



西安交通大学

计算机网络原理与应用课程作业

姓名： 屈彬

学号： 2140505062

次数： 第 1 次

日期： 2018 年 3 月 15 日

## 目录

第 1 章 实验背景	1
1.1 TLS 简介 . . . . .	1
1.2 实验目标 . . . . .	3
第 2 章 实验工具	3
第 3 章 实验过程与结果	3
3.1 抓包过程 . . . . .	3
3.2 不同层重要字段与 TLS 握手过程分析 . . . . .	5
3.3 表单解密 . . . . .	8

## 第 1 章 实验背景

### 1.1 TLS 简介

TLS 全称“Transport Layer Security（安全传输层）”，用于在两个通信应用程序之间提供保密性和数据完整性，常用于 Web 应用中客户端与服务器的加密通信。虽然 TLS 协议被定义为传输层的安全协议，但其实 TLS 协议部分作为 SSL 层的协议是被封装在 TCP 等传输层协议内的，而应用层的内容则封装在 TLS 协议的应用数据层内。在登录西安交通大学学生版统一认证网关的过程中，客户端页面也采用 TLS 协议向服务器提交表单，因此通过本实验也能对西安交通大学统一认证网关的安全机制有更好的了解。

通过查阅国内相关文献<sup>1</sup>以及国外的一些文献<sup>3</sup>，可以了解到 TLS 协议最底层协议为记录协议，上层协议包括握手协议、应用数据协议、更改密码协议以及警告协议。记录协议从子协议收到数据后，对它们进行数据分段、压缩、认证和加密形成记录；更改密码规格协议将密文状态由挂起状态复制到当前状态；警告协议用来传递相关警告。由于握手过程是可以被 Wireshark 软件轻易捕获的，在抓包时，我们主要关心的是 TLS 协议中的握手过程。TLS 的握手过程如图 1.1 所示。

在图 1.1 中，左侧代表客户端，右侧代表服务端。当客户端向服务端发起握手时，客户端首先向服务端发送“ClientHello”字段，包含会话 ID（Session ID）等信息；服务端收到“ClientHello”字段后，返回一个“ServerHello”字段，同样包含会话 ID 等信息；之后服务端和客户端之间会进行密钥交换，在结束握手前，客户端和服务端还向对方发送更改密钥的信息。

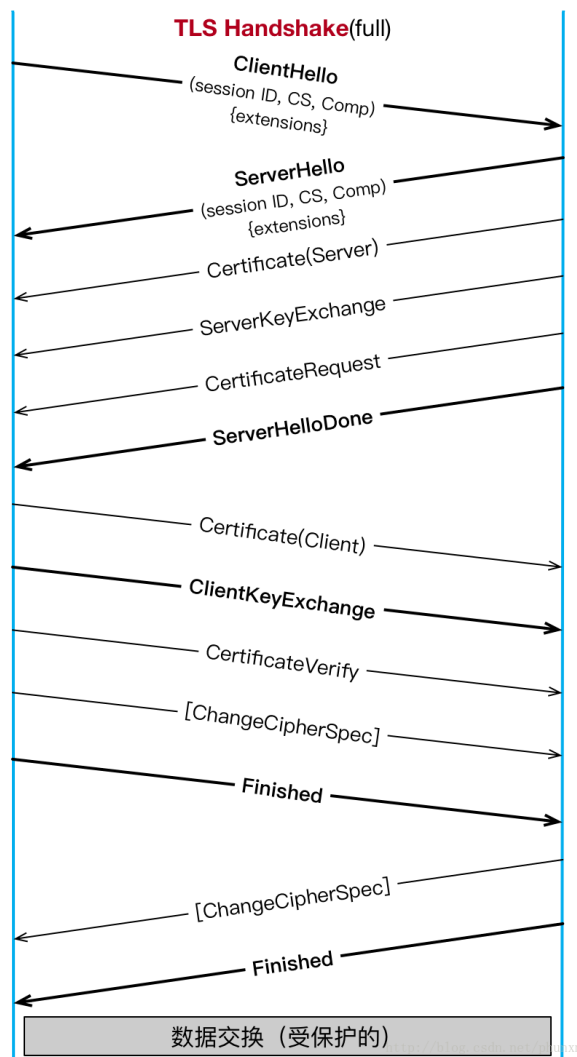


图 1.1: TLS 协议握手过程<sup>2</sup>

---

但在实际的网络应用中，TLS 协议的协议组成和握手过程是否真的如上所述。本实验将通过 Wireshark 软件对西安交通大学学生版统一身份认证网关的登录过程进行抓包分析，以探究竟。

## 1.2 实验目标

1. 利用 Wireshark 软件对西安交通大学学生版统一认证网关的登录过程进行抓包。
2. 分析不同层的重要协议头字段。
3. 分析 TLS 协议中记录协议与其子协议的关系。
4. 理解和分析 TLS 协议的握手过程。
5. 说明 TLS 协议握手过程体现了计算机网络的何种概念和原理。
6. 解密客户端向服务端提交的表单。

## 第 2 章 实验工具

1. Wireshark 软件：用于获取网络通信过程经过本地的数据包。本实验所使用的版本为 2.4.5。
2. 支持 IE6 以上内核的浏览器：用于访问西安交通大学学生版统一身份认证网关。
3. CTeX + TeXstudio：用于报告写作（首次使用）。

## 第 3 章 实验过程与结果

### 3.1 抓包过程

首先确保网络通畅，查看本机 IPv4 地址（调用 Windows 控制台 ipconfig 命令查看）和服务端 IPv4 地址（打开相应的网页，按 F12 查看），本实验中客户端地址为 192.168.123.64，客户端处于一个子网中，服务端地址为 202.117.1.185。

在浏览器 URL 栏输入“<https://cas.xjtu.edu.cn/login>”，回车后打开西安交通大学学生版统一身份认证网关，如图 3.1 所示。

打开 Wireshark 软件，在初始界面选择活跃的网络连接，如图 3.2 所示。本实验中客户端计算机通过无线局域网上网，所以选择 WLAN 连接。

在打开验证页面上输入用户名和密码，在点击登录之前，先确定 Wireshark 开始捕获分组，然后再进行登录。当页面上提示登陆成功后，停止捕获分组。将捕获到的分组保存为本地文件，以方便日后查看。本次实验中捕获到的分组如图 3.3 所示。



图 3.1: 西安交通大学学生版统一身份认证网关

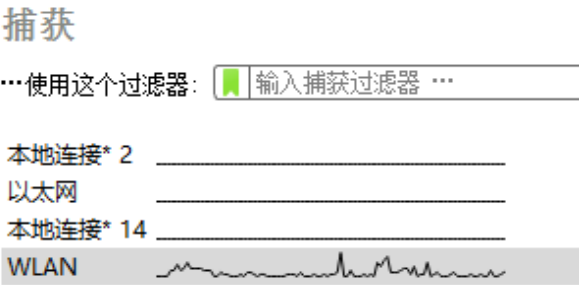


图 3.2: 在 Wireshark 初始界面选择活跃的网络连接

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	221.181.72.211	192.168.123.24	TLSv1.2	85	Encrypted Alert
2	0.000221	221.181.72.211	192.168.123.24	TCP	54	443 → 53379 [FIN, ACK] Seq=32 Ack=1 Win=60 Len=0
3	0.000336	192.168.123.24	221.181.72.211	TCP	54	53379 → 443 [ACK] Seq=1 Ack=33 Win=257 Len=0
4	0.495925	192.168.123.24	221.181.72.244	SSL	55	Continuation Data
5	2.281661	192.168.123.24	202.117.1.185	TCP	54	53446 → 443 [FIN, ACK] Seq=1 Ack=1 Win=256 Len=0
6	2.281791	192.168.123.24	202.117.1.185	TCP	54	53446 → 443 [RST, ACK] Seq=2 Ack=1 Win=0 Len=0
7	2.281967	192.168.123.24	202.117.1.185	TCP	54	53447 → 443 [FIN, ACK] Seq=1 Ack=1 Win=256 Len=0
8	2.282043	192.168.123.24	202.117.1.185	TCP	54	53447 → 443 [RST, ACK] Seq=2 Ack=1 Win=0 Len=0
9	2.282194	192.168.123.24	202.117.1.185	TCP	54	53449 → 443 [FIN, ACK] Seq=1 Ack=1 Win=64 Len=0
10	2.282266	192.168.123.24	202.117.1.185	TCP	54	53449 → 443 [RST, ACK] Seq=2 Ack=1 Win=0 Len=0
11	2.282453	192.168.123.24	202.117.1.185	TCP	54	53448 → 443 [FIN, ACK] Seq=1 Ack=1 Win=64 Len=0
12	2.282527	192.168.123.24	202.117.1.185	TCP	54	53448 → 443 [RST, ACK] Seq=2 Ack=1 Win=0 Len=0
13	2.282683	192.168.123.24	202.117.1.185	TCP	54	53450 → 443 [FIN, ACK] Seq=1 Ack=1 Win=64 Len=0

图 3.3: Wireshark 捕获的分组（局部）

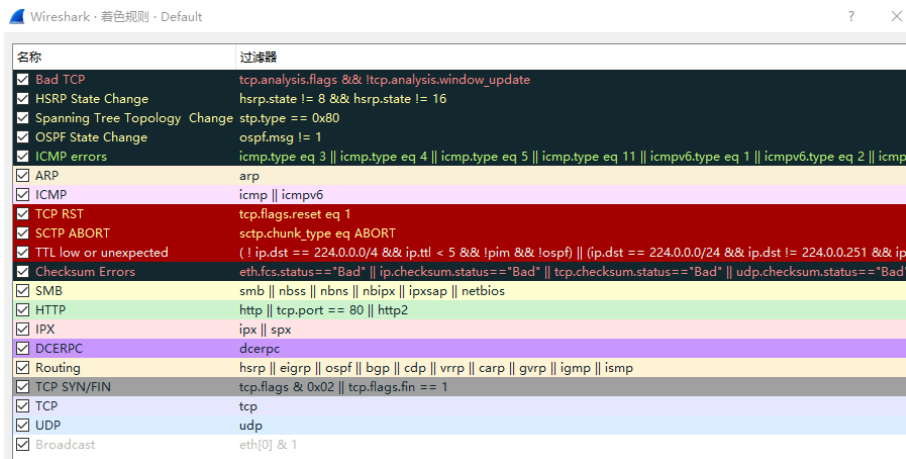


图 3.4: 着色规则

141	2.297846	192.168.123.24	202.117.1.185	TLSv1	295 Client Hello
142	2.300194	202.117.1.185	192.168.123.24	TCP	54 443 → 53459 [ACK] Seq=1 Ack=242 Win=7168 Len=0
143	2.300479	202.117.1.185	192.168.123.24	TLSv1	140 Server Hello
144	2.300480	202.117.1.185	192.168.123.24	TLSv1	60 Change Cipher Spec
145	2.300604	192.168.123.24	202.117.1.185	TCP	54 53459 → 443 [ACK] Seq=242 Ack=93 Win=65536 Len=0
146	2.302050	202.117.1.185	192.168.123.24	TLSv1	107 Finished
147	2.302358	192.168.123.24	202.117.1.185	TLSv1	113 Change Cipher Spec, Finished

图 3.5: 与 TLS 握手过程相关的分组

在图 3.3 中，不同类型的分组用不同的颜色标出，可以通过点击“视图 -> 着色规则”菜单项查看不同颜色所代表的的分组类型，如图 3.4 所示。根据图 3.4，可以看到图 3.3 捕获的分组中既包括 UDP 分组、TCP 分组、TCP 握手分组和 TCP RST（复位）分组。其中，客户端（192.168.123.64）在与认证网关服务端（202.117.1.185）建立连接的过程中有多个 RST 分组（红色）；产生 RST 的原因很多，不过根据图 3.3 捕获的分组来看是由于服务端的 443 端口未被进程监听或被占用，而客户端又向服务端发送目标端口为 443 的连接请求，这时客户端则会发送 RST 分组以重新建立连接。

## 3.2 不同层重要字段与 TLS 握手过程分析

在过滤栏上输入“ip.addr == 202.117.1.185”过滤掉与认证登录过程不相关的分组，最后确定与 TLS 协议握手过程相关的分组为第 141 号分组到第 147 号分组，如图 3.5 所示。点击 141 号（Client Hello）分组，可以看到该分组中不同网络层的重要字段信息，如图 3.6 所示。

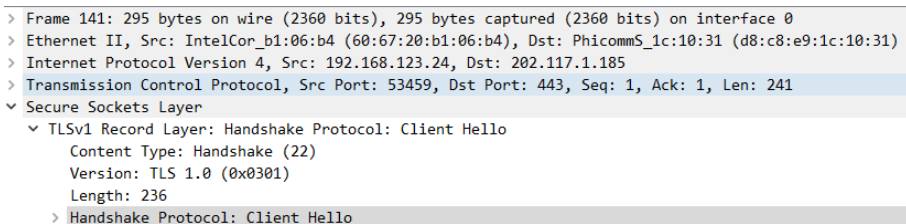
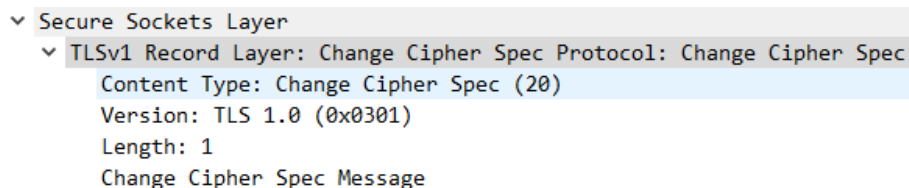


图 3.6: Client Hello 分组中不同层次的重要字段



```

  Secure Sockets Layer
    TLSv1 Record Layer: Change Cipher Spec Protocol: Change Cipher Spec
      Content Type: Change Cipher Spec (20)
      Version: TLS 1.0 (0x0301)
      Length: 1
      Change Cipher Spec Message

```

图 3.7: Change Cipher Spec 分组中 TLS 协议层

根据图 3.6 反映的分组字段信息，可见最上方的物理层字段反映了分组帧长度为 295 个字节。被物理层封装的数据链路层，协议为“Ethernet II（第二代以太网）”，显示客户端的设备为“Intel Core（英特尔酷睿系列核心）”，MAC 地址为 60:67:20:b1:06:b4（十六进制格式）；服务端的设备为 phi 卡，MAC 地址为 d8:c8:e9:1c:10:31。被数据链路层封装的网络层，协议为 IPv4，源地址为 192.168.123.24，目标地址为 202.117.1.185。被网络层封装的传输层，协议为 TCP，源端口为 53459，目标端口为 443，分组序号（Seq）为 1，应答序号（Ack）为 1，分组长度为 241 个字节。重点的 SSL（安全传输层）被封装在 TCP 层中，协议为 TLSv1，其中又封装了长度为 236 个字节的 TLS 记录层，TLS 记录层中又封装了 TLS 握手协议层，而该握手协议层的握手信息为“Client Hello”。该分组逐层封装的过程体现了计算机网络中的分层思想，将一个分组按类型分为不同的层次有利于逐层分析，简化网络模型。

和握手协议一样，TLS 协议中的更改密码协议也是封装在记录协议中的，通过点击 144 号分组可以发现这一点，如图 3.7 所示。通过观察其他 TLS 分组的协议结构，验证了记录协议与握手协议、更改密码协议、应用数据协议等其他子协议的层次关系，即这些子协议都封装在记录协议中。

根据图 3.5 以及各个分组的分层字段信息，可以归纳出本实验中 TLS 协议握手过程如图 3.8 所示。根据图 3.8，TLS 协议的握手过程可分为以下步骤：

1. 客户端向服务端发送 Client Hello 分组，这组分组包含一段会话编号（Session ID）以及一段随机密钥（如图 3.9所示，这是客户端的私钥），期望建立 TLS 连接，开辟临时会话。
2. 服务端回复一段 TCP 应答分组，作为对客户端上一个 TCP 分组的应答。
3. 服务端向客户端发送 Server Hello 分组，这组分组包含一段与 Client Hello 分组中一致的会话编号（Session ID）和一段随机密钥（如图 3.10所示，服务端使用客户端之前发送的私钥给自己的随机密钥加密，这段密钥正是服务端加密后的给客户端共用的公钥），表示同意建立 TLS 连接，并且接着 Server Hello 分组的 Change Cipher Spec 分组中通知客户端使用服务端提供的公钥加密。
4. 客户端对服务端上一个 TCP 分组进行应答。

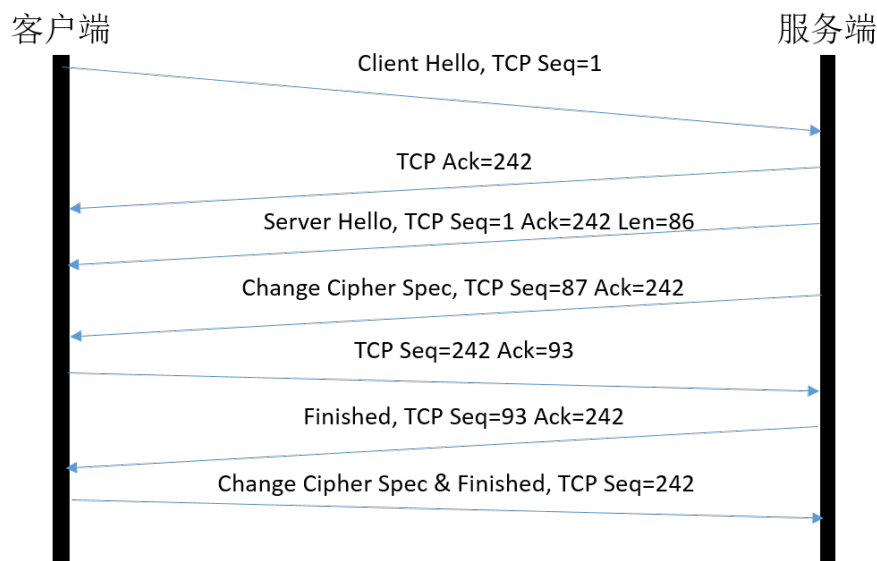


图 3.8: 本实验中 TLS 协议握手过程

```

v Secure Sockets Layer
  v TLSv1 Record Layer: Handshake Protocol: Client Hello
    Content Type: Handshake (22)
    Version: TLS 1.0 (0x0301)
    Length: 236
  v Handshake Protocol: Client Hello
    Handshake Type: Client Hello (1)
    Length: 232
    Version: TLS 1.2 (0x0303)
  v Random: 81ec61d28198114cd5749ab77467d8517afdf72aec90c337...
    GMT Unix Time: Jan 27, 2039 22:46:42.000000000 中国标准时间
    Random Bytes: 8198114cd5749ab77467d8517afdf72aec90c337c5bc7dc6...
    Session ID Length: 32
    Session ID: 5a9d02fb6e1ce96d9d65d6fe7578a81b0f6353661d1f5a02...
  
```

图 3.9: Client Hello 分组 TLS 协议部分

5. 服务端收到客户端应答后，确认客户端已经知道要使用约定好的公钥加密和解密数据，发送 Finished 分组期望结束握手。
6. 客户端收到结束握手分组后，发送 Change Cipher Spec 分组和 Finished 分组，表明已完成密钥的更改，结束握手。

完成 TLS 握手后，客户端与服务端的临时会话便使用来自服务端的相同的密钥进行数据的加密和解密，本次临时会话结束后，TLS 连接关闭，本次临时会话的密钥无法用于下次会话。以上 TLS 协议的握手过程说明 TLS 协议采用面向连接的服务，对应用数据采用对称加密技术。



```

    Secure Sockets Layer
    TLSv1 Record Layer: Handshake Protocol: Server Hello
      Content Type: Handshake (22)
      Version: TLS 1.0 (0x0301)
      Length: 81
    Handshake Protocol: Server Hello
      Handshake Type: Server Hello (2)
      Length: 77
      Version: TLS 1.0 (0x0301)
    Random: 5a9d03429c3b8e6ca96cedf4501e652d2afe5372fd9351e6...
      GMT Unix Time: Mar 5, 2018 16:43:46.000000000 中国标准时间
      Random Bytes: 9c3b8e6ca96cedf4501e652d2afe5372fd9351e6d45d9d4b...
    Session ID Length: 32
    Session ID: 5a9d02fb6e1ce96d9d65d6fe7578a81b0f6353661d1f5a02...

```

图 3.10: Server Hello 分组 TLS 协议部分

148	2.302995	192.168.123.24	202.117.1.185	HTTP	1168	POST /login?service=http%3A%2F%2Fehall.xjtu.edu.cn%2Famp-auth-adapter%2FloginSuccess.
149	2.305328	202.117.1.185	192.168.123.24	TCP	54	443 → 53459 [ACK] Seq=146 Ack=1415 Win=9216 Len=0
156	2.338813	202.117.1.185	192.168.123.24	TCP	1514	443 → 53459 [ACK] Seq=146 Ack=1415 Win=9216 Len=1460 [TCP segment of a reassembled P.
157	2.338813	202.117.1.185	192.168.123.24	HTTP	1095	HTTP/1.1 200 OK (text/html)
158	2.338903	192.168.123.24	202.117.1.185	TCP	54	53459 → 443 [ACK] Seq=1415 Ack=2647 Win=65536 Len=0
148	2.302995	192.168.123.24	202.117.1.185	HTTP	1168	POST /login?service=http%3A%2F%2Fehall.xjtu.edu.cn%2Famp-auth-adapter%2FloginSuccess.
149	2.305328	202.117.1.185	192.168.123.24	TCP	54	443 → 53459 [ACK] Seq=146 Ack=1415 Win=9216 Len=0
156	2.338813	202.117.1.185	192.168.123.24	TCP	1514	443 → 53459 [ACK] Seq=146 Ack=1415 Win=9216 Len=1460 [TCP segment of a reassembled P.
157	2.338813	202.117.1.185	192.168.123.24	HTTP	1095	HTTP/1.1 200 OK (text/html)

图 3.11: 解密后分组（局部）

### 3.3 表单解密

为了充分验证本次实验中 Wireshark 软件真的抓到了登录过程中客户端向服务端提交的表单，下面将对捕获到的分组中 TLS 协议应用数据层的内容进行解密，以得到表单明文。完成解密后的分组如图 3.11 所示。其中，标记为绿色，协议为 HTTP 的分组正是解密后的 TLS 应用数据层内容。

在解密之前，TLS 应用数据层的内容是加密后的乱码，无法看到明文。通过上一小节对 TLS 握手过程的分析，得知这些内容是使用服务端提供的公钥加密的，因此客户端上应该保存有这份公钥，只要拿到这份公钥就可以对内容进行解密。事实上，一般的浏览器都会将 TLS 握手后得到的公钥保存在本地，通过设置环境变量可以改变公钥存放的位置，如图 3.12 所示。然后在 Wireshark 菜单栏中依次点击“编辑 -> 首选项 -> Protocol->SSL”，在出现的窗口中找到“(Pre)-Master-Secret log filename”下方的文本框，输入公钥保存的位置，如图 3.13 所示。

完成以上设置后再开始抓包。得到分组后，右键点击需要解密的分组，在弹出的菜单中点击“解码为...”，根据分组字段内容设置好相应的参数后即可完成解密。本实验中最终解密得到的客户端向服务端提交的登录表单的内容如图 3.14 所示，表单中包含用户名、密码等信息。

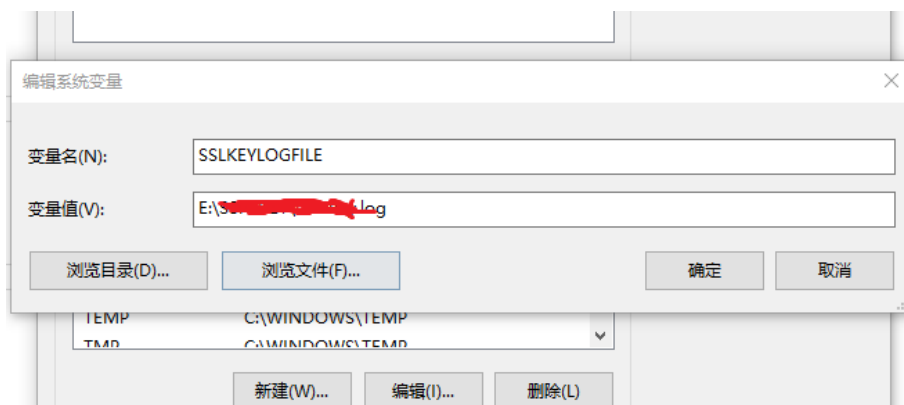


图 3.12: 通过设置环境变量改变浏览器保存公钥的位置

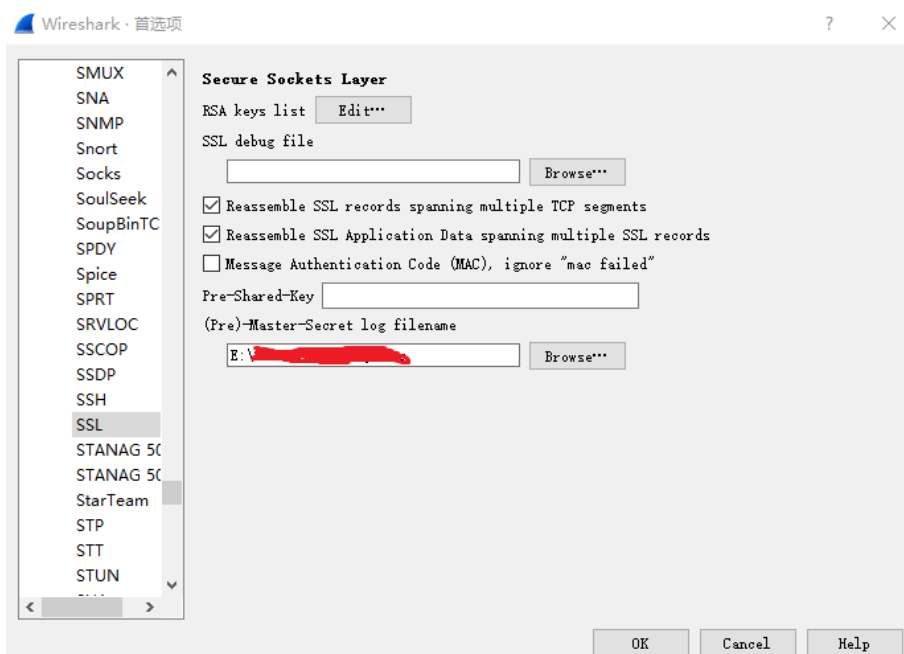


图 3.13: 在 Wireshark 中设置公钥保存位置

```
> Frame 148: 1168 bytes on wire (9344 bits), 1168 bytes captured (9344 bits) on interface 0
> Ethernet II, Src: IntelCor_b1:06:b4 (60:67:20:b1:06:b4), Dst: PhicommS_1c:10:31 (d8:c8:e9:1c:10:31)
> Internet Protocol Version 4, Src: 192.168.123.24, Dst: 202.117.1.185
> Transmission Control Protocol, Src Port: 53459, Dst Port: 443, Seq: 301, Ack: 146, Len: 1114
> Secure Sockets Layer
> [2 Reassembled SSL segments (1042 bytes): #148(1), #148(1041)]
> Hypertext Transfer Protocol
> HTML Form URL Encoded: application/x-www-form-urlencoded
  > Form item: "username" = "[redacted]"
  > Form item: "password" = "[redacted]"
  > Form item: "code" = ""
  > Form item: "lt" = "LT-88827-u090nqCI1nmXZw1fr14ibW0UPPeYSS"
  > Form item: "execution" = "e2s1"
  > Form item: "_eventId" = "submit"
  > Form item: "submit" = "登录"
```

图 3.14: 解密后登录表单的内容

---

## 参考文献

- [1] 李天目. Ssl/tls 协议的安全分析与改进. 信息安全, 1:51–54, 2005.
- [2] CSDN. TLS 握手协商流程解析. <http://blog.csdn.net/phunxm/article/details/72853552>, 2017. [Online; accessed 14-Mar-2018].
- [3] S Turner. Transport layer security. *IEEE Internet Computing*, 18(6):60–63, 2014.