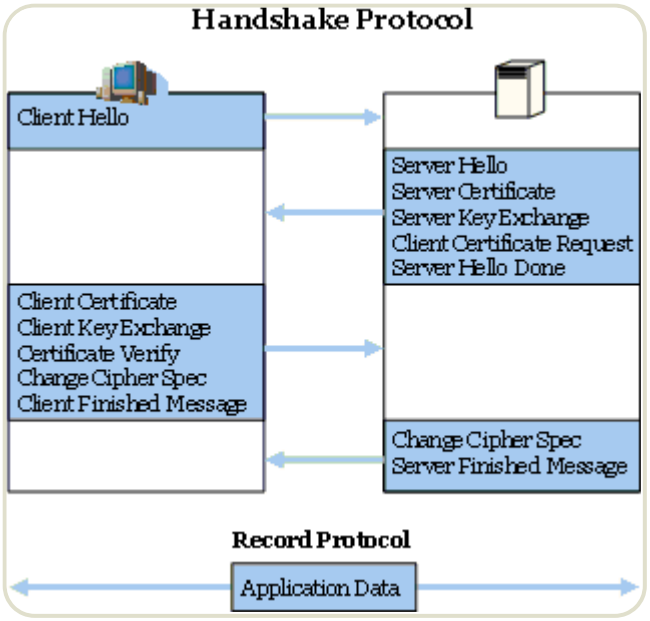
此外，如果前一步，服务器要求客户端证书，客户端会在这一步发送证书及相关信息。

### 4.4 服务器的最后回应

服务器收到客户端的第三个随机数pre-master key之后，计算生成本次会话所用的"会话密钥"。然后，向客户端最后发送下面信息。

> （1）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。
>
> （2）服务器握手结束通知，表示服务器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，用来供客户端校验。

至此，整个握手阶段全部结束。接下来，客户端与服务器进入加密通信，就完全是使用普通的HTTP协议，只不过用"会话密钥"加密内容。




**五、参考链接**

- MicroSoft TechNet, SSL/TLS in Detail
- Jeff Moser, The First Few Milliseconds of an HTTPS Connection
- Wikipedia, Transport Layer Security
- StackExchange, How does SSL work?

（完）


## 文档信息

- 版权声明：自由转载-非商用-非衍生-保持署名（创意共享3.0许可证）
- 发表日期： 2014年2月 5日
- 更多内容： 档案 » 开发者手册
- 文集：《前方的路》，《未来世界的幸存者》
- 社交媒体： twitter， weibo

## 相关文章

- **2018.03.05:** HTTP/2 服务器推送（Server Push）教程

  > HTTP/2 协议的主要目的是提高网页性能。

- **2018.02.27:** Nginx 容器教程

  > 春节前，我看到 Nginx 加入了 HTTP/2 的 server push 功能，就很想试一下。

- **2018.02.13:** Docker 微服务教程

  > Docker 是一个容器工具，提供虚拟环境。很多人认为，它改变了我们对软件的认识。

- **2018.02.09:** Docker 入门教程

  > 2013年发布至今， Docker 一直广受瞩目，被认为可能会改变软件行业。

## 广告（购买广告位）