<https://blog.csdn.net/wm_1991/article/details/51954571>

# HTTPS详解SSL/TLS

2016年07月19日 11:52:30 [juary\_01](https://me.csdn.net/wm_1991) 阅读数：11406更多

个人分类： [网络编程](https://blog.csdn.net/wm_1991/article/category/5855639)

### 研究HTTPS

曾几何时，只记得HTTPS的端口和HTTP的不同，一个是443，一个是80。以前做项目也是只晓得用第三方的jar包，只晓得怎么生成证书和使用方法与流程，对原理并不是很清楚。这里接着上一篇[RSA算法](http://www.haomou.net/2014/08/27/2014_rsa/)，本篇就详细介绍HTTPS协议和相关的SSL/TLS加密协议。  
  
皓眸大前端开发学习  
  
  
转载请注明出处：<http://www.haomou.net/2014/08/30/2014_https/>

### 什么是HTTPS

[百科定义](http://baike.baidu.com/view/14121.htm?fr=aladdin)：  
HTTPS（全称：Hyper Text Transfer Protocol over Secure Socket Layer），是以安全为目标的HTTP通道，简单讲是HTTP的安全版。即HTTP下加入SSL层，HTTPS的安全基础是SSL，因此加密的详细内容就需要SSL。 它是一个URI scheme（抽象标识符体系），句法类同http\:体系。用于安全的HTTP数据传输。https\:URL表明它使用了HTTP，但HTTPS存在不同于HTTP的默认端口及一个加密/身份验证层（在HTTP与TCP之间）。这个系统的最初研发由网景公司(Netscape)进行，并内置于其浏览器Netscape Navigator中，提供了身份验证与加密通讯方法。现在它被广泛用于万维网上安全敏感的通讯，例如交易支付方面。  
HTTPS和HTTP的区别  
一、https协议需要到ca申请证书，一般免费证书很少，需要交费。  
二、http是超文本传输协议，信息是明文传输，https 则是具有安全性的ssl加密传输协议。  
三、http和https使用的是完全不同的连接方式，用的端口也不一样，前者是80，后者是443。  
四、http的连接很简单，是无状态的；HTTPS协议是由SSL+HTTP协议构建的可进行加密传输、身份认证的网络协议，比http协议安全。  
采用https的服务器必须从CA （Certificate Authority）申请一个用于证明服务器用途类型的证书。该证书只有用于对应的服务器的时候，客户端才信任此主机。所以所有的银行系统网站，关键部分应用都是https 的。客户通过信任该证书，从而信任了该主机。其实这样做效率很低，但是银行更侧重安全。这一点对局域网对内提供服务处的服务器没有任何意义。局域网中的服务器，采用的证书不管是自己发布的还是从公众的地方发布的，其客户端都是自己人，所以该局域网中的客户端也就肯定信任该服务器。  
数据的泄密和被篡改  
1． 一般意义上的https，就是服务器有一个证书。  
a) 主要目的是保证服务器就是他声称的服务器，这个跟第一点一样。  
b)服务端和客户端之间的所有通讯，都是加密的。  
i. 具体讲，是客户端产生一个对称的密钥，通过服务器的证书来交换密钥，即一般意义上的握手过程。  
ii. 接下来所有的信息往来就都是加密的。第三方即使截获，也没有任何意义，因为他没有密钥，当然篡改也就没有什么意义了。  
2． 少许对客户端有要求的情况下，会要求客户端也必须有一个证书。  
a) 这里客户端证书，其实就类似表示个人信息的时候，除了用户名/密码，还有一个CA 认证过的身份。因为个人证书一般来说是别人无法模拟的，所有这样能够更深的确认自己的身份。  
b) 目前大多数个人银行的专业版是这种做法，具体证书可能是拿U盘（即U盾）作为一个备份的载体。

### SSL/TLS由来

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。  
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。

### SSL介绍

SSL（Secure Sockets Layer，安全套接层），及其继任者 TLS（Transport Layer Security，传输层安全）是为网络通信提供安全及数据完整性的一种安全协议。TLS与SSL在传输层对网络连接进行加密。

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：  
a)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。  
b)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。

SSL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。

SSL或者Secure Socket Layer，是一种允许web浏览器和web服务器通过一个安全的连接进行交流的技术。这意味着将被发送的数据在一端被翻译成密码，传送出去，然后在另一端解开密码，再进行处理。这是一个双向的过程，也就是浏览器和服务器都需要在发送数据之前对它们进行加密。

SSL协定的另一个重要方面是认证(Authentication)。这就是说，在你开始试图通过一个安全连接与一个Web服务器交流的时候，这个服务器会要求你的浏览器出示一组证件，通过“鉴定”的方式来证明这就是你所声明的网站。在某些情况下，服务器还会要求你的web浏览器的认证书，证明你就是你所说的那个人。这就是所知的“客户认证”，尽管实际情况中，更多地用在商务-对-商务(B2B)交易，而不是对个人用户。大多数有SSL功能的web服务器不要求客户认证(Client Authentication)。

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  2  3 | （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）。

### 握手阶段详解

皓眸大前端开发学习

“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。

#### 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”会话密钥”(session secret)。  
（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。

#### 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. （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. （2） 一个服务器生成的随机数，稍后用于生成"会话密钥"(session secret)。
3. （3） 确认使用的加密方法，比如RSA公钥加密。
4. （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。

#### 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：

1. "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。
2. 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（Master Secret）最终导出一个对称密钥。
3. 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。

#### 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。  
皓眸大前端开发学习

###Secret Keys产生与使用  
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都是一代而过，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的的生成过程以及作用流程图：  
皓眸大前端开发学习  
reMaster 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。在客户端使用服务单的公钥对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[《Htttps SSL/TLS PreMaster/Master Secret(Key)计算》](http://www.fenesky.com/blog/2014/07/25/how-premaster-secret.html)。

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。Master secret最终解析出来的数据如下：  
皓眸大前端开发学习  
其中，write MAC key，就是session secret或者说是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MAC的工作原理详见[MAC](http://en.wikipedia.org/wiki/Message_authentication_code)。

关于Session Secret(Key)的计算请参考[《Htttps SSL/TLS Session Secret(Key)计算》](http://www.fenesky.com/blog/2014/07/25/how-session-secret.html)。

### 加密算法

加密算法有俩大类，第一种是不基于KEY的，另一种是不基于KEY的。

1. 基于KEY的，举个简单的例子，我要加密"fordesign"这么一串字符，就把每个字符都变成它的后一个字符，那么就是"gpseftjhm"了，这样的东西人家当然看不明白，接收方用相反的方法就可以得到原文。当然这只是个例子，现在应该没人用这么搞笑的加密算法了吧。
3. 不基于KEY，好象一直用到了计算机出现。我记得古中国军事机密都是用这种方式加密的。打战的时候好象军队那些电报员也要带着密码本，也应该是用这种方式加密的。这种算法的安全性以保持算法的保密为前提。
4. 这种加密算法的缺点太明显了，就是一旦你的加密算法给人家知道，就肯定挂。日本中途岛惨败好象就是密码给老米破了。设计一种算法是很麻烦的，一旦给人破了就没用了，这也忑浪费。
6. 我们现在使用的加密算法一般是基于key的，也就是说在加密过程中需要一个key，用这个key来对明文进行加密。这样的算法即使一次被破，下次改个key，还可以继续用。key是一个什么东西呢？随便你，可以是一个随机产生的数字，或者一个单词，啥都行，只要你用的算法认为你选来做key的那玩意合法就行。
8. 这样的算法最重要的是：其安全性取决于key，一般来说取决于key的长度。也就是说应该保证人家在知道这个算法而不知道key的情况下，破解也相当困难。其实现在常用的基于KEY的加密算法在网络上都可以找到。

基于key的加密算法又包括俩类：对称加密和不对称加密。

1. 对称加密指的是双方使用完全相同的key， 最常见的是DES. DES3, RC4等。对称加密算法的原理很容易理解，通信一方用KEK加密明文，另一方收到之后用同样的KEY来解密就可以得到明文。
3. 不对称加密指双方用不同的KEY加密和解密明文，通信双方都要有自己的公共密钥和私有密钥。
4. 举个例子比较容易理解， 我们们假设通信双方分别是A、B。
5. A 拥有 KEY\_A1、KEY\_A2, 其中KEY\_A1是A的私有密钥，KEY\_A2是A的公共密钥。
6. B 拥有 KEY\_B1、KEY\_B2, 其中KEY\_B1是B的私有密钥，KEY\_B2是B的公共密钥。
7. 公共密钥和私有密钥的特点是，经过其中任何一把加密过的明文，只能用另外一把才能够解开。也就是说经过KEY\_A1加密过的明文，只有KEY\_A2才能够解密，反之亦然。

通信过程如下：

1. A-------->KEY\_A2------------>B
2. A<--------KEY\_B2<------------A
4. 这个过程叫做公共密钥交换，老外管这叫key exchange。
5. 之后A和B就分别用对方的公共密钥加密，用自己的私有密钥解密。
6. 一般公共密钥是要发布出去的，然后你通过自己的私有密钥加密明文，人家用你的公共密钥解密，如果能解开，那么说明你是加密人，这就是SSL使用的验证机制。

我如果要发给你，我就用你的公钥加密，这让只有你能用自己的私钥解密。如果我用自己的私钥加密，然后发布出去，这就不属于保密，而是属于认证，证明这条信息是我发的，而且我不能赖帐，因为私钥只有我自己知道。

常用的不对称加密一般有RSA、 DSA、 DH等。我们一般使用RSA。

### 数字签名

数字签名也是不对称加密算法的一个重要应用，理解它对于理解SSL很重要的，放在这里一起介绍一下。

签名是什么大家都很熟悉吧？证明该东西是你写的，是你发布的，你就用签名搞定。看看那些重要文件都要头头签名。数字签名就是数字化的签名了。记得公用密钥和私有密钥的特征吗？只有你一个人有你自己的私有密钥。而你的公用密钥是其他人都知道的 了。那么你在写完一封邮件之后，用自己的私有密钥加密自己的名字，接收人用你的公共密钥解开一看，哦，是你发的。这就是你的数字签名过程了。

上面的解释是很简化的了，其实数字签名比这个复杂多了，但我们没有了解的必要，知道数字签名是这么一回事就可以了。

还有一种我们需要知道的加密算法，其实我不觉得那是加密算法 ，应该叫哈希算法，英文是message digest, 是用来把任何长度的一串明文以一定规则变成固定长度的一串字符串。它在SSL中的作用也很重要，以后会慢慢提及的。一般使用的是MD5、SHA、base64不是加密算法，但也是SSL经常使用的一种算法，它是编码方式，用来把asc码和二进制码转来转去的。

具体的加密解密过程我们不需要了解，因为SSL根本不关心。但了解加密算法的一些基本原理是必要的，否则很难理解SSL。

### 关于证书

为了能实施SSL，一个web服务器对每个接受安全连接的外部接口(IP地址)必须要有相应的认证书(Certificate)。关于这个设计的理论是一个服务器必须提供某种合理的保证以证明这个服务器的主人就是你所认为的那个人，特别是在接收任何敏感信息之前要这样做。关于Certificates的更广泛的解释超过了这个文档资料的范围，就把一个认证书当作一个英特网地址的“数码驾驶执照”。这个认证书要陈述与这个网站相关联的公司，以及这个网站的所有者或系统管理员的一些基本联系信息。

这个”驾驶执照”由所有人以密码方式签字，其他人非常难伪造。对于进行电子商务 (e-commerce)的网站，或其他身份认证至关重要的任何商业交易，认证书要向大家所熟知的认证权威 (Certificate Authority (CA))如VeriSign或Thawte来购买。这样的认证书可用电子技术证明属实——实际上，认证权威单位会担保它发出的认证书的真实性，如果你信任发出认证书的认证权威单位的话，你就可以相信这个认证书是有效的。

从技术上说，SSL 证书（也称为数字证书）将身份与一对可用于加密和签名数字信息的电子密钥绑定。SSL 证书能够实现对某人自称有权使用特定密钥的声明的验证，有助于防止有人使用欺骗性密钥来模拟其他用户。当与加密配合使用时，SSL 证书可提供完整的安全解决方案，可以保证参与事务的一方或各方的身份。

SSL 证书是由受信任的第三方（称为证书颁发机构 (CA)）发放的。CA 的作用有些像护照办理处。CA 必须采取一些措施来确定要向其发放 ID 的人或组织的身份。一旦 CA 建立某个组织的身份后，就可以发出一个包含该组织的公钥的证书，并用 CA 的私钥对其签名。

通过使用 SSL 证书，您就能在自己的站点上开展验证身份的、加密的在线商务活动。访问您站点的用户将可以向站点提交信用卡号或其他个人信息，从而保证他们真心实意与您进行业务交易（并非是骗子），并且他们发送给您的信息不会被目标接收者以外的任何人截取或解密。您的 SSL 证书将包含下列信息：

您的组织的公用名称（如 www.bea.com）  
1)其它标识性信息（如 IP 和物理地址）  
2)您的公钥  
3)公钥的到期日期  
4)发出此 ID 的 CA 的名称（如 VeriSign）  
5)一个唯一的序列号  
6)VeriSign 的数字签名

主要的证书类型有：PEM、DER、PKCS#12  
PEM  
可以包括所有私钥（RSA 和 DSA）、公钥（RSA 和 DSA）和 (x509) 证书。它存储用 Base64 编码的 DER 格式数据，用 ascii 报头包围，因此适合系统之间的文本模式传输。  
—–BEGIN CERTIFICATE—–  
MIICJjCCAdCgAwIBAgIBITANBgkqhkiG9w0BAQQFADCBqTELMAkGA1UEBhMCVVMx  
EzARBgNVBAgTCkNhbGlmb3JuaWExFjAUBgNVBAcTDVNhbiBGcmFuY2lzY28xFTAT  
BgNVBAoTDEJFQSBXZWJMb2dpYzERMA8GA1UECxMIU2VjdXJpdHkxIzAhBgNVBAMT  
GkRlbW8gQ2VydGlmaWNhdGUgQXV0aG9yaXR5MR4wHAYJKoZIhvcNAQkBFg9zdXBw  
b3J0QGJlYS5jb20wHhcNMDAwNTMwMjEzODAxWhcNMDQwNTEzMjEzODAxWjCBjDEL  
MAkGA1UEBhMCVVMxEzARBgNVBAgTCkNhbGlmb3JuaWExFjAUBgNVBAcTDVNhbiBG  
cmFuY2lzY28xFTATBgNVBAoTDEJFQSBXZWJMb2dpYzEZMBcGA1UEAxMQd2VibG9n  
aWMuYmVhLmNvbTEeMBwGCSqGSIb3DQEJARYPc3VwcG9ydEBiZWEuY29tMFwwDQYJ  
KoZIhvcNAQEBBQADSwAwSAJBALdsXEHqKHgs6zj0hU5sXMAUHzoT8kgWXmNkKHXH  
79qbPh6EfdlriW9G/AbRF/pKrCQu7hhllAxREbqTuSlf2EMCAwEAATANBgkqhkiG  
9w0BAQQFAANBACgmqflL5m5LNeJGpWx9aIoABCiuDcpw1fFyegsqGX7CBhffcruS  
1p8h5vkHVbMu1frD1UgGnPlOO/K7Ig/KrsU=  
—–END CERTIFICATE—–  
DER  
辨别编码规则 (DER) 可包含所有私钥、公钥和证书。它是大多数浏览器的缺省格式，并按 ASN1 DER 格式存储。它是无报头的 － PEM 是用文本报头包围的 DER。

PKCS#12  
公钥加密标准 #12 (PKCS#12) 可包含所有私钥、公钥和证书。其以二进制格式存储，也称为 PFX 文件。

###参考链接  
MicroSoft TechNet, [SSL/TLS in Detail](http://technet.microsoft.com/en-us/library/cc785811(v=ws.10).aspx )  
Jeff Moser, [The First Few Milliseconds of an HTTPS Connection](http://www.moserware.com/2009/06/first-few-milliseconds-of-https.html)  
Wikipedia, [Transport Layer Security](http://en.wikipedia.org/wiki/Transport_Layer_Security)  
StackExchange, [How does SSL work?](http://security.stackexchange.com/questions/20803/how-does-ssl-work)

### 谢谢！

转载请注明出处：<http://www.haomou.net/2014/08/30/2014_https/>

有问题请留言。T\_T 皓眸大前端开发学习 T\_T