

计算机网络 课程实验报告

学号：202000130143	姓名：郑凯饶	班级：2020 级 1 班
实验题目：SSL		
实验学时：2	实验日期：2022-6-7	
实验目的： 研究安全套接字协议，关注通过 TCP 传送的 SSL 记录。		
硬件环境： Dell Latitude 5411 Intel (R) Core (TM) i5-10400H CPU @ 2.60GHz (8GPUs), ~2.6GHz		
软件环境： Windows 10 家庭中文版 64 位 (10.0, 版本 18363) Wireshark-win64-3.6.2		
实验步骤与内容： <ol style="list-style-type: none"> 问题： <ol style="list-style-type: none"> 对于前八个以太网帧，指出每一帧的来源，确定每帧包含的 SSL 记录数量，并且列出记录类型，绘制时序图。 列出 SSL 记录所有字段及长度。 展开 ClientHello 记录，回答内容类型的值。 ClientHello 是否包含不重数，十六进制值是多少？ ClientHello 通知了它所支持的加密套件 (cyber suites) 对第一个密码套件，指出非对称、对称、哈希算法分别是什么？ 找到 ServerHello SSL 记录。此记录是否指定了之前的密码套件，选择的密码套件中有哪些算法？ 是否包含不重数，它的用途？ 会话 ID。 是否包含证书，证书是否适合用单一的以太网帧传输？ 找到客户端密钥交换记录，是否包含前主密钥，它的用途，是否加密，加密后的长度？ Chipher Spec record 的作用？字节数？ Handshake record 中什么被加密了，为什么？ 服务器是否向客户端发送 Chipher Spec record 和 Handshake record，与客户端发送的有什么不同？ 应用程序数据如何加密？记录是否包含消息认证码 MAC？Wireshark 是否区分数据和 MAC？ 指出并解释你的其他发现？ 阐述基本方法 TSL 的四次握手过程： 		

TLS第一次握手

clientHello阶段，客户端向服务器发送请求，发送SSL/TLS版本，发送客户端支持的密码套件列表，然后会产生一个随机数（Client Random），发送给服务器，这是生成对称密钥的材料。

第一次握手客户端做的事：

客户端clientHello，客户端发送请求

- 确定SSL/TLS版本
- 客户端生成的随机数，用于生成会话密钥（Client Random）
- 客户端支持的密码套件列表，如RSA算法

TLS第二次握手

服务器会确定自己所支持的SSL/TLS版本，如果不支持，则直接关闭此次连接。在客户端发送来的密码套件中选择一套。生成一个随机数（Server Random）

serverHello阶段，服务器响应客户端的请求，发送SSL/TLS版本，发送服务器选择的密码套件，发送随机数

```
Length: 96
Version: TLS 1.2 (0x0303)
> Random: b02cb6ee7dc846d167cc5975ea87ac45a79a3dea5c76d02141baef772c556363
Session ID Length: 32
Session ID: 1c49fc3db5466549109d7b068859195b931509620e91eadf3b5e0c07680bcb10
Cipher Suite: TLS_RSA_WITH_AES_128_GCM_SHA256 (0x009c)
Compression Method: null (0)
```

确认支持 TLS 的版本

随机数

选择的密码套件

密码套件：密钥交换算法 + 签名算法 + 对称加密Q 算法 + 摘要算法

WITH单词前面只有一个单词RSA

- 密钥交换算法和签名算法都是RSA
- 加密对称算法采用的是AES算法，密钥长度为128位，分组模式为GCM
- 摘要算法为SHA256，用于消息认证和产生随机数

发送数字证书

最后会发一个Server Hello Done的信息

第二次握手服务器做的事：

服务器接收到客户端请求，进行响应

- 确定SSL/TLS版本，如果不支持，直接关闭
- 服务器生成的随机数，后面用于生成会话密钥（**Server Random**）
- 确定服务器支持的密码套件列表，如RSA加密算法
- 发送数字证书

客户端拿到服务器的响应后，会去验证证书的证实有效性

客户端拿到服务器的响应后，会去验证证书的证实有效性

数字证书包含

- 公钥
- 持有者信息
- 有效时间
- 证书认证机构（CA）的信息
- 其他信息

数字证书的签发流程

- CA会把一些基本信息例如持有者、用途、有效信息等打包，然后运用Hash算法得到一个Hash值
- CA会使用自己的私钥将这个Hash值加密，生成一个Certificate Signature（证书签名），CA对证书进行了签名
- 最后将签名加到证书文件，形成数字证书

客户端验证证书的流程

- 客户端使用同样的Hash算法，得到该证书的Hash值H1
- 浏览器使用CA的公钥，解密Certificate Signature中的内容，得到Hash值H2
- 如果H1等于H2，则证明可信

TLS的第三次握手

证书有效，客户端生成一个随机数（pre-master），用服务器的RSA公钥加密，然后通过**Change Cipher Key Exchange**消息发给服务器

服务器收到后，用私钥解密收到pre-master

到此阶段，客户端和服务端共享了三个数，Client Random、Server Random、pre-master，双方根据这三个数生成密钥

生成密钥后客户端再发一个**Change Cipher Spec**告诉服务器开始使用加密通话

最后在发送一个**Encrypted HandShare Message**消息，向服务器发送握手摘要，再加密一下，让服务器验证

第三次握手客户端做的事：

向服务器进行响应

- 发送一个随机数（密钥），该随机数会被公钥加密（pre-master）
- 加密算法改变的通知，表示之后的会话都将用会话密钥进行加密（Change Cipher Key Exchange）
- 客户端响应结束通知，握手数据的摘要（Encrypted HandShare Message）

TLS的第四次握手

服务器也是相同操作，发送**Change Cipher Spec**和**Encrypted HandShare Message**

服务器进行响应

- 加密算法改变通知，表示之后的信息会用加密的密钥进行加密（Change Cipher Spec）
- 服务器响应结束通知，握手数据的摘要（Encrypted HandShare Message）

3. 实验结果展示与分析

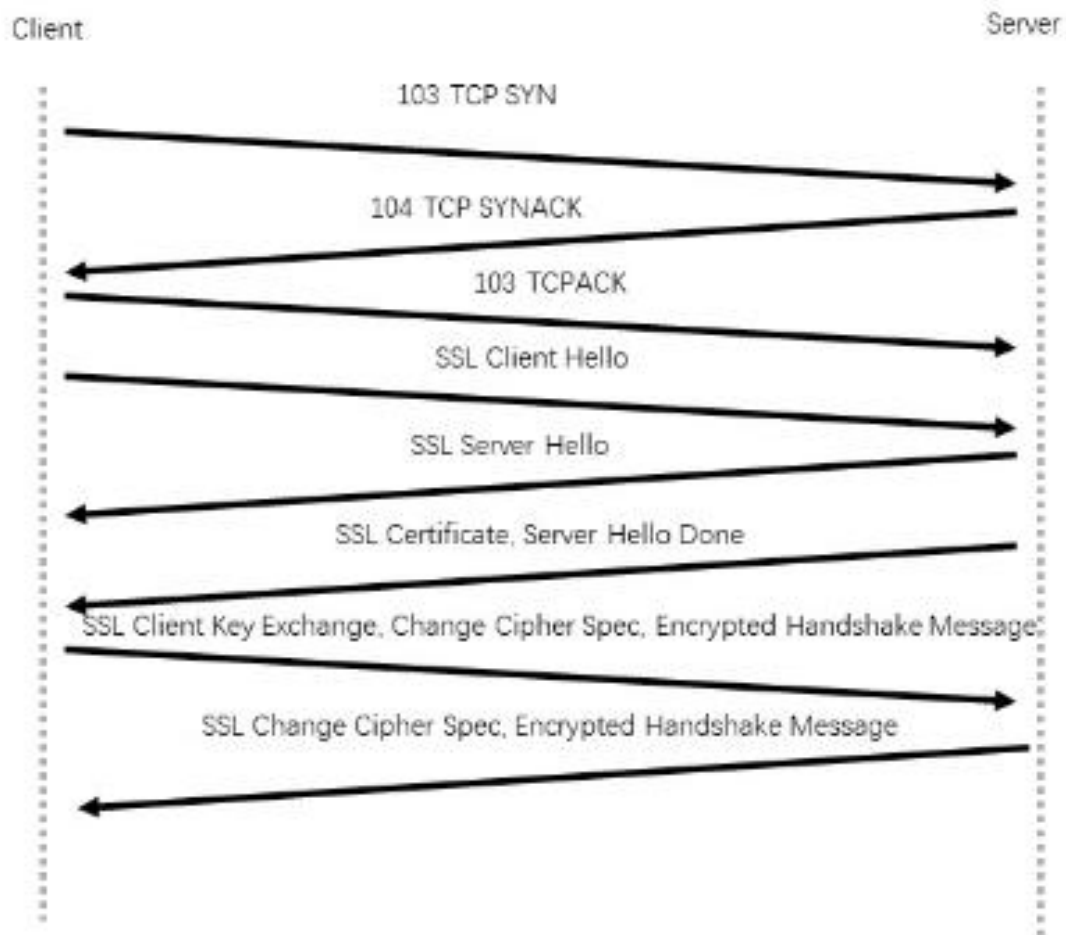
（1）筛选得

	Time	Source	Destination	Protocol	Length	Info
106	21.805705	128.238.38.162	216.75.194.220	SSLv2	132	Client Hello
108	21.830201	216.75.194.220	128.238.38.162	SSLv3	1434	Server Hello
111	21.853520	216.75.194.220	128.238.38.162	SSLv3	790	Certificate, Server Hello Done
112	21.876168	128.238.38.162	216.75.194.220	SSLv3	258	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
113	21.945667	216.75.194.220	128.238.38.162	SSLv3	121	Change Cipher Spec, Encrypted Handshake Message
114	21.954189	128.238.38.162	216.75.194.220	SSLv3	806	Application Data
122	23.480352	216.75.194.220	128.238.38.162	SSLv3	272	Application Data
149	23.559497	216.75.194.220	128.238.38.162	SSLv3	1367	Application Data
158	23.560866	216.75.194.220	128.238.38.162	SSLv3	1367	Application Data
163	23.566451	128.238.38.162	216.75.194.220	SSLv3	156	Client Hello
165	23.586650	216.75.194.220	128.238.38.162	SSLv3	1329	Application Data
169	23.591590	216.75.194.220	128.238.38.162	SSLv3	200	Server Hello, Change Cipher Spec, Encrypted Handshake Message
171	23.599417	128.238.38.162	216.75.194.220	SSLv3	121	Change Cipher Spec, Encrypted Handshake Message
172	23.602696	128.238.38.162	216.75.194.220	SSLv3	470	Application Data
103	21.774217	128.238.38.162	216.75.194.220	TCP	62	2271 → 443 [SYN] Seq=0 Win=65535 Len=0
104	21.796243	216.75.194.220	128.238.38.162	TCP	62	443 → 2271 [SYN, ACK] Seq=0 Ack=65535 Len=0
105	21.796299	128.238.38.162	216.75.194.220	TCP	54	2271 → 443 [ACK] Seq=1 Ack=1 Len=0
106	21.805705	128.238.38.162	216.75.194.220	SSLv2	132	Client Hello
107	21.828148	216.75.194.220	128.238.38.162	TCP	60	443 → 2271 [ACK] Seq=1 Ack=79 Len=0
108	21.830201	216.75.194.220	128.238.38.162	SSLv3	1434	Server Hello
109	21.830228	216.75.194.220	128.238.38.162	TCP	722	443 → 2271 [PSH, ACK] Seq=1381 Ack=79 Len=0
110	21.830263	128.238.38.162	216.75.194.220	TCP	54	2271 → 443 [ACK] Seq=79 Ack=2060 Len=0
111	21.853520	216.75.194.220	128.238.38.162	SSLv3	790	Certificate, Server Hello Done
112	21.876168	128.238.38.162	216.75.194.220	SSLv3	258	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
113	21.945667	216.75.194.220	128.238.38.162	SSLv3	121	Change Cipher Spec, Encrypted Handshake Message
114	21.954189	128.238.38.162	216.75.194.220	SSLv3	806	Application Data

103–113 帧为前八帧。SSL 记录来源及类型如下：

106	客户到服务器	1	Client Hello
108	服务器到客户	1	Server Hello
111	服务器到客户	2	Certificate, Server Hello Done
112	客户到服务器	3	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
113	服务器到客户	3	Change Cipher Spec, Encrypted Handshake Message

时序图如下：



(2) 依次占 2Byte, 1B, 2B

Length: 76

Handshake Message Type: Client Hello (1)

Version: SSL 3.0 (0x0300)

(3) 0x01

(4)

Challenge

03 00 80 00 00 09 06 00 40 00 00 64 00 00 62 00 @..d..b.
00 03 00 00 06 02 00 80 04 00 80 00 00 13 00 00
12 00 00 63 66 df 78 4c 04 8c d6 04 35 dc 44 89	...cf..xL5.D.
89 46 99 09	.F..

(5) 非对称: RSA 对称: RC4 哈希: MD5

▼ Cipher Specs (17 specs)

Cipher Spec: TLS_RSA_WITH_RC4_128_MD5 (0x000004)

Cipher Spec: TLS_RSA_WITH_RC4_128_SHA (0x000005)

Cipher Spec: TLS_RSA_WITH_3DES_EDE_CBC_SHA (0x00000a)

Cipher Spec: SSL2_RC4_128_WITH_MD5 (0x010080)

Cipher Spec: SSL2_DES_192_EDE3_CBC_WITH_MD5 (0x0700c0)

Cipher Spec: SSL2_RC2_128_CBC_WITH_MD5 (0x030080)

Cipher Spec: TLS_RSA_WITH_DES_CBC_SHA (0x000009)

(6)

▼ Handshake Protocol: Server Hello

Handshake Type: Server Hello (2)

Length: 70

Version: SSL 3.0 (0x0300)

➤ Random: 0000000042dbed248b8831d04cc98c26e5badc4e267c391944f0f070ece57745

Session ID Length: 32

Session ID: 1bad05faba02ea92c64c54be4547c32f3e3ca63d3a0c86ddad694b45682da22f

Cipher Suite: TLS_RSA_WITH_RC4_128_MD5 (0x0004)

Compression Method: null (0)

[JA3S Fullstring: 768,4,]

[JA3S: 1f8f5a3d2fd435e36084db890693eafd]

(7) 有如上图，防止“重放攻击”

(8) 如图

(9) 是，在单独的记录中。证书较长需要分片传输。

➤ Transmission Control Protocol, Src Port: 443, Dst Port: 2271, Seq: 2049, Ack: 79, Len: 736

➤ [3 Reassembled TCP Segments (2696 bytes): #108(1301), #109(668), #111(727)]

➤ Transport Layer Security

➤ Transport Layer Security

(10) 如图，通过 RSA 加密（使用服务器公钥），长度为 128B，和不重数一起用于生成主密钥。

▼ Transport Layer Security

▼ SSLv3 Record Layer: Handshake Protocol: Client Key Exchange

Content Type: Handshake (22)

Version: SSL 3.0 (0x0300)

Length: 132

▼ Handshake Protocol: Client Key Exchange

Handshake Type: Client Key Exchange (16)

Length: 128

▼ RSA Encrypted PreMaster Secret

Encrypted PreMaster: bc49494729aa2590477fd059056ae78956c77b12af08b47c609e61f104b0fbf83e41c08d...

(11) 指示之后发送信息是经过加密的。长度为 1B

▼ SSLv3 Record Layer: Change Cipher Spec Protocol: Change Cipher Spec

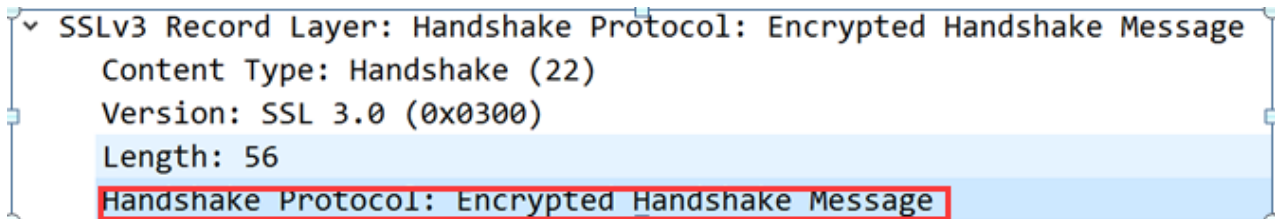
Content Type: Change Cipher Spec (20)

Version: SSL 3.0 (0x0300)

Length: 1

Change Cipher Spec Message

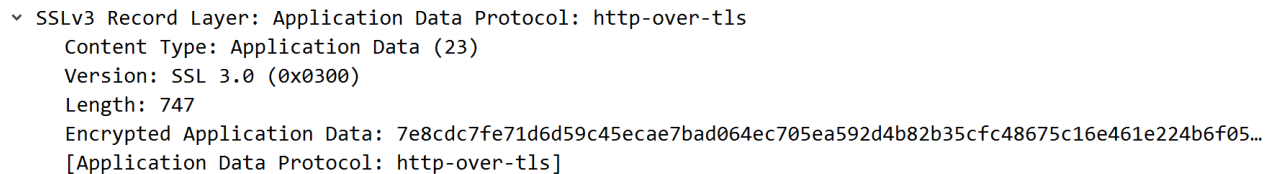
(12) Encrypted Handshake Message 被加密



SSLv3 Record Layer: Handshake Protocol: Encrypted Handshake Message
Content Type: Handshake (22)
Version: SSL 3.0 (0x0300)
Length: 56
Handshake Protocol: Encrypted Handshake Message

(13) 是，部分不同，Handshake Message 的明文是所有握手机报文的 MAC，但由于双方 MAC 密钥不同，摘要后的密文不同。

(14) 使用之前协商的方法进行加密，而相应的密钥通过主密钥切片而得。包含用于报文完整性鉴别，Wireshark 不区分。



SSLv3 Record Layer: Application Data Protocol: http-over-tls
Content Type: Application Data (23)
Version: SSL 3.0 (0x0300)
Length: 747
Encrypted Application Data: 7e8cdc7fe71d6d59c45ecae7bad064ec705ea592d4b82b35cfc48675c16e461e224b6f05...
[Application Data Protocol: http-over-tls]

结论分析与体会：

理论上如果我们是信息交流的一方，我们可以解密所有会话（好像这是显然的），只不过对于普通用户来说加解密是透明的。假期我希望尝试一下通过 Wireshark 进行 SSL 报文的解密，或者自己尝试编程解密算法。TLS 四次握手进行了身份认证、会话密钥交换，保证之后会话的完整性、保密性、权威性，是网络通信的重要组分。希望以后还有机会学习网络安全、密码学相关的知识。