<https://www.jianshu.com/p/33f43ca8f062>

# Chrome是如何实现TLS协议的

[](https://www.jianshu.com/u/2636ba4b7d36)

[咖啡不加盐0915](https://www.jianshu.com/u/2636ba4b7d36) 关注

2017.09.28 16:54\* 字数 2397 阅读 498评论 0喜欢 4

说起TLS（Transport Layer Security) 协议大家可能不是那么熟悉，但是说起HTTPS协议，大家肯定都或多或少听过。记得之前有个梗，要把服务器的应用层协议从HTTP切到HTTPS，有人说了，加个“S”不就可以了，一句话的事情……，实际上我一开始我也以为HTTPS只不过比HTTP多了一个“S”而已，这句话还真毛病，但并不是简单的在后面加个“S”就完事了，实际上这里的“S”就是我们今天要介绍的TLS协议。

TLS协议的前身是SSL协议，后者的全称是 Socket Security Layer，可以看到两者都涉及到了安全，实际上TLS协议的目标就是解决以下几个问题：

**\*\*信息被窃听(wiretap)，第三方随时随地获得通讯内容；**

**SSL/TLS 实现了传输信息的加密。**

**数据被篡改(tampering)，第三方可修改传输中的数据；**

**SSL/TLS 实现了数据签名及校验。**

**身份被冒充(pretending)，第三方可冒充通讯者身份传输数据；**

**SSL/TLS 采用了CA数字证书认证机制。\*\***

我们今天就从理论层面、生产环节下的实际表现以及chromium的代码实现这三个维度来详细地分析TLS协议，看看它究竟是如何实现上面这三方面的。

关于TLS协议的过程，大家肯定都知道TCP的三次握手，TLS也有类似的握手过程，和TCP为了建立稳定连接才进行握手类似，TLS协议也要进行客户端与服务端之间的通信，而通信的内容，就是在整个TLS协议中最为重要的密钥。这里的密钥是客户端和服务端动态生成，然后通过信息交互的方式保持一致，在这里涉及到很多安全学和密码学的知识，例如各种加密算法，动态密钥算法等等，有兴趣的同学可以去关注一下这方面的知识，在这里就不再赘述了。大致的握手流程如下：

客户端发出一个 client hello 消息，携带的信息包括：

所支持的SSL/TLS 版本列表；支持的加密算法；所支持的数据压缩方法；客户端生成的随机数A;

服务端响应一个 server hello 消息，携带的信息包括：

协商采用的SSL/TLS 版本号；会话ID；服务器端生成的随机数B；服务端数字证书 serverCA；

由于双向认证需求，服务端需要对客户端进行认证，会同时发送一个 client certificate request，表示请求客户端的证书；

客户端校验服务端的数字证书；校验通过之后发送随机数C，该随机数称为pre-master-key，使用数字证书中的公钥加密后发出；

由于服务端发起了 client certificate request，客户端使用私钥加密一个随机数 clientRandom随客户端的证书 clientCA一并发出；

服务端校验客户端的证书，并成功将客户端加密的随机数clientRandom 解密；

根据随机数A/随机数B/随机数C(pre-master-key) 产生动态密钥 master-key，加密一个finish 消息发至客户端；

客户端根据同样的随机数和算法 生成master-key，加密一个finish 消息发送至服务端；

服务端和客户端分别解密成功，至此握手完成，之后的数据包均采用master-key进行加密传输。

下面这张图更为直观地将整个过程表现了出来：

可以看到，整个过程至少需要两个RTT(往返时延)才可以完成认证，也就是说至少要两个RTT以后客户端才可以向服务端发送请求，如果再加上TCP三次握手用去的1个RTT，那么一共就要3个RTT后才可以进行数据的传输，这显然大大降低了传输效率，因此TLS协议也在不断地优化其速度，这也是未来TLS协议的一个重要研究方向，后续我们讲到优化的时候会着重再讲这部分的。

相信理论层的知识，通过看图并结合文字描述，大多数人都可以明白基本的道理，当然也仅限于浅显的理论，拉开来讲每部分都很需要花费一定时间才能理解和掌握，这些都留给后续的章节再来细说，下面我们就看看在实际的生产环境中TLS协议是如何工作的。

先介绍一下我们的帮手，Wireshark，一款非常著名的抓包软件，通过它我们便可以抓取一些我们想要的信息，然后再加以分析。关于软件的下载安装和使用，大家直接上网搜一下，这里不再赘述，下面我们登陆一个使用了https协议的网站，直接用Wireshark进行抓包，得到的部分报文如下：

这就是一个典型的三次握手过程，客户端发送SYN报文，服务端受到后发SYN ACK报文，然后客户端发相应的ACK回应报文，想学习TCP的同学强烈建议你们去试一下，绝对比你看书学得更快，更牢固。

接下来我们就来看一下TLS协议的交换，

直接将所有的TLS报文过滤出来，可以看到，首先是客户端发Client Hello报文，然后是服务端接收并且发送Server Hello报文，同时我们注意到，服务器端紧接着又发送了两个报文，分别是Certificate报文和Server Key报文，这就是前文中提到的一些已经经过加密的证书信息，当客户端收到这些证书以后，则发送相应的prr-master-key，服务端在接收到这个key以后，生成相应的master-key值，这样，一个完整的TLS握手协议过程就算完成了，后续就可以看到客户端和服务器之间已经开始用TLS传输加密后的数据了。

抓取其中的一个报文来看一下：

可以看到TCP下面就是SSL，在这里我们抓取的是Client Hello报文，也就是客户端发起握手协议后的第一个报文，可以看到这个报文包含了不少字段，首先是Content Type字段，这里的的类型是Handshake,也就是握手协议，还有紧接着跟的是使用的TLS协议版本号以及数据包的长度。

接下来便是Handshake层的具体内容了，几个主要字段的含义和相应的赋值对应关系：

Handshake Type ：Client Hello，在这里表示握手协议的类型，一般有这几种类型值：

0) hello\_request:

1) client\_hello:

2) server\_hello:

4) certificate:

5) server\_key\_exchange:

6) certificate\_request:

7) server\_done:

8) certificate\_verify:

9) client\_key\_exchange:

10) :

Length ： 508 表示这个握手协议的总长度，注意到TLS的总长度是512，如果除去TLS头内容的¥字节，刚好就是508

Version：TLS 1.2 要注意这里的Version是代表客户端可以支持的最高协议版本号，与之前的Version代表的客户端正在使用的版本号是有区别的。

Random: 客户端生成的一个随机数 这就是我们之前提到的客户端会生成一个随机数，然后再发送到服务端，作为生成master-key的依据。

除了Handshake之外，TLS报文还有另外3种类型，分别是:

警告层(Alert): 如果在通信过程中某一方发现任何异常，就需要给对方发送一条警示消息通告。

改变密码格式层（Client Key Exchange): SSL协议要求客户端或服务器端每隔一段时间必须改变其加解密参数。当某一方要改变其加解密参数时，就发送一个简单的消息通知对方下一个要传送的数据将采用新的加解密参数

应用数据层(Application Data): 进行加密后的数据传输

通过抓包并分析具体报文的构成，对于TLS报文的构成应该有一个更为直观的了解，其中最为关键也是最复杂的就是握手的过程，以后我们会对这部分内容做更为深入的分析。

讲完了TLS的理论基础，也看到了其在实际生产环境中一些应用，相信对于这个协议也有一个大致的了解，后续还会讲到chromium对于TLS的代码实现，敬请期待。

**小礼物走一走，来简书关注我**