文章已收录我的仓库：Java学习笔记

HTTPS

什么是 HTTPS

HTTPS （全称：Hyper Text Transfer Protocol over SecureSocket Layer），是以安全为目标的 HTTP 通道，在HTTP的基础上通过传输加密和身份认证保证了传输过程的安全性。 HTTPS 中，使用 `传输层安全性(TLS)` 或 `安全套接字层(SSL)` 对通信协议进行加密，即 HTTP + SSL(TLS) = HTTPS。

阅读本文之前，希望你已经掌握了计算机网络方面的安全知识，可参考我的博客计算机网络中的安全。

> TLS 是 SSL3.0 的改进版本，它们之间的主要差别是加密的算法不同，从思想上看，它们的语义是相同的。由于 TLS 是基于 SSL 发展过来的，在下文中我们仅采用 SSL 以指代 SSL/TLS。

SSL 握手

SSL 握手阶段是整个 SSL 协议最复杂也是最核心的阶段，**SSL 在握手阶段中协商算法与密匙**，就像我们在计算机网络中的安全所说的，握手阶段采用的是非对称密码体制，而一旦算法与密匙协商完成，通信则使用对称密码体制。

在这一步中，SSL 利用非对称密码加密体制以协商**供通信使用的对称密码体制加密的算法和密匙 和 用于报文完整性加密(MAC)的算法和密匙**，此外为了防止**重放攻击**，SSL 采用生成随机数的方式去应对，具体的握手步骤如下：

1. 客户明文发送它支持的**密码算法的列表**，连同一个**客户的随机不重数**。
2. 从客户发来的列表中，服务器选择一种公匙算法(例如RSA)、一种对称算法(例如AES、DES) 和一种MAC算法(例如SHA-3)。服务器把它的**算法选择**和**证书**以及一个服务器的不重数发送(明文)给客户端，服务器还会**签名**还报文，**此时算法协商完毕**。

> 服务器可以要求客户端发送它的证书。

3. 服务器**生成一个前主密匙**（Pre-Master Secret，PMS，预主密匙），发送给客户端。

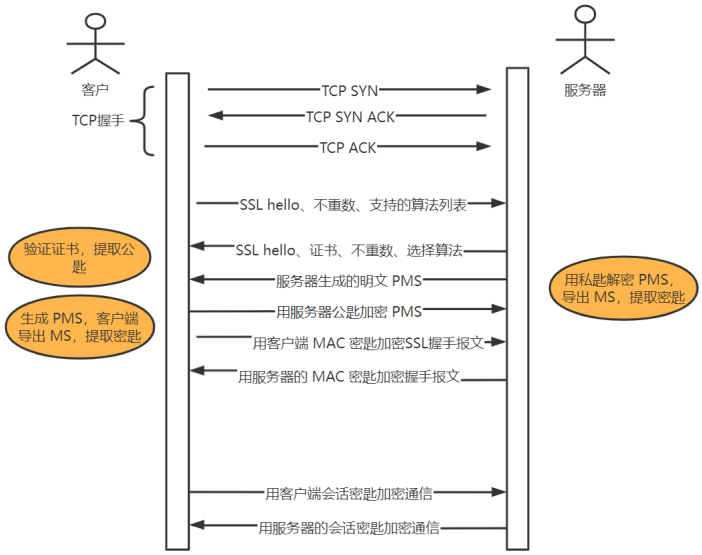
4. 客户端验证该证书，提取服务器公匙，**生成一个前主密匙**（Pre-Master Secret，PMS，预主密匙），用服务器的公匙加密 PMS，然后发送给服务器。

> 如果服务器要求客户端发送证书，这一步还必须发送证书。

4. 使用 SSL 标准定义密匙导出函数，客户端与服务器同时利用双方的不重数从各自收到的 PMS 中导出相同的**主密匙**（Master Secret，MS）。**从 MS 中导出双方需要的对称密匙以及MAC密匙，到此密匙协商完毕**。

5. 客户端利用 MAC 密匙发送所有握手报文的一个 **MAC（报文摘要）**。

6. 服务器利用 MAC 密匙发送所有握手报文的一个 **MAC（报文摘要）**。



> 注意，**根据上述步骤，这里有 4 个密匙**，E_A、M_A、E_B 和 M_B，分别是发送方和接收方的会话密匙和 MAC 密匙，双方共享四个密匙，例如 A 利用 E_A、M_A 发送加密消息给 B，B 利用 E_A、M_A 解密；而 B 利用 E_B 和 M_B 发送加密消息给 A，A 利用 E_B 和 M_B 解密。
>
> 双方发送利用各自的密匙发送消息，使得整个会话安全更为健壮，即使有一方的密文被破解，另一方的密文仍然无法破解。
>
> 在下文中，我们可能会抽象的将这 4 个密匙当作两个密匙，E 和 M。

分析：

- 不重数是有必要的，利用不重数可以防止重放攻击，如果黑客原封不动的重放报文，那么服务器将会拒绝此次连接，因为不重数重复。你可能会有疑问：黑客不可以更改不重数吗？正是考虑到了这一点，所以**主密匙的生成需要利用到不重数**，如果不重数不同，生成的密匙肯定是不同的，密匙不同，黑客自然无法之前破解双方通信的内容了。
- **数字签名**也是有必要的，这是为了防止黑客劫持该证书报文，而伪装成服务器与客户端通信(黑客会更改不重数)。
- 最后的报文摘要也是必要的，这是为了防止黑客篡改报文，例如黑客劫持步骤 1 中的报文，删掉其中比较强的算法，迫使服务器与客户端选择较弱的算法通信。

密匙交换 ECDH算法

> 我们假定双方只需要两个密匙，一个密匙用于对称密码加密，定义为 E，一个密匙用于 MAC 加密，定义为 M，而不是前面说的 4 个。

前主密匙怎么生成的？主密匙又是怎么生成的？如何重主密匙中导出双方需要的密匙？

我们先回答第三个问题：切分主密匙即可，换句话说，主密匙就是就是 E 和 M 的合并字符串，E 与 M 的长度在协商算法时就已经确认了，例如 E = 101，M = 011，则主密匙可能是 101011，切分时按照长度切割即可。

现在我们来讨论如何生成 PMS 以及如何从 PMS 中解析出 MS，ECDH 算法已经成为了当下主流的实现，因此我们主要介绍一下 ECDH 算法。

我们定义：

S_A 是客户端生成的一个私匙 S_B 是服务器生成的一个私匙 P 是一个大质数，G 是一个整数，P、G是公开的

假设 PMS_A是客户端生成发送给服务器的前主密匙，PMS_B是服务器生成发送给客户端的前主密匙，那么其计算公式为：

$$PMS_A = G^{S_A} mod\ P \quad PMS_B = G^{S_B} mod\ P$$

那么经过交换后，客户端握有 PMS_B，而服务器拥有 PMS_A，现在它们分别对收到的 PSM 进行如下运算：

客户端：$(PMS_B)^{S_A}\ mod\ P = (G^{S_B} mod\ P)^{S_A}\ mod\ P = K_1$ 服务器：$(PMS_A)^{S_B}\ mod\ P = (G^{S_A} mod\ P)^{S_B}\ mod\ P$

在计算机网络中的安全安全中我们证明过一个结论：

$$(m^d\ mod\ n)^e\ mod\ n = (m^e\ mod\ n)^d\ mod\ n$$

因此可以得：

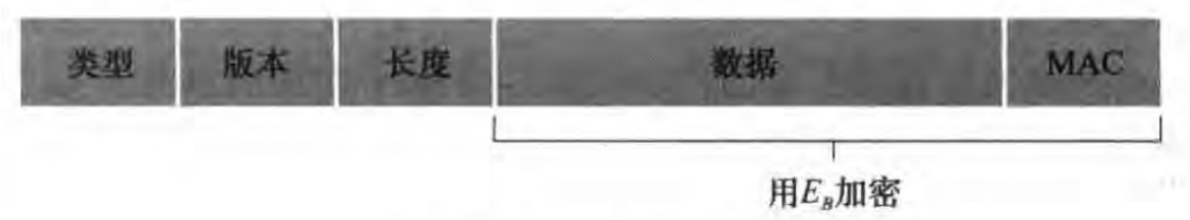
$$K_1 = K_2 = K$$

即，**双方根据收到的 PSM 经过运算后会得到一致的数据**，这个 K 就是我们要的主密匙 MS，从主密匙中就可以切分出我们具体需要的密匙了！而**密匙 S_A、S_B的生成依赖于各自的随机数(不等于随机数)**，P、G 是 SSL 标准规定的，这样我们就彻底揭开了密匙交换过程的面纱。

在 SSL 握手的第 4 步中，注意到服务器发送给客户端 PSM_B 是没有加密的，这意味着服务器的 S_B 是暴露在外的，然而即使黑客得知了 S_B 也无能为例，因为在生成 MS 的计算中，是同时需要 S_A、S_B 的，而 S_A 是加密的，因为客户发送的 PSM_A 会用服务器的公匙加密，只有服务器自己能够解密，然后得出 S_A，黑客无法获取 S_A ，故我们可以保证**整个密匙交换过程是安全的**！

数据传输

运输层仍然是使用 TCP 传输，这就可能会遭受 **重排序攻击**，因此 SSL 设计了 SSL 记录以防止该攻击（同时也用于关闭连接），SSL 记录位于应用层与传输层之间，应用层的数据需要被处理成 SSL 记录才会交给 TCP 层。



- 类型。该字段指示这是 SSL 握手报文，还是通信报文，或是 SSL 连接关闭报文。
- 版本。指示 SSL 报文协议，例如 TLS。
- 长度。指示 SSL 记录的长度，以便 SSL 处理程序从 TCP 字节流中提取出 SSL 记录（解决粘包问题）。
- 数据。要发送的应用程序的数据。
- MAC。对整个 SSL 记录的摘要，注意，这个摘要还包含了序号。

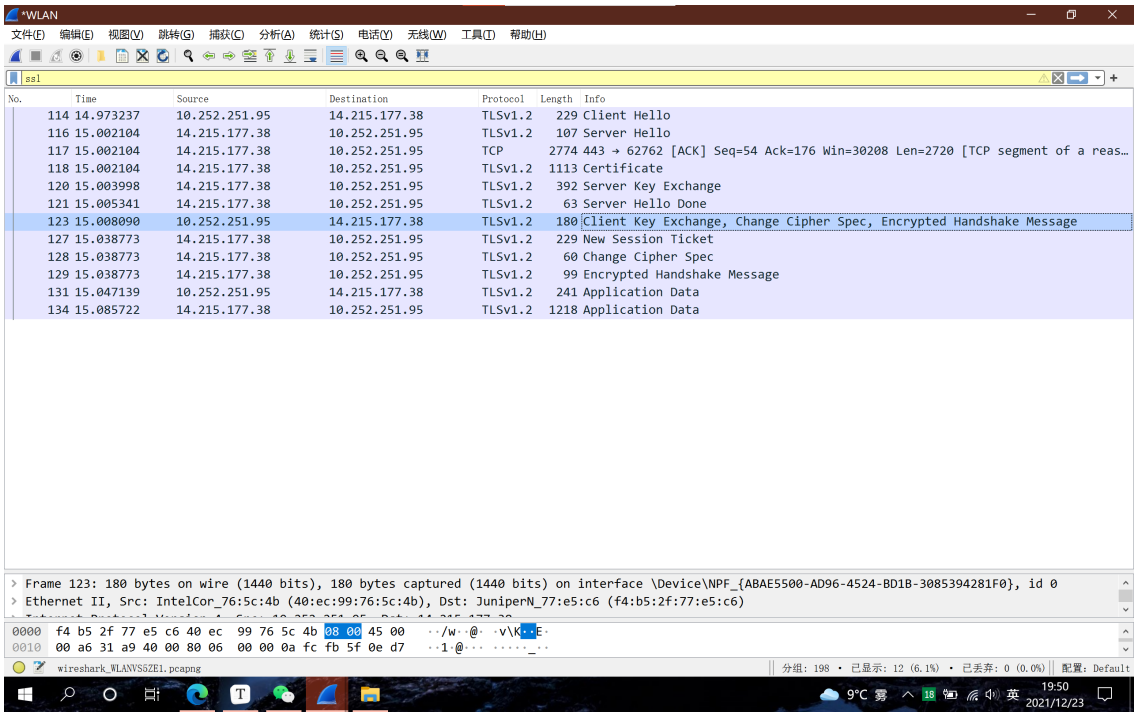
SSL 采用序号解决重排序攻击，SSL 记录除 数据 和 MAC 外都是明文发送的（但它们都可以通过 MAC 验证，所以无须担心这些明文被修改），序号没有一个显示的字段指示，而是被保存在 MAC 中，这样更安全。双方遵守 SSL 标准以获取/添加序号，例如可以选择 MAC 中的前 4 字节作为序号。发送方每次发送数据都会递增 SSL 序号，初始时它为 0，序号被加密在 MAC 中，通过这种方式以防止重排序攻击。

中止连接

通过 SSL 记录中的类型以发出中止连接，无须担心黑客更改类型或仿造发出中止报文，因为加密的 MAC 对它进行了鉴别。服务器应答后客户端即可发出 TCP FIN。

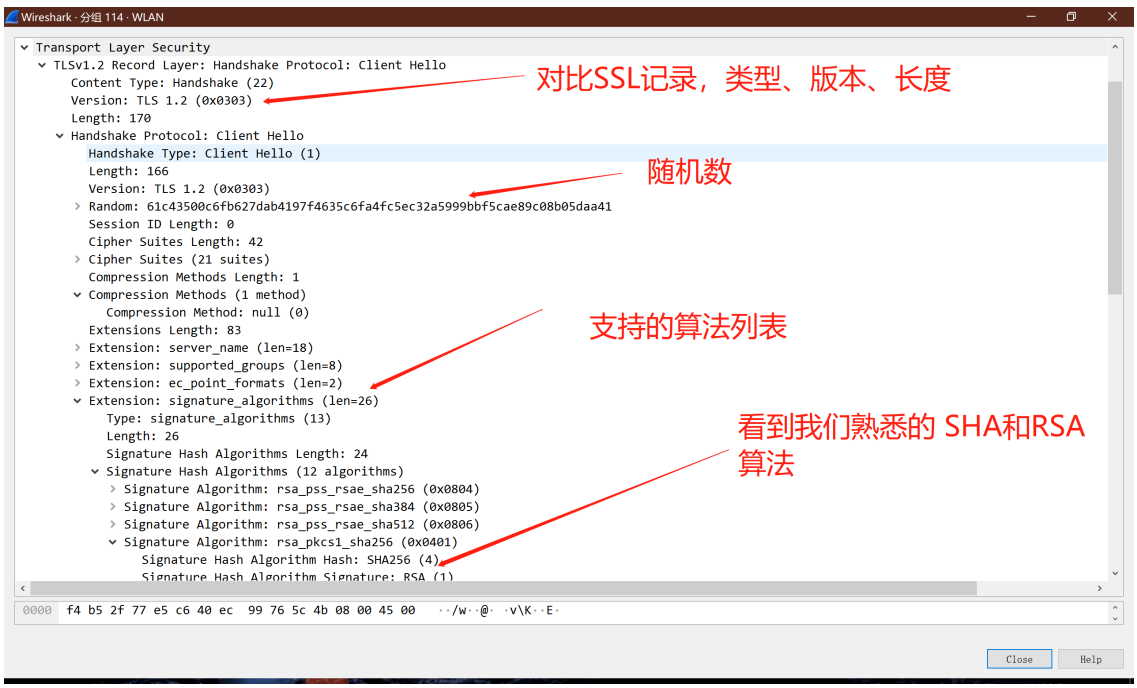
SSL 握手抓包实践

随便搜一个 HTTPS 的网站，例如：`curl https://www.baidu.com`，得到数据包如下：

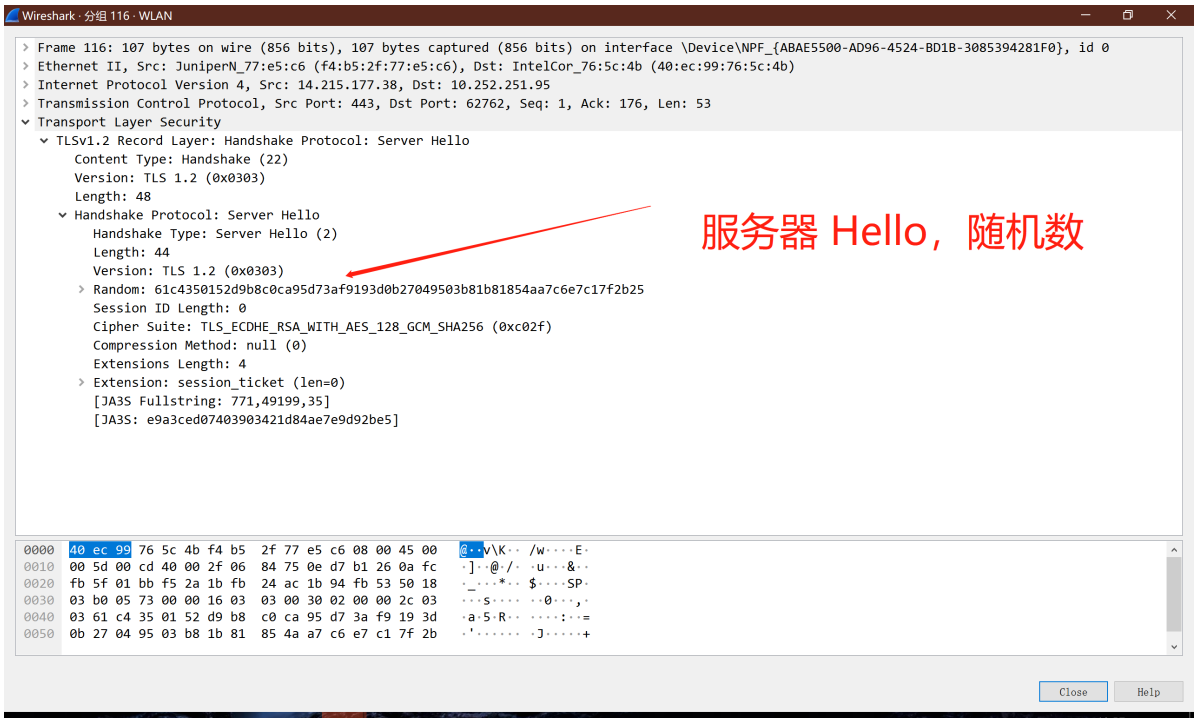


> 10.252.251.95 是我自己的 IP，14.215.177.38 是服务器 IP

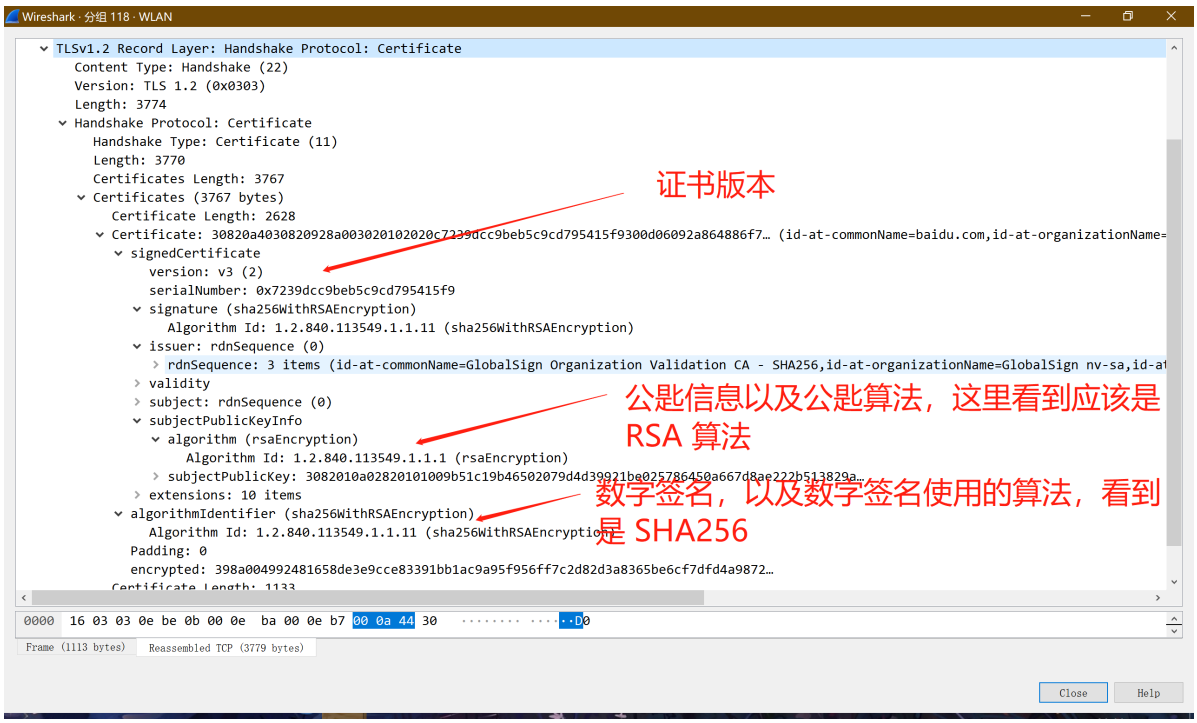
Cilent Hello 包



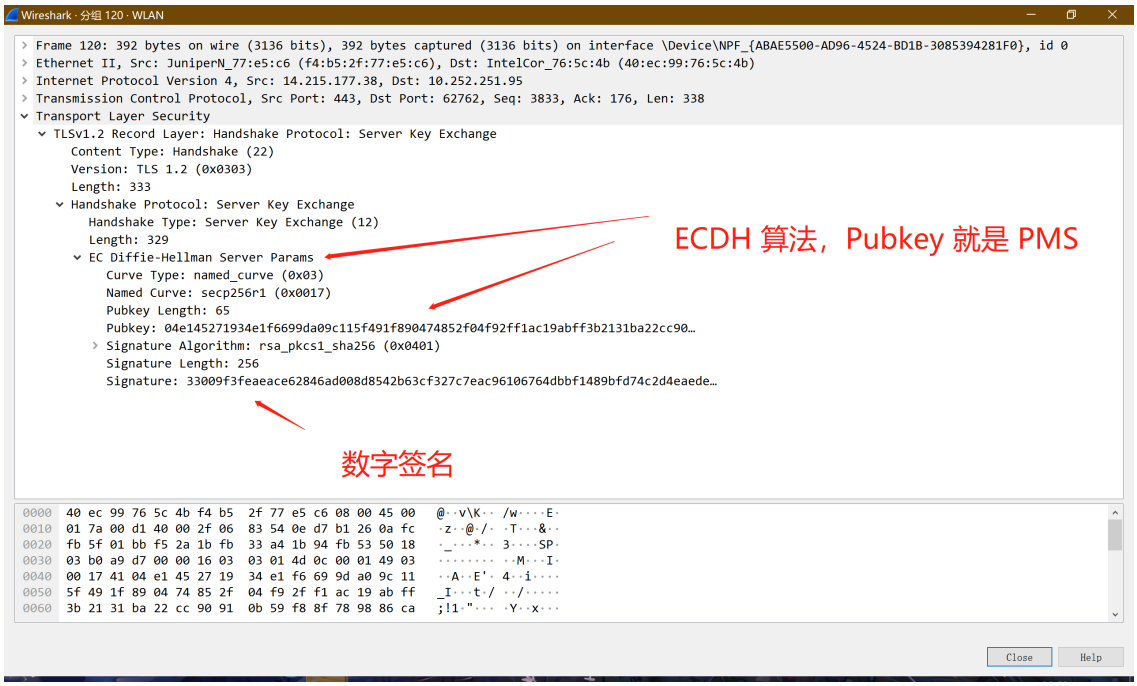
Server Hello 包



Certificate 包



Server Key Exchange

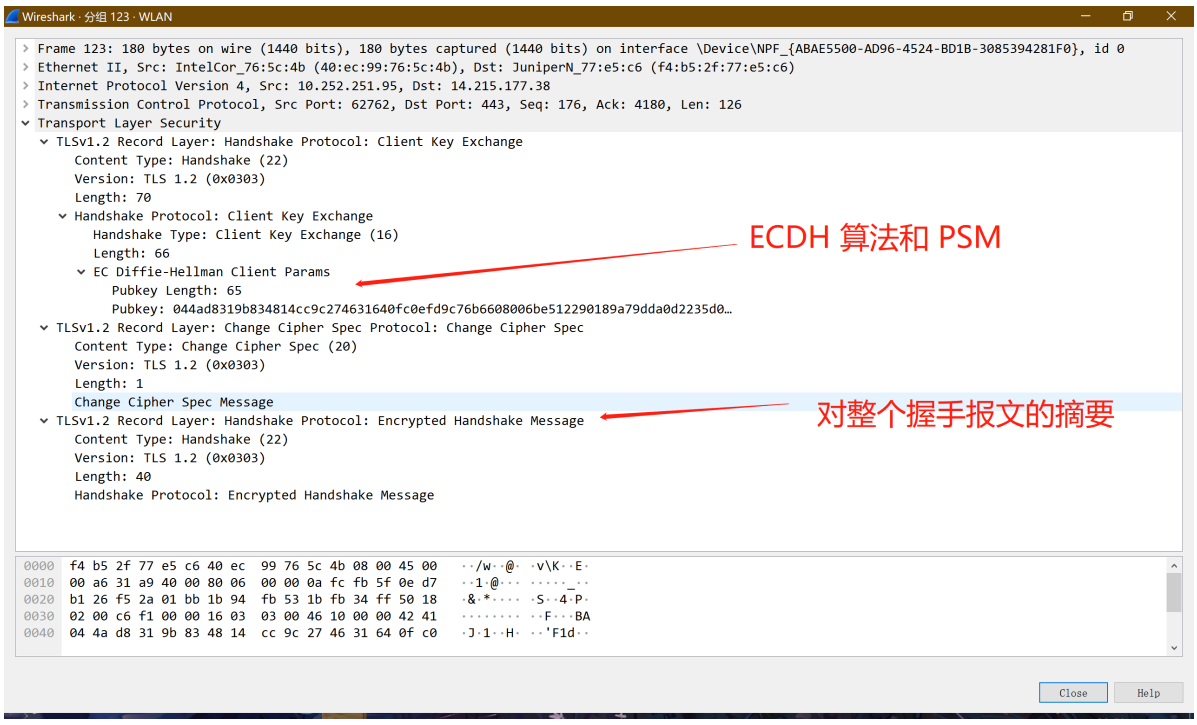


Server Hello Done 包

这个没啥说的，就是结束 Server Hello。

Cilent Key Exchange, Change Cipher Spec, Encrypted HandShack Message 包

看名字也能猜出个大概，**客户密匙交换、更改密码规范以及对整个客户端握手的MAC报文摘要**，更改密码规范我们没说，这是 TLS 引入的新规范，**即要求通信双方每隔一段时间后就更换新的密匙**，以加强安全，本文会略过对该规范的讲解。



Server Encrypted HandShack Message

剩下的就是服务器生成的握手报文摘要了，整个握手环节到此结束。

请比对一下我们之前说的步骤，看看是否吻合，结果应该是一致的，只是某些报文合并成一个报文了或者一些报文拆分成多个报文。

总结

通过理论分析和实践抓包，相信理解 HTTPS 应该没有任何困难了，我们已经彻底揭开 HTTPS 神秘的面纱。