HTTPS 协议原理分析

HTTPS 协议需要解决的问题

HTTPS 作为安全协议而诞生，那么就不得不面对以下两大安全问题：

身份验证：确保通信双方身份的真实性。直白一些，A 希望与 B 通信，A 如何确认 B 的身份不是由 C 伪造的。（由 C 伪造 B 的身份与 A 通信，称为中间人攻击）

通信加密：通信的机密性、完整性依赖于算法与密钥，通信双方是如何选择算法与密钥的。

能同时解决以上两个问题，就能确保真实有效的通信双方采取有效的算法与密钥进行通信，便完成了协议安全的初衷。

在介绍 HTTPS 协议如何解决两大安全问题前，我们首先了解几个概念。

数字证书：数字证书是互联网通信中标识双方身份信息的数字文件，由 CA 签发。

CA：CA（certification authority）是数字证书的签发机构。作为权威机构，其审核申请者身份后签发数字证书，这样我们只需要校验数字证书即可确定对方的真实身份。

HTTPS 协议、SSL 协议、TLS 协议、握手协议的关系

HTTPS 是 Hypertext Transfer Protocol over Secure Socket Layer 的缩写，即 HTTP over SSL，可理解为基于 SSL 的 HTTP 协议。HTTPS 协议安全是由 SSL 协议（目前常用的，本文基于 TLS 1.2 进行分析）实现的。

SSL 协议是一种记录协议，扩展性良好，可以很方便的添加子协议，而握手协议便是 SSL 协议的一个子协议。

TLS 协议是 SSL 协议的后续版本，本文中涉及的 SSL 协议默认是 TLS 协议 1.2 版本。

HTTPS 协议的安全性由 SSL 协议实现，当前使用的 TLS 协议 1.2 版本包含了四个核心子协议：握手协议、密钥配置切换协议、应用数据协议及报警协议。

解决身份验证与通信加密的核心 ,便是握手协议 ,接下来着重介绍握手协议。

握手协议

握手协议的作用便是通信双方进行身份确认、协商安全连接各参数（加密算法、密钥等），确保双方身份真实并且协商的算法与密钥能够保证通信安全。

对握手协议的介绍限于客户端对服务端的身份验证 ,单向身份验证也是目前互联网公司最常见的认证方式。

首先我们看一下协议交互，如图 1 所示：

图 1 握手协议

接下来以 Wireshark 抓取接口的握手协议过程为例，针对每条协议消息分析。

ClientHello **消息**

ClientHello 消息的作用是，将客户端可用于建立加密通道的参数集合，一次性发送给服务端。

消息内容包括：期望协议版本(TLS 1.2)、可供采用的密码套件(Cipher Suites)、客户端随机数(Random)及扩展字段内容(Extension)等信息，如图 2 所示。

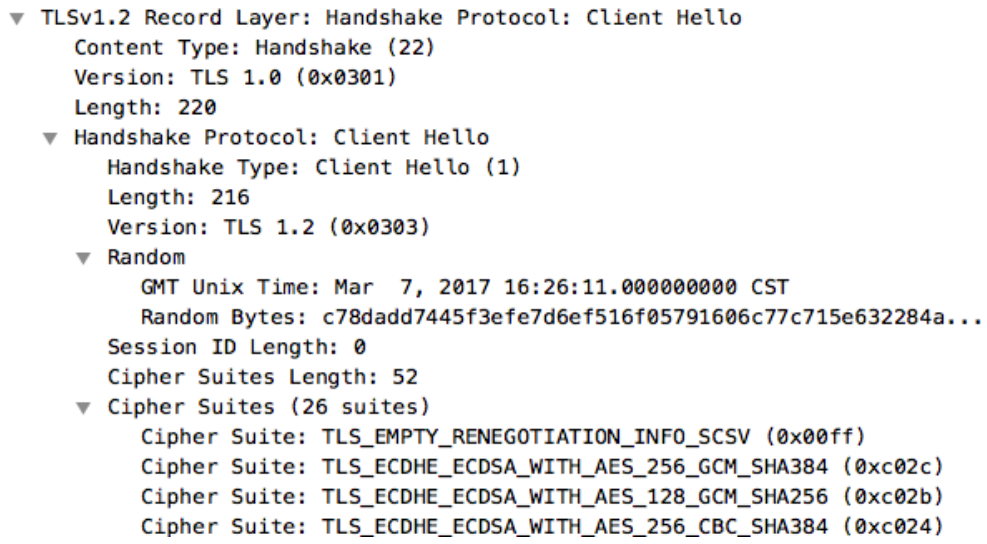

图 2 ClientHello

ServerHello **消息**

ServerHello 消息的作用是，在 ClientHello 参数集合中选择适合的参数，并将服务端用于建立加密通道的参数发送给客户端。

消息内容包括：采取的协议版本(TLS 1.2)、采用的密码套件(Cipher Suite)、服务端随机数(Random)、用于恢复会话的会话 ID(Session ID)及扩展字段等信息，如图 3 所示。

自此客户端与服务端的协议版本、密码套件已经协商完毕。

这里服务端下发的会话 ID 可用于后续恢复会话。若客户端在 ClientHello 中携带了会话 ID，并且服务端认可，则双方直接通过原主密钥生成一套新的密钥即可继续通信。将两个网络往返降低为一个网络往返，提高通道建立的效率。

图 3 ServerHello

Certificate **消息**

Certificate 消息的作用是，将服务端证书的详细信息发送给客户端，供客户端进行服务端身份校验。

消息内容：服务端下发的证书链，如图 4 所示。

服务端为了保证下发的证书能够被客户端正确识别，就需要将签发此证书的 CA 证书一同下发，构成证书链，保证客户端可以根据证书链的信息在系统配置中找到根证书，并通过根证书的公钥逐层向下验证证书的合法性。

如图所示，五八服务器下发了两个证书：自己的证书与签发 CA 的证书。通过签发 CA 的证书信息，能够直接找到根证书。

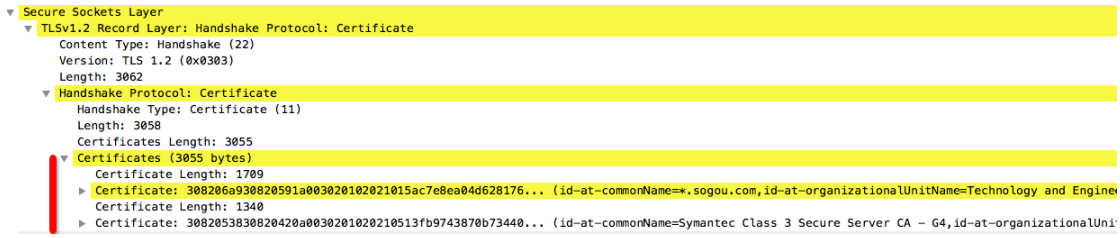

图 4 Certificate

客户端本地校验服务端证书，若校验通过，则客户端对服务端的身份验证便完成了。

Certificate 这个阶段解决了两端的身份验证问题。借助 CA 的力量，通过 CA 签发证书，将身份验证的工作交给了 CA 处理。

只要是我们认可的 CA，签发的证书我们均认可证书持有者的身份。由于 CA 的介入，解决了中间人攻击的问题，因为中间人并没有服务端的证书可供客户端验证。

ServerKeyExchange **消息(可能不发送)**

ServerKeyExchange 消息的作用是，将需要服务端提供的密钥交换的额外参数，传给客户端。有的算法不需要额外参数，则 ServerKeyExchange 消息可不发送。

消息内容：用于密钥交换的额外参数，如图 5 所示。

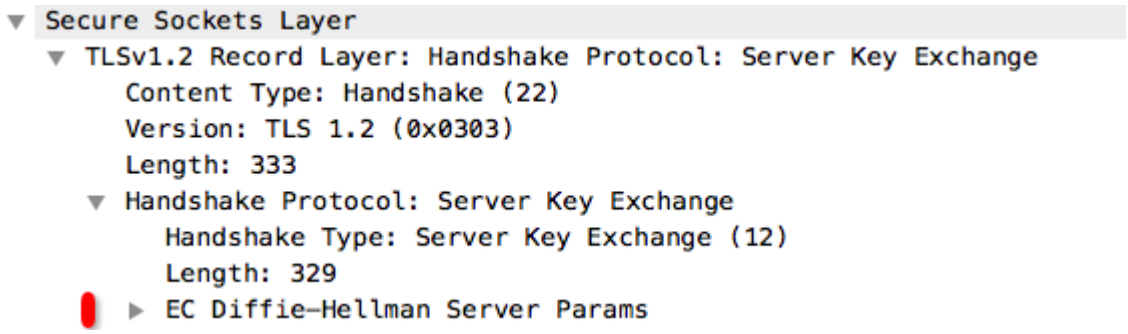

图 5 ServerKeyExchange

如图 5，服务端下发了“EC Diffile-Hellman”密钥交换算法所需要的参数。

ServerHelloDone **消息**

ServerHelloDone 消息的作用是，通知客户端 ServerHello 阶段的数据均已发送完毕，等待客户端下一步消息。

ClientKeyExchange **消息**

ClientKeyExchange 消息的作用是，将客户端需要为密钥交换提供的数据发送给服务端。

当我们选用 RSA 密钥交换算法时，此消息的内容便是通过证书公钥加密的用于生成主密钥的预主密钥。

如图 6 所示，由于选用的密钥交换算法是“EC Diffie-Hellman”，所以 ClientKeyExchange 消息发送的是”EC Diffie-Hellman”算法需要的客户端参数。

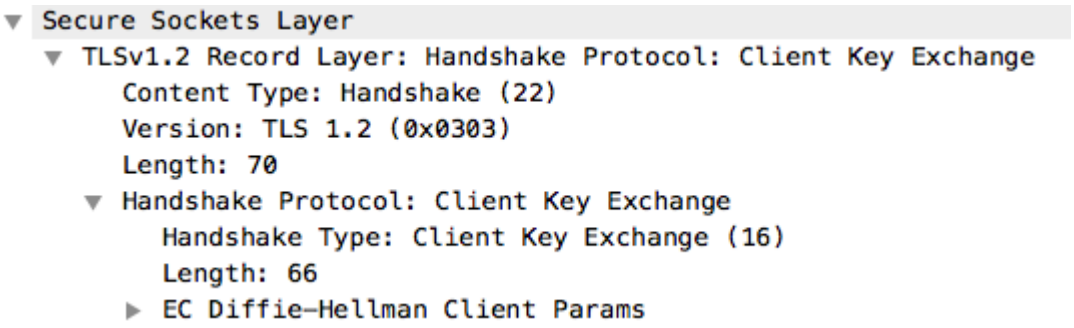

图 6 ClientKeyExchange

当发送了 ClientKeyExchange 后，两端均具有了生成主密钥的完整密钥数据与随机数，两端分别根据所选算法计算主密钥即可。

至此，ClientKeyExchange 发送后，两端均可生成主密钥，密钥交换问题便解决了。

有的读者可能对随机数的采用有些疑惑，笔者觉得随机数的加入是为了提高密钥的随机性。

由于客户端直接生成的密钥很有可能不够随机，而通过预主密钥加上两端提供的两个随机数做种子，创建的主密钥可以保证更加贴近真实随机的密钥。

ChangeCipherSpec 消息

经过以上六条消息，我们已经解决了身份认证问题、密码套件选取问题、密钥交换问题。双方也已经通过主密钥生成了实际使用的六个加解密密钥。

ChangeCipherSpec 消息的作用，便是声明后续消息均采用密钥加密。在此消息后，我们在 WireShark 上便看不到明文信息了。

Finished 消息

Finished 消息的作用，是对握手阶段所有消息计算摘要，并发送给对方校验，避免通信过程中被中间人所篡改。

HTTPS 协议总结

自此，HTTPS 如何保证通信安全，通过握手协议的介绍，我们已经有所了解。

但是，在全面使用 HTTPS 前，我们还需要考虑一个众所周知的问题——HTTPS 性能。

相对 HTTP 协议来说，HTTPS 协议建立数据通道的更加耗时，若直接部署到 App 中，势必降低数据传递的效率，间接影响用户体验。