wireshark抓包分析HTTPS

白帽青年
不以物喜，不以己悲
关注他

一、名词解释

TLS

安全传输层协议（TLS）用于在两个通信应用程序之间提供保密性和数据完整性。该协议由两层组成： TLS 记录协议（TLS Record）和 TLS 握手协议（TLS Handshake）。

记录协议主要负责使用对称密码对消息进行加密；握手协议分为握手协议，密码规格变更协议、警告协议和应用数据协议4个部分

RTT

`RTT(Round Trip Time)`：一个tcp连接的往返时间，即数据发送时刻到接收到确认的时刻的差值；

密码套件

```
Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 (0xc02b)
```

TLS：指使用的协议是TLS

ECDHE：密钥交换算法

ECDSA：签名或验证算法

AES_128_GCM：对称加密算法。对称密钥加密算法是AES，强度即密钥长度为128位，GCM是工作模式，AES是**块密码**，也就是对输入的纯文本用固定长度的块来进行加密，加密后的每个块按再顺序发送，最后按类似的方式来

SHA256：签名哈希算法

Diffie Hellman 密钥协商算法

使用Diffie Hellman算法进行TLS密钥交换具有优势。客户端和服务器都为每个新会话生成一个新密钥对。一旦计算出预主密钥，将立即删除客户端和服务器的私钥。这意味着私钥永远不会被窃取，确保完美的前向保密。

如果使用RSA，则客户端将如上所述通过其自己计算预主密钥，使用服务器的公钥（RSA公钥）对其进行加密，并通过客户端密钥交换消息将其发送回服务器。然后，服务器可以使用其私钥解密它。

二、TLS 握手原理：

Client Hello

1.1、Client Hello 报文：客户端对加密算法的支持度不同，因此需要向服务端发送客户端支持的加密套件（Cipher Suite），同时还要生成一个 随机数 同时保存在客户端和发送给服务

Server Hello

2.1、Server Hello 报文：服务端收到 Client Hello 之后，生成一个 随机数 同时保存在服务端和发送给客户端

2.2、Server Certificate 报文：向客户端发送 CA 认证的数字证书，用来鉴别服务端身份信息

2.3、Server Hello Done 报文：表示服务端宣告第一阶段的客户端服务端握手协商结束

2.4、可选：Certificate Request 报文：必要情况下，要求客户端发送证书验证身份

2.5、可选：Server Key Exchange 报文：仅当服务器提供的证书不足以允许客户端交换预主密钥时，才会发送此消息

Client Finish

3.1、Client Key Exchange 报文：客户端收到 CA 数字证书并通过验证，获取服务端公钥。Client Key Exchange 报文包括有一个随机数，这个随机数被称为 Pre-master key/secret；一个表示随后的信息使用双方协商好的加密方法和密钥发送的 通知

3.2、Client Cipher Spec 报文：该报文通知服务端，此后的通信都将使用协商好的加密算法计算对称密钥进行加密通信（也就是使用两个随机数以及第三个 Pre-master key/secret 随机数一起算出一个对称密钥 session key/secret）

3.3、Finished 报文：该报文包括连接至此的所有报文的校验值，使用服务端公钥进行加密

3.4、可选：Client Certificate 报文：如果服务端请求，客户端需要发送 CA 数字证书

3.5、可选：Certificate Verify 报文：服务端如果要求 CA 数字证书，那么需要通过 HASH 算法计算一个服务端发送来的信息摘要

Server Finish

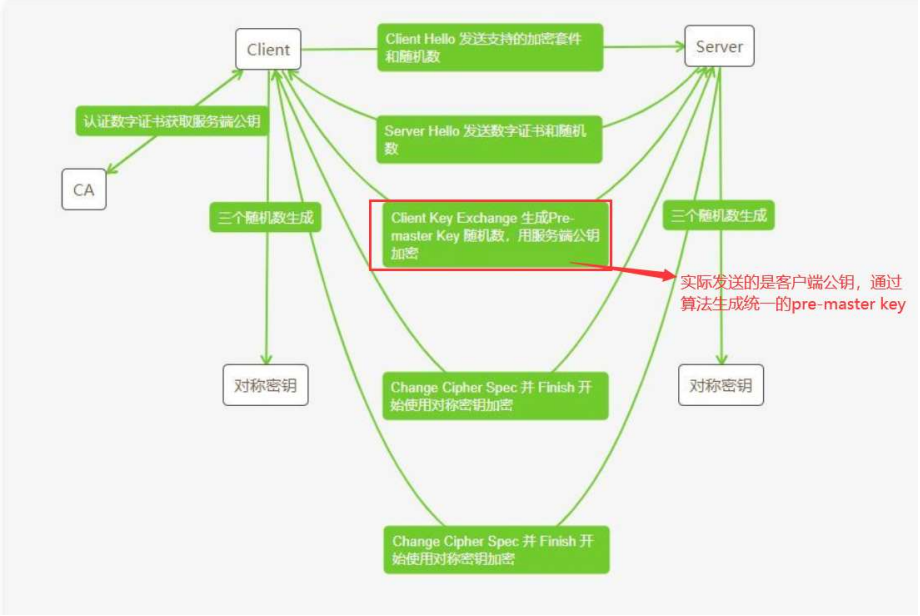
4.1、服务端最后对客户端发送过来的 Finished 报文使用服务端私钥进行解密校验

4.2、Client Cipher Spec 报文：报文通知服务端，此后的通信都将使用协商好的加密算法计算对称密钥 session key/secret 进行加密通信

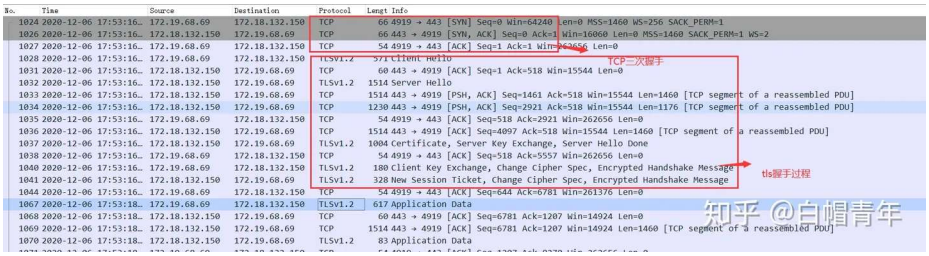
4.3、Finished 报文：标志 TLS 连接建立成功

TLS 握手成功，此后通过对称密

大致图解：



三、TLS 1.2 握手过程抓包：

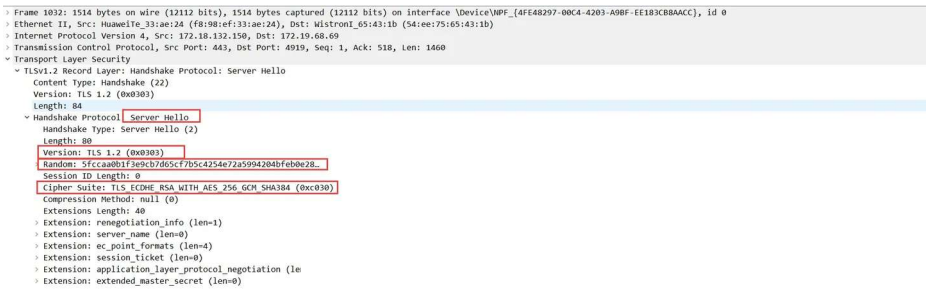


Client Hello：

客户端发送32位随机数、所支持的加密套件、session id（用于协商会话关闭后，重新打开时需不需要再次握手）、extension（添加一些拓展功能，tls1.3使用）；compress（支持的压缩方法）

Server Hello：

服务端给客户端发送一个32位随机数，以及选中的 Cipher Suite 加密算法，和tls版本

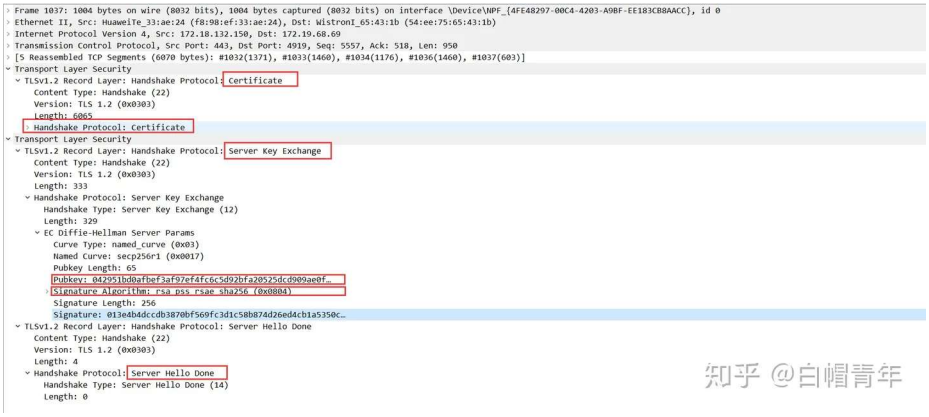


Certificate、Server Key Exchange、Server Hello Done：

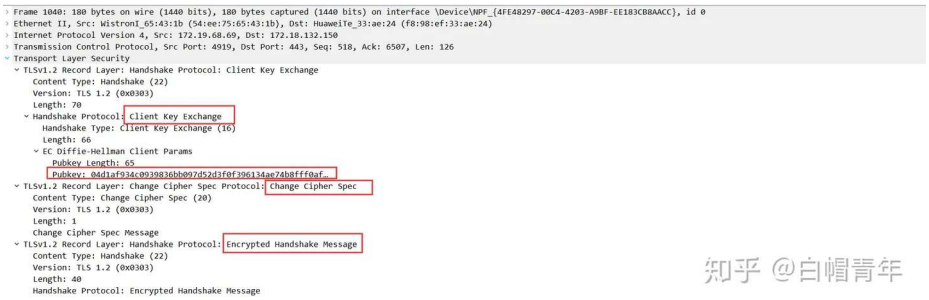
certificate：该过程中服务器用私钥签名证书，发送给客户端以认证身份

server key exchange：由于服务端选择了Elliptic Curve Diffie Hellman算法交换密钥，在此过程服务端将生成一对DH公钥和私钥，私钥保留（用于服务器端生成预主密钥（Pre-master key）），并将公钥发送给客户端（用于客户端生成预主密钥），同时将前一阶段所有的会话内容利用私钥进行签名发给客户端，用于验证服务端身份，防止中间人攻击。有没有这一过程都是却决于密钥交换算法自身。在这个数据包中，还给出了服务端生成公私钥所用的算法 sec256r1

server hello done：表示server hello结束，这是个空消息



Client Key Exchange、Change Cipher Spec、Encrypted Handshake Message：



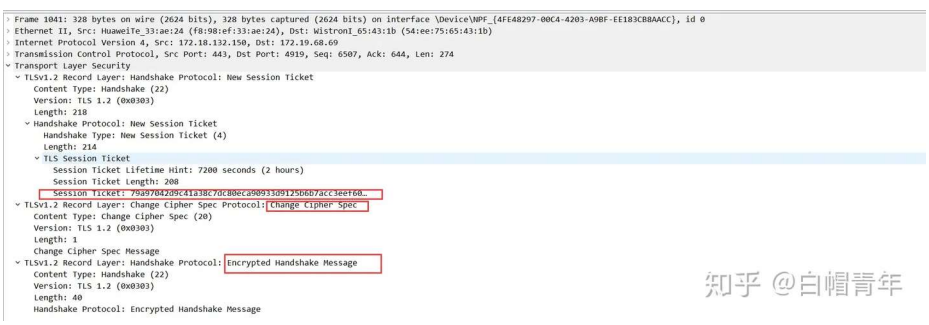
client key exchange：客户端也生成一对DH公钥和私钥，私钥保留（用于客户端生成预主密钥），公钥发给服务端（用于服务端生成预主密钥（Pre-master key））

此时客户端在本地计算出预主密钥Pre-master key，由于算法特性，客户端和服务端根据各自的DH密钥计算的结果是相等的，并通过Pre-master key计算出**主密钥**

change cipher spec：客户端根据交互过程中获得的信息，以及应用服务端规定的密码套件，已经生成了相应的密钥。通过这条消息，客户端告诉服务器端：从现在起，我将使用双方约定的密码规范进行通信

encrypted handshake message：客户端利用生成的密钥加密一段finishde数据传送给服务端，此数据是为了在正式传输应用之前对刚刚握手建立起来的加解密通道进行验

New Session Ticket、Change Cipher Spec、Encrypted Handshake Message：



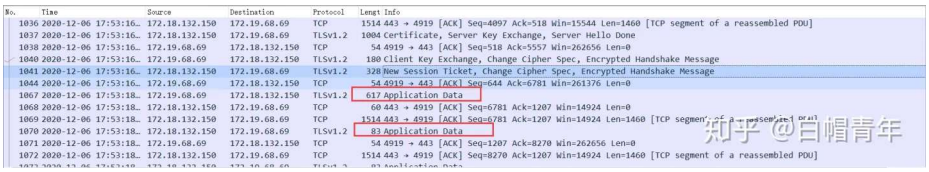
服务端在本地计算出预主密钥Pre-master key，由于算法特性，客户端和服务端根据各自的DH密钥计算的结果是相等的，并通过Pre-master key计算出**主密钥**

new session ticket：服务端告知客户端将生成新的session ticket用于保持会话（session ticket与前面提到的session id作用类似，但两者实现方式不同）

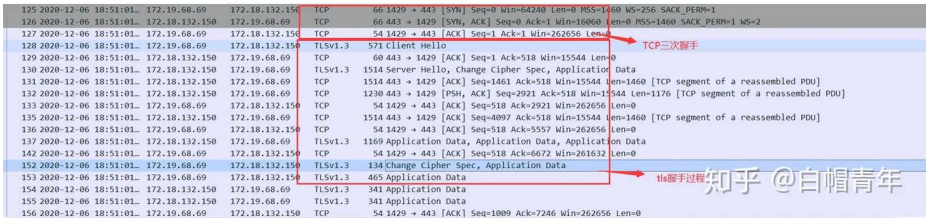
change cipher spec：通过这条消息，服务端告诉客户端：从现在起，我将使用双方约定的密码规范进行通信

encrypted handshake message：作用与客户端一致，至此，握手协议结束，双方开始建立加密通道。

可观察到后续的报文都是经过加密，表明已使用**对称密钥进行**加密传输

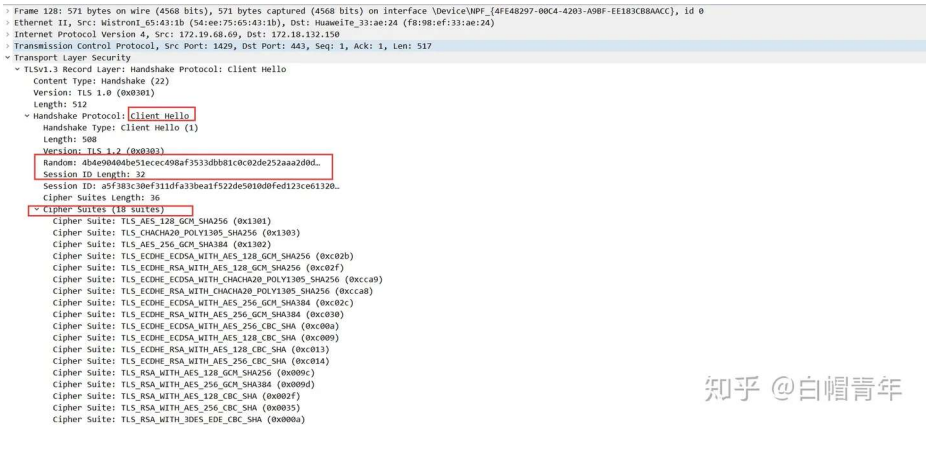


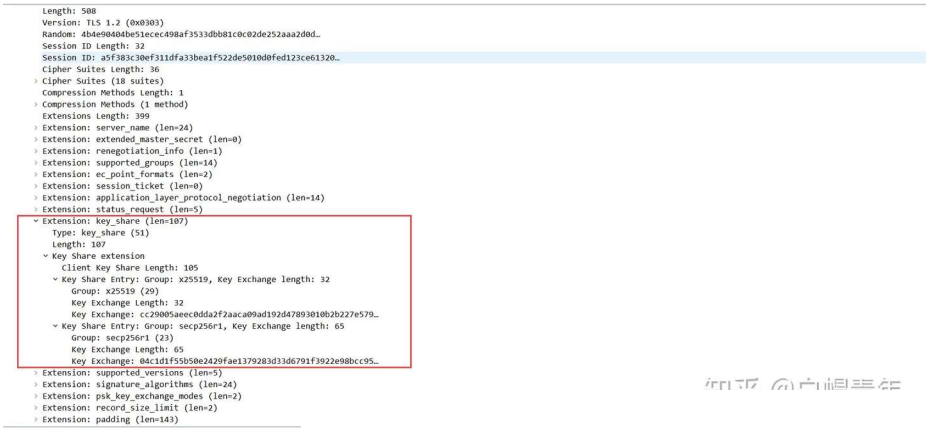
四、TLS 1.3握手过程抓包



Client Hello：

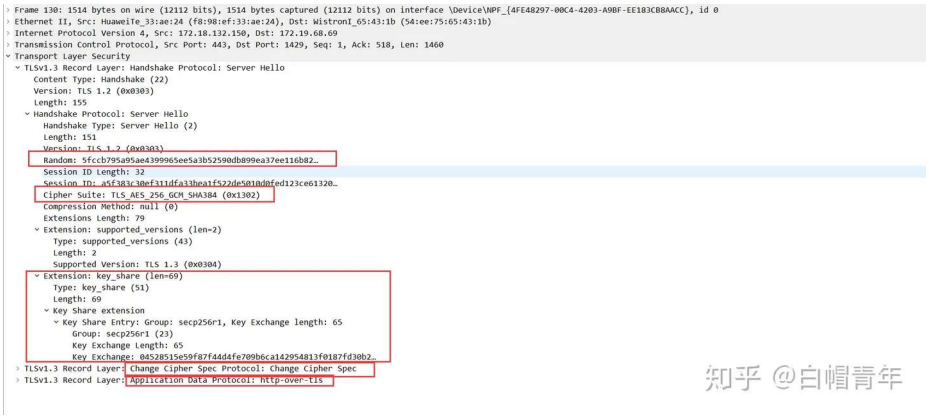
客户端发送32位随机数、所支持的加密套件、session id、compress（支持的压缩方法）、并预先使用一些安全套件，生成客户端公、私钥，将公钥放在key_share中发送给服务端。



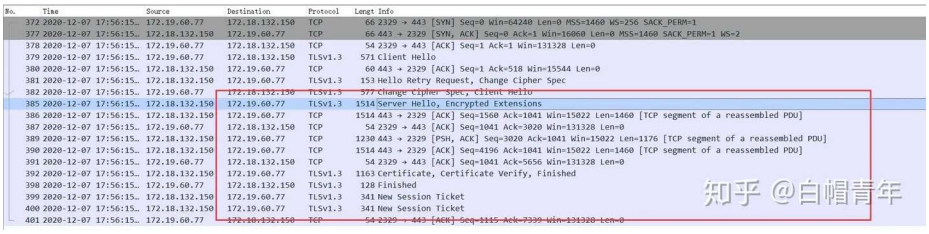


Server Hello 、Change Cipher Spec、Application Data

此时服务端已具备足够信息生成预主密钥，进而生成主密钥，因此会发送 `Change Cipher Spec` 告知客户端，之后的消息将会加密传输；此后所有的报文都将被加密成Application Data（由上述握手过程可见）



使用wireshark解密后的数据包



此后客户端接收到服务端创建的公钥和随机数后，根据对应算法计算出预主密钥和主密钥，进行加密消息验证和加密传输。

五、TLS 1.3握手过程优化

使用 TLS 1.2 需要两次往返（ 2-RTT ）才能完成握手，然后才能发送请求。

TLS 1.3 协议只需要一次往返（ 1-RTT ）就可以完成握手，发送加密请求，更加安全。

发布于 2020-12-08 11:09

抓包 HTTPS Wireshark

文章被以下专栏收录

网络协议

推荐阅读

Wireshark抓包(网络分析)

前言贴一张wireshark抓包的总图，便于理解分析 网络分层 为了让大家更容易「看得见」 TCP，我搭建不少测试环境，并且数据包抓很多次，花费了不少时间，才抓到比较容易分析的数据包。 接下来...

你瞅啥

WireShark 抓包及常用协议分析

你瞅啥

WireShark 抓包及常用协议分析

like you

如何

叶焕