山东大学 计算机 学院

计算机网络 课程实验报告

学号：202400130240	姓名： 贾宗翰	班级： 24.6
实验题目：Wireshark Lab: TLS v8.1		
实验学时：2h	实验日期： 2025.12.2	
实验目的： 研究传输层安全性（称为 TLS）以及 TLS 提供的身份验证、数据完整性和机密性服务的各个方面		
硬件环境： 联想拯救者		
软件环境： Wireshark，edge 浏览器		

实验步骤与内容：

1. 在 TLS 会话中捕获数据包

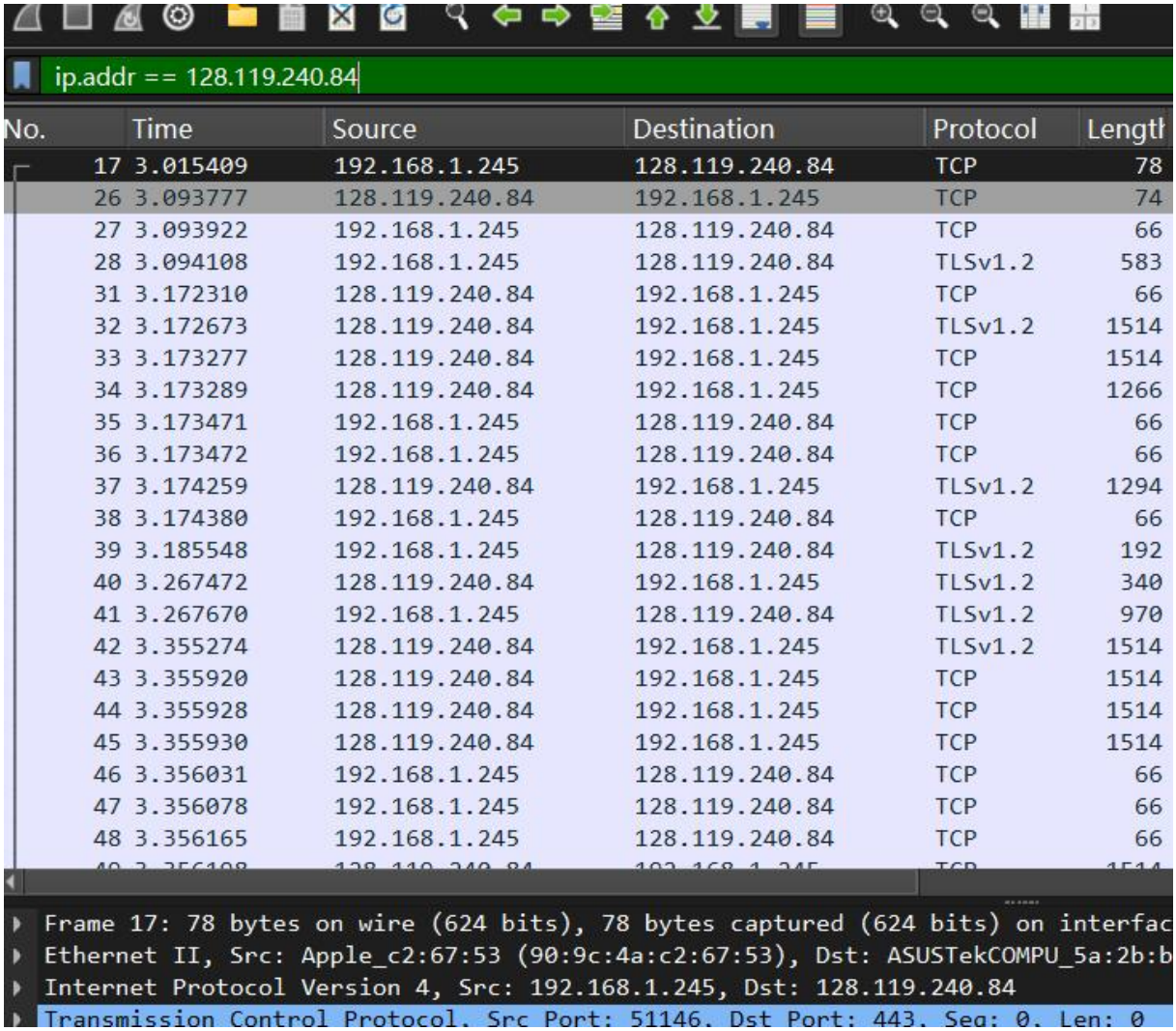
本实验的第一步是捕获 TLS 会话中的数据包。为此，您应该启动 Wireshark 并开始数据包捕获，使用您选择的浏览器从 https://www.cics.umass.edu 检索主页，然后停止 Wireshark 数据包捕获。“http”后面的“s”将导致使用安全超文本传输协议（HTTPS）- HTTP 的扩展 - 安全地从 www.cics.umass.edu 检索主页。此处，“安全”意味着 www.cics.umass.edu 服务器将由 Web 浏览器进行身份验证，客户端 HTTP GET 请求的传输和服务器的回复将被加密，并且所有消息内容的完整性都将进行加密验证。当然，计算机科学系网页的身份验证、完整性和加密可能不像 Internet 商务和银行网站那样重要，但在所有情况下都使用相同的 TLS 协议和 TLS 消息。

如果您无法在实时网络连接上运行 Wireshark，则可以下载在作者的其中一台计算机上按照上述步骤作时捕获的数据包跟踪 5。此外，当您浏览以下问题时，即使您已捕获并使用自己的跟踪记录，您也可能会发现下载此跟踪记录也很有价值。

2. 首先查看捕获的跟踪

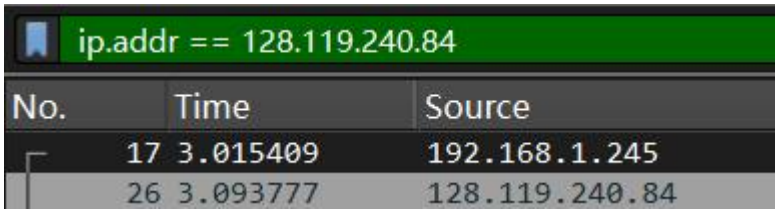
我们首先设置 Wireshark 的显示，以便仅显示进出 IP 地址为 128.119.240.84 的 www.cics.umass.edu 的数据包。为此，请在 Wireshark 的显示过滤器窗口中输入 ip.addr == 128.119.240.84。您的屏幕应类似于图 1 中所示的屏幕。

请务必记住，以太网帧（包含包含 TCP 段的 IP 数据报）可能包含一个或多个 TLS 记录。（这与 HTTP 非常不同，后者的每个帧都包含一条完整的 HTTP 消息或一条 HTTP 消息的一部分。此外，TLS 记录可能无法完全放入以太网帧中，在这种情况下，需要多个帧来承载记录。



在回答以下问题 6 时，您可以使用自己的实时跟踪，也可以使用 Wireshark 捕获的数据包文件 TLS-wireshark-tracel http://gaia.cs.umass.edu/wireshark-labs/wireshark-traces-8.1.zip 我们之前已经说过，HTTPS 实现“通过”TCP 运行的 TLS。这意味着必须先在浏览器和 Web 服务器之间建立 TCP 连接以进行 www.cics.umass.edu，然后才能交换 TLS 和 HTTP 消息，就像我们在普通（非 TLS）HTTP 协议中看到的那样。

1. 跟踪中包含初始 TCP SYN 消息的数据包编号是多少？（我们所说的“数据包编号”是指 Wireshark 显示屏左侧“No.”列中的数字，而不是 TCP 段本身中的序列号）。



17

2. TCP 连接是在从客户端发送到服务器的第一条 TLS 消息之前还是之后设置的？

```
TCP          78 51146 → 443 [SYN] Seq=0 Win=65535 Len=0 M
TCP          74 443 → 51146 [SYN, ACK] Seq=0 Ack=1 Win=28
TCP          66 51146 → 443 [ACK] Seq=1 Ack=1 Win=131712
TLSv1.2     583 Client Hello (SNI=www.cics.umass.edu)
TCP          66 443 → 51146 [ACK] Seq=1 Ack=518 Win=30080
TLSv1.2    1514 Server Hello
TCP        1514 443 → 51146 [ACK] Seq=1449 Ack=518 Win=30
TCP        1266 443 → 51146 [PSH, ACK] Seq=2897 Ack=518 W
```

分析内容，TCP 连接是在发送第一条消息之前建立的。

3. TLS Handshake： Client Hello 消息

我们了解到，如图 2 所示，客户端-服务器 TLS 握手从客户端发送 TLS Client Hello 消息开始。

3. 跟踪中包含 TLS 客户端 Hello 消息的数据包编号是多少？

```
28 3.094108    192.168.1.245    128.119.240.84    TLSv1.2    583 Client Hello (SNI=www.cics.umass.ed
```

显然是 28.

4. 您的客户端运行的 TLS 版本是什么，如客户端 Hello 消息中声明的那样？

见上，是 TLSv1.2

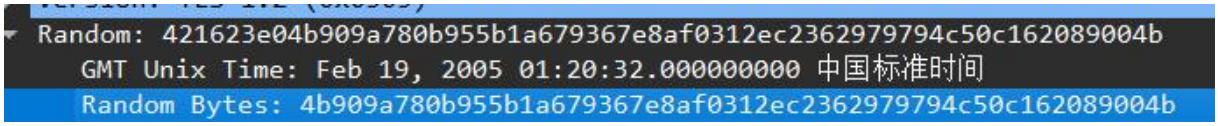
5. 您的客户端支持多少 Cipher 套件，如 Client Hello 消息中声明的那样？密码套件是一组相关的加密算法，用于确定如何派生会话密钥，以及如何通过 HMAC 算法对数据进行加密和数字签名。

17 个：

```
Cipher Suites (17 suites)
    Cipher Suite: TLS_AES_128_GCM_SHA256 (0x1301)
    Cipher Suite: TLS_CHACHA20_POLY1305_SHA256 (0x1303)
    Cipher Suite: TLS_AES_256_GCM_SHA384 (0x1302)
    Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 (0xc02b)
    Cipher Suite: TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (0xc02f)
    Cipher Suite: TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256 (0xcca9
    Cipher Suite: TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256 (0xcca8)
    Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 (0xc02c)
    Cipher Suite: TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (0xc030)
    Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA (0xc00a)
    Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA (0xc009)
    Cipher Suite: TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA (0xc013)
    Cipher Suite: TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (0xc014)
    Cipher Suite: TLS_RSA_WITH_AES_128_GCM_SHA256 (0x009c)
    Cipher Suite: TLS_RSA_WITH_AES_256_GCM_SHA384 (0x009d)
    Cipher Suite: TLS_RSA_WITH_AES_128_CBC_SHA (0x002f)
    Cipher Suite: TLS_RSA_WITH_AES_256_CBC_SHA (0x0035)
```

6. 您的客户端在 Client Hello 消息中生成一串“随机字节”并将其发送到服务器。Client Hello 消息的随机字节字段中的前两个十六进制数字是什么？输入两个十六进制数

字（十六进制数字之间没有空格，也没有任何前导“0x”，必要时使用小写字母）。提示：请小心地完全深入 Random 字段以找到 Random Bytes 子字段（不要考虑 Random 的 GMT UNIX Time 子字段）。



是 4b。

7. Client Hello 消息中的“random bytes”字段有什么用途？注意：您必须进行一些搜索和阅读才能获得这个问题的答案;参见 Section 8.6 和 RFC 5246 （特别是 RFC 5246 中的 Section 8.1）。

建立密钥保证 TLS 连接的唯一性。

3. TLS 握手：服务器 Hello 消息

接下来，我们来看看 TLS 握手的第二步，即 TLS 服务器 Hello 消息，该消息是为响应之前的 TLS 客户端 Hello 消息而发送的。

8. 跟踪中包含 TLS 服务器 Hello 消息的数据包编号是多少？

32

9. 服务器从之前的 Client Hello 消息中提供的密码套件中选择了哪个密码套件？

10. Server Hello 消息是否包含随机字节，类似于 Client Hello 消息包含随机字节的方式？如果是这样，它们的目的是什么？

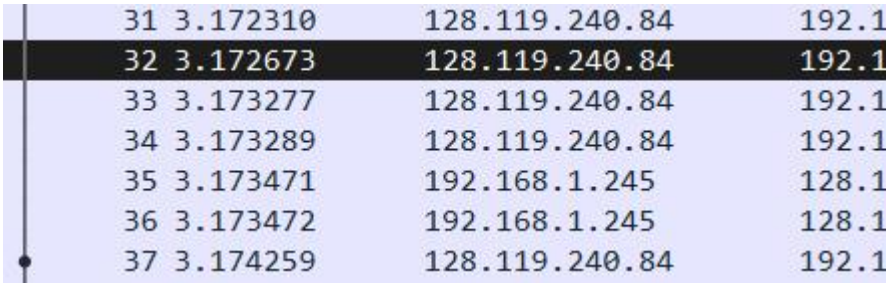
目的：与刚才的 random byte 一起保证此 TLS 的唯一性（密钥）

为了响应初始 Client Hello 消息，服务器还会发回两个重要的附加信息（可能包含在与 Server Hello 消息的初始部分不同的数据包中）。返回的第一条重要信息是由受信任的证书颁发机构 （CA;请参阅正文中的第 8.3 节） 颁发的证书，该证书将 Web 服务器的公钥（和其他信息）绑定到该 Web 服务器的身份。您的客户端可以将服务器的公钥用于多种不同的目的，包括获取对称密钥以加密通过此 HTTPS/TLS/TCP 会话发送的数据以及生成 HMAC 数字签名。当然，您可以在 Wireshark 中查看服务器的证书。您还可以在 www.cs.umass.edu 服务器响应您的请求后在 Web 浏览器中查看服务器的证书（并且证书的格式也更漂亮）——方法如下。

从服务器返回的第二条附加信息是对正在交换的数据（在本例中为 HTTP 消息）进行对称加密所需的参数。

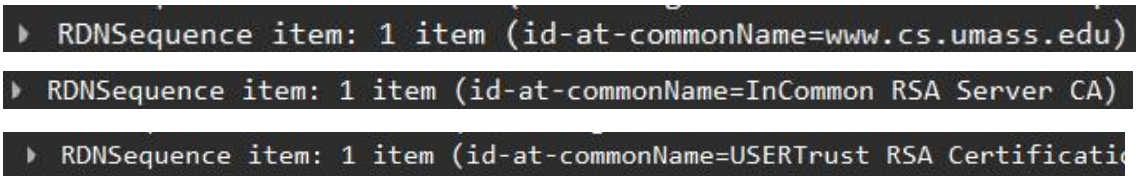
让我们更深入地研究一下公钥证书。

11. 包含 www.cics.umass.edu 服务器（实际上是 www.cs.umass.edu 服务器）的公钥证书的 TLS 消息部分的跟踪中的数据包编号是多少？

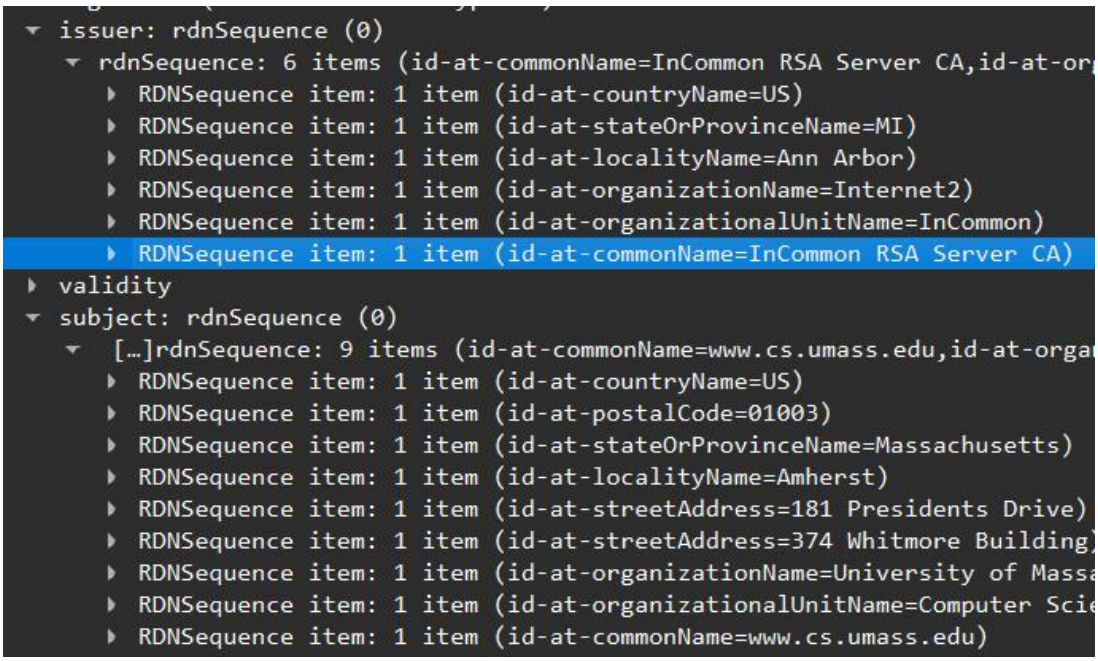


观察得知是 37.

12. 服务器可能会返回多个证书。如果返回多个证书，所有这些证书是否都用于 www.cs.umass.edu？如果不是全部用于 www.cs.umass.edu，那么这些其他证书适用于谁？您可以通过检查重新调整的证书中的 id-at-commonName 字段来确定证书的适用对象。

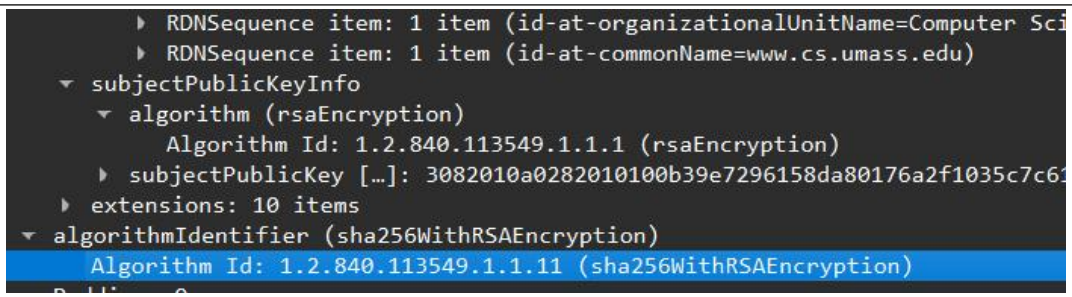


13. 颁发 ID 为 at-commonName=www.cs.umass.edu 的证书的证书颁发机构的名称是什么？



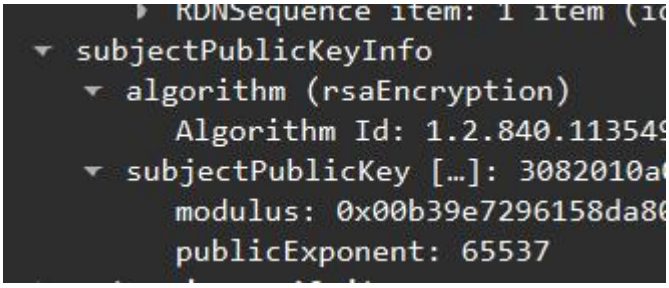
RDNSequence item: 1 item (id-at-commonName=InCommon RSA Server CA)

14. CA 使用什么数字签名算法来签署此证书？提示：此信息可以在 www.cs.umass.edu 证书的 SignedCertificate 字段的 signature 子字段中找到。



Algorithm Id: 1.2.840.113549.1.1.11 (sha256WithRSAEncryption)

15. 我们来看看真正的公钥是什么样子的！www.cics.umass.edu 使用的公钥模数的前四个十六进制数字是多少？输入四个十六进制数字（十六进制数字之间没有空格，没有任何前导 '0x'，需要时使用小写字母，并在 '0x' 后包括任何前导 0）。提示：此信息可以在 www.cs.umass.edu 证书的 SignedCertificate 字段的 subjectPublicKeyInfo 子字段中找到。

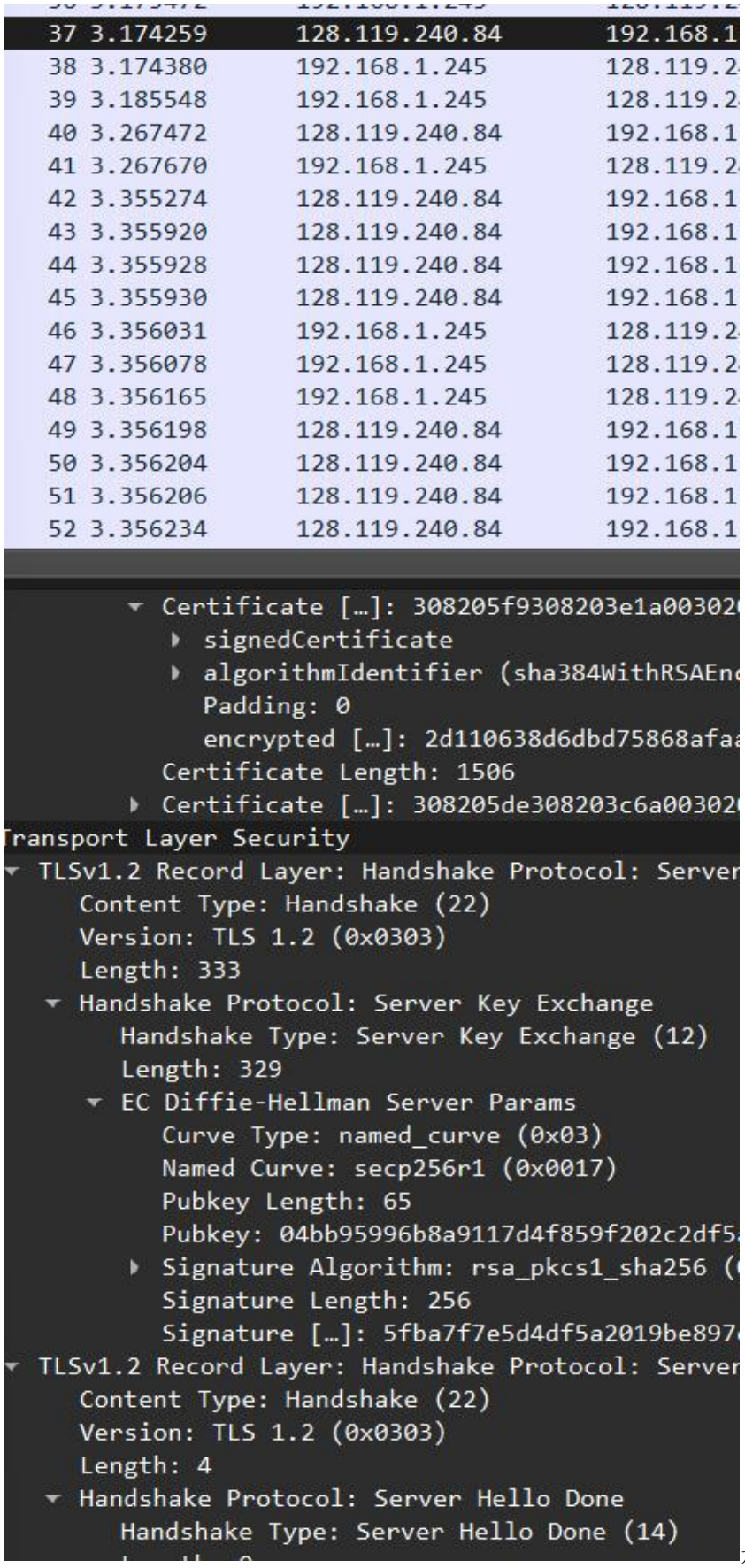


是 00b3.

16. 查看跟踪以查找客户端与 CA 之间的消息，以获取 CA 的公钥信息，以便客户端可以验证服务器发送的 CA 签名证书是否确实有效，并且未被伪造或更改。您是否在跟踪中看到此类消息？如果是这样，从您的客户端发送到 CA 的第一个数据包的跟踪中的编号是多少？如果没有，请解释客户端未联系 CA 的原因。Server Hello 消息始终由显式 Server Hello Done 记录终止。

没有找到与 CA 通信的数据包，分析原因在于客户端会内置信任 CA 的证书信息，所以不需要通信。

17. 包含服务器 Hello Done TLS 记录的 TLS 消息部分的跟踪中的数据包编号是多少？



是 37.

4. TLS 握手：结束握手

交换 Hello 消息交换后，客户端使用自己的公钥信息、声明（通过更改密码规范记录）

来响应 TLS 服务器 Hello 消息，声明（通过更改密码规范记录）所有进一步的通信都将通过协商的算法和密钥进行加密，以及包含加密信息的加密握手消息记录（例如 此握手期间交换的所有消息的加密哈希值），以防止中间人重播攻击。

18. 包含从客户端发送到服务器的公钥信息、更改密码规范和加密握手消息的 TLS 消息的跟踪中的数据包编号是多少？

显然是 39.

19. 客户端是否向服务器提供自己的 CA 签名的公钥证书？如果是这样，则跟踪中包含客户端证书的数据包编号是多少？

客户端没有向服务器提供自己的 CA 签名的公钥证书。

最后，服务器使用自己的声明（通过 Change Cipher Spec 记录）响应客户端，该声明表明所有进一步的通信都将通过协商的算法和密钥进行加密，以及 Encrypted Handshake Message 记录，该记录还包含加密信息（例如，在此握手期间交换的所有消息的加密哈希值），以防止中间人重放攻击。

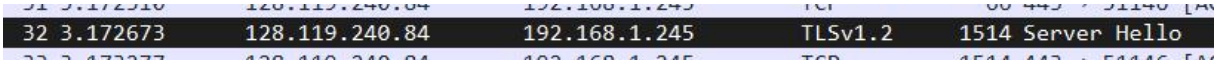
5. 应用程序数据

TLS 握手完成后，加密的应用程序数据可以开始通过 HTTP-over-TLS-over-TCP 连接流动。当然，由于这些数据是加密的，我们实际上无法检查这些加密消息的内容（这不就是重点吗！但我们仍然可以问一些关于加密流量的最后问题。

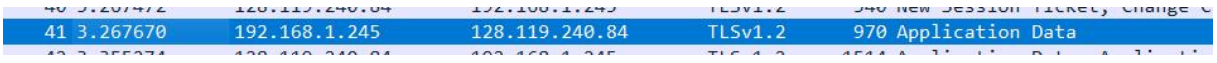
20. 客户端和服务器使用哪种对称密钥加密算法来加密应用程序数据（在本例中为 HTTP 消息）？

21. 这种对称密钥加密算法最终决定并声明在哪条 TLS 消息中？

参考上文，得知在编号 32 的消息中。



22. 对于将应用程序数据从客户端传输到服务器的第一条加密消息，跟踪中的数据包编号是多少？



41.

23. 鉴于此跟踪是通过获取 www.cics.umass.edu 的主页生成的，您认为此加密应用程序数据的内容是什么？

应该是 http 相关的信息，比如 http Get 请求。

最后，我们来看看客户端如何关闭 TLS 连接。您必须深入研究才能回答这个问题（请参阅前面的脚注参考文献）。

24. 哪个数据包编号包含关闭 TLS 连接的客户端到服务器 TLS 消息？由于 TLS 消息在我们的 Wireshark 跟踪中是加密的，因此我们实际上无法查看 TLS 消息的内部，因此我们必须在此处进行有根据的猜测。

猜测在 358 终止：

358	4.428470	192.168.1.245	128.119.240.84	TLSv1.2	97 Encrypted Alert
393	4.503442	128.119.240.84	192.168.1.245	TLSv1.2	1514 Application Data
394	4.503516	192.168.1.245	128.119.240.84	TCP	54 51146 → 443 [RST] Seq=3213 Win=0 Len=0
395	4.504172	128.119.240.84	192.168.1.245	TCP	1514 443 → 51146 [ACK] Seq=35007 Ack=3213 Win=354
396	4.504176	128.119.240.84	192.168.1.245	TCP	1514 443 → 51146 [ACK] Seq=36455 Ack=3213 Win=354
397	4.504177	128.119.240.84	192.168.1.245	TLSv1.2	997 Application Data
398	4.504240	192.168.1.245	128.119.240.84	TCP	54 51146 → 443 [RST] Seq=3213 Win=0 Len=0
399	4.504240	192.168.1.245	128.119.240.84	TCP	54 51146 → 443 [RST] Seq=3213 Win=0 Len=0
400	4.504240	192.168.1.245	128.119.240.84	TCP	54 51146 → 443 [RST] Seq=3213 Win=0 Len=0
403	4.506718	128.119.240.84	192.168.1.245	TLSv1.2	97 Encrypted Alert
404	4.506768	192.168.1.245	128.119.240.84	TCP	54 51146 → 443 [RST] Seq=3245 Win=0 Len=0
405	4.507288	128.119.240.84	192.168.1.245	TCP	66 443 → 51146 [FIN, ACK] Seq=38865 Ack=3245 Wi
406	4.507335	192.168.1.245	128.119.240.84	TCP	54 51146 → 443 [RST] Seq=3245 Win=0 Len=0
468	4.534193	192.168.1.245	128.119.240.84	TCP	78 [TCP Retransmission] 51156 → 443 [SYN] Seq=0

可见 358 之后，进行了终止部分的执行。但是不确定这里是正常结束还是其他，因为分析此数据包：

```
TLSv1.2 Record Layer: Encrypted Alert
    Content Type: Alert (21)
    Version: TLS 1.2 (0x0303)
    Length: 26
    Alert Message: Encrypted Alert
```

，并没有获得 close notify 消息。

结论分析与体会：

问题和解答见上。通过此次实验，我对于 TLS 的理解更进一步