

The exercises are designed for students to finish in an individual capacity. The exercises are not designed to be completed in tutorial sessions but rather to give you some tasks and a starting point to continue and complete on your own.

1 TLS

1.1 Lab Tasks

A TLS handshake takes place whenever a user navigates to a website over HTTPS; it involves multiple steps, as the client and server exchange the information necessary for completing the handshake and making further conversation possible. In this lab, we will capture a TLS handshake using Wireshark and analyse the details of the connection establishment.

Open SecureCorp network configuration in GNS3 and start all nodes. You can use one of the workstations to perform the lab tasks.

- **Browse an HTTPS website with lynx**

Start packet capture on the link between **Internal-Client** and **Switch3**, and browse to `monash.edu`:

```
cd
lynx https://www.monash.edu
```

Wireshark should have captured the lynx traffic.

- **Client Hello**

Open Wireshark and find "Client Hello" packet. TLS wraps all traffic in "records" of different types. We see that the first byte out of our browser is the hex byte `0x16 = 22` which means that this is a "handshake" record:

No.	Time	Source	Destination	Protocol	Length	Info
7	10.848223	10.10.5.50	8.8.8.8	DNS	74	Standard query 0x6a72 A
8	10.848271	10.10.5.50	8.8.8.8	DNS	74	Standard query 0x6071 A
9	10.891315	8.8.8.8	10.10.5.50	DNS	193	Standard query response
10	10.891748	8.8.8.8	10.10.5.50	DNS	124	Standard query response
11	10.893175	10.10.5.50	202.9.95.188	TCP	74	39110 → 443 [SYN] Seq=
12	10.922687	202.9.95.188	10.10.5.50	TCP	58	443 → 39110 [SYN, ACK]
13	10.932942	10.10.5.50	202.9.95.188	TCP	54	39110 → 443 [ACK] Seq=
14	10.955895	10.10.5.50	202.9.95.188	TLSv1.2	414	Client Hello
15	10.960500	202.9.95.188	10.10.5.50	TCP	54	443 → 39110 [ACK] Seq=
16	10.988711	202.9.95.188	10.10.5.50	TLSv1.2	1474	Server Hello
17	10.988772	202.9.95.188	10.10.5.50	TCP	1474	443 → 39110 [ACK] Seq=
18	10.989001	10.10.5.50	202.9.95.188	TCP	54	39110 → 443 [ACK] Seq=
19	10.989024	10.10.5.50	202.9.95.188	TCP	54	39110 → 443 [ACK] Seq=
20	10.993463	202.9.95.188	10.10.5.50	TLSv1.2	1474	Certificate [TCP segment
21	10.994213	10.10.5.50	202.9.95.188	TCP	54	39110 → 443 [ACK] Seq=

▶ Frame 14: 414 bytes on wire (3312 bits), 414 bytes captured (3312 bits) on interface -, id 0
 ▶ Ethernet II, Src: 46:b2:4c:85:c3:cd (46:b2:4c:85:c3:cd), Dst: 0c:97:81:3c:f2:00 (0c:97:81:3c:f2:00)
 ▶ Internet Protocol Version 4, Src: 10.10.5.50, Dst: 202.9.95.188
 ▶ Transmission Control Protocol, Src Port: 39110, Dst Port: 443, Seq: 1, Ack: 1, Len: 360
 ▶ Transport Layer Security

- TLSv1.2 Record Layer: Handshake Protocol: Client Hello
 Content Type: Handshake (22)
 Version: TLS 1.2 (0x0303)
 Length: 355
 - Handshake Protocol: Client Hello
 Handshake Type: Client Hello (1)
 Length: 351
 Version: TLS 1.2 (0x0303)
 Random: 70153b7e0e992dd0e5d59200c2aadd55dc323d51a35129e0...
 Session ID Length: 0
 Cipher Suites Length: 58
 Cipher Suites (29 suites)
 Compression Methods Length: 1
 Compression Methods (1 method)
 Extensions Length: 252
 Extension: status_request (len=5)
 Extension: supported_groups (len=22)

0030 fa f0 92 ad 00 00 16 03 03 01 63 01 00 01 5f 03
 0040 03 70 15 3b 7e 0e 99 2d d0 e5 d5 92 00 c2 aa dd
 0050 55 dc 32 3d 51 a3 51 29 e0 67 48 27 1a f5 ba fe
 0060 a3 00 00 3a 13 02 13 03 13 01 13 04 c0 2c cc a9

First byte
 Next two bytes (0x0303)

Figure 1: Client Hello

The next two bytes are 0x0303 which indicate that this is a version 3.3 record which shows that TLS 1.2 is essentially SSL 3.3. The handshake record is broken out into several messages. The first is our “Client Hello” message (0x01). There are a few important things here:

1. Random

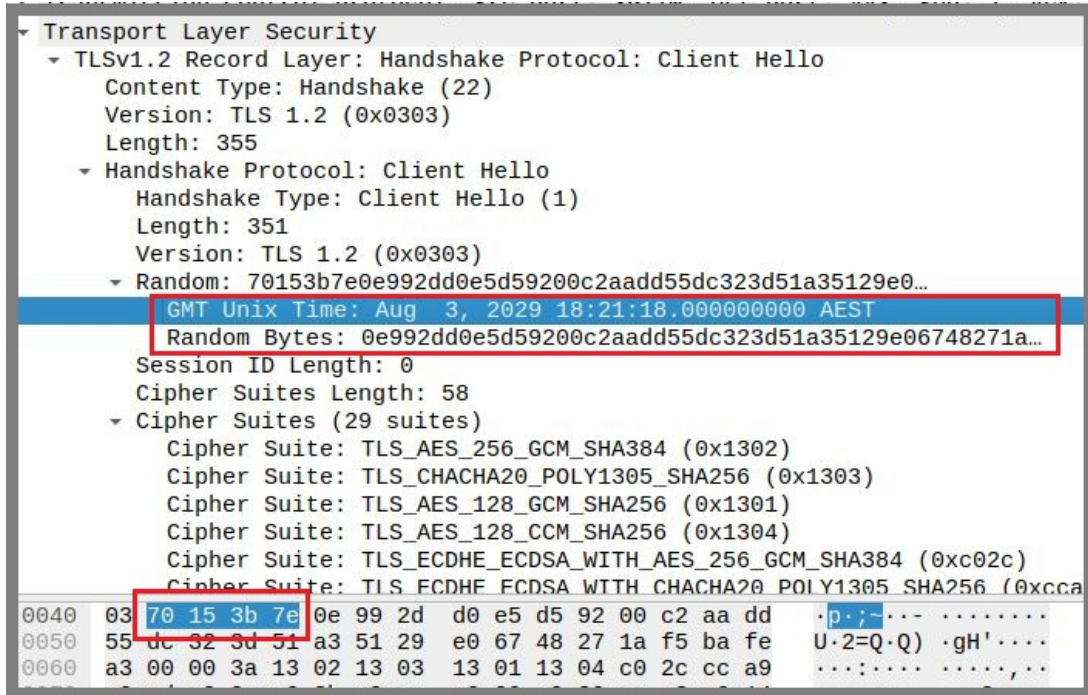


Figure 2: Random:

There are four bytes representing the current Coordinated Universal Time (UTC) in the Unix epoch format, which is the number of seconds since January 1, 1970. In this case, 0x70153b7e. It's followed by 28 random bytes. This will be used later on.

2. Cipher Suites:

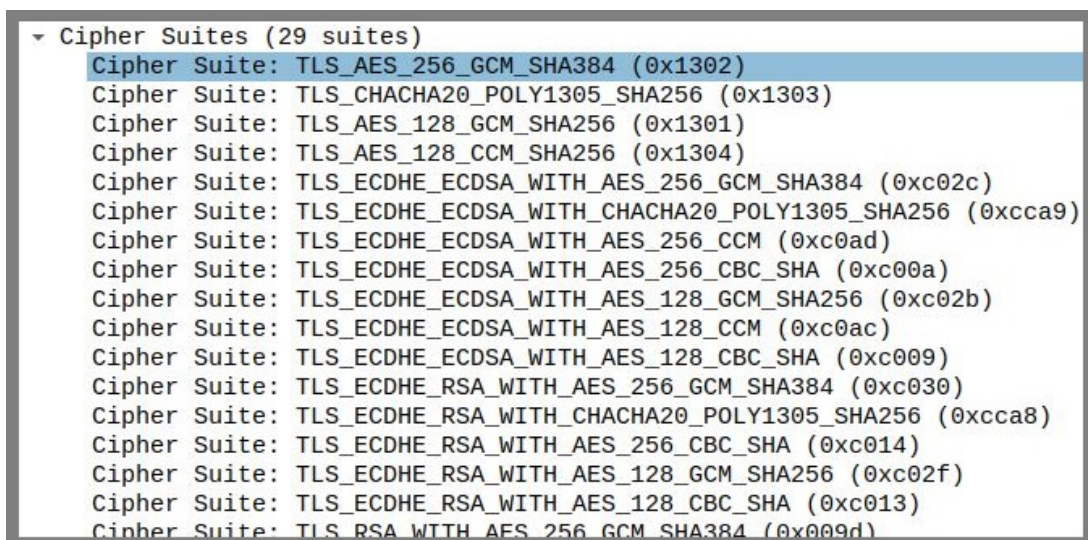


Figure 3: Random

This is a list of all of the encryption algorithms that the browser is willing to support. Its top pick is a very strong choice of TLS_AES_256_GCM_SHA384 followed by 28 others that it's willing to accept.

3. server_name extension:

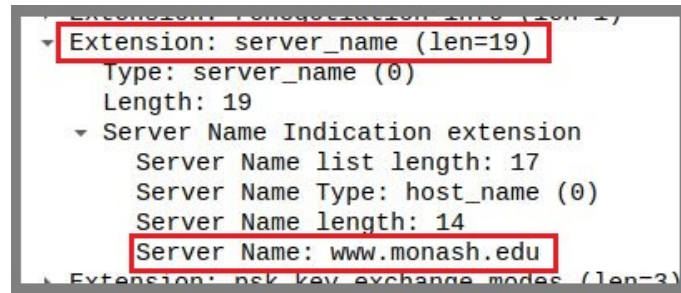


Figure 4: Random

This is a way to tell monash.edu that our browser is trying to reach <https://www.monash.edu>. This is really convenient because our TLS handshake occurs long before any HTTP traffic. HTTP has a “Host” header which allows hosting companies to host hundreds of websites onto a single IP address. SSL has traditionally required a different IP for each site, but this extension allows the server to respond with the appropriate certificate that the browser is looking for.

4. Session ID

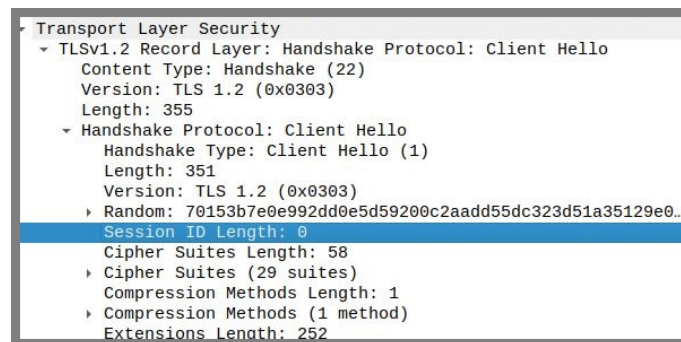


Figure 5: Session ID

Here it’s empty/null. If we had previously connected to Monash a few seconds ago, we could potentially resume a session and avoid a full handshake.

- **Server Hello**

Monash replies with a handshake record. The record has version bytes of 0x0303 meaning that Monash agreed to our request to use TLS 1.2. This record has three sub-messages with some interesting data:

1. “Server Hello” Message (2)

No.	Time	Source	Destination	Protocol	Length	Info
10	10.891748	8.8.8.8	10.10.5.50	DNS	124	Standard query r
11	10.893175	10.10.5.50	202.9.95.188	TCP	74	39110 → 443 [SYN
12	10.922687	202.9.95.188	10.10.5.50	TCP	58	443 → 39110 [SYN
13	10.932942	10.10.5.50	202.9.95.188	TCP	54	39110 → 443 [ACK
14	10.955895	10.10.5.50	202.9.95.188	TLSv1.2	414	Client Hello
15	10.960500	202.9.95.188	10.10.5.50	TCP	54	443 → 39110 [ACK
16	10.988711	202.9.95.188	10.10.5.50	TLSv1.2	1474	Server Hello
17	10.988772	202.9.95.188	10.10.5.50	TCP	1474	443 → 39110 [ACK
18	10.989001	10.10.5.50	202.9.95.188	TCP	54	39110 → 443 [ACK

Frame 16: 1474 bytes on wire (11792 bits), 1474 bytes captured (11792 bits) on interface -, id 0
Ethernet II, Src: 0c:97:81:3c:f2:00 (0c:97:81:3c:f2:00), Dst: 46:b2:4c:85:c3:cd (46:b2:4c:85:c3:cd)
Internet Protocol Version 4, Src: 202.9.95.188, Dst: 10.10.5.50
Transmission Control Protocol, Src Port: 443, Dst Port: 39110, Seq: 1, Ack: 361, Len: 1420
Transport Layer Security
TLSv1.2 Record Layer: Handshake Protocol: Server Hello
Content Type: Handshake (22)
Version: TLS 1.2 (0x0303)
Length: 65
Handshake Protocol: Server Hello
Handshake Type: Server Hello (2)
Length: 61
Version: TLS 1.2 (0x0303)
Random: 1af19dfc09e91dae8693a28d02e0f219b3cbfcb01c5a6b52...
GMT Unix Time: Apr 29, 1984 09:17:16.000000000 AEST
Random Bytes: 09e91dae8693a28d02e0f219b3cbfcb01c5a6b528ea76212...
Session ID Length: 0
Cipher Suite: TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (0xc02f)
Compression Method: null (0)
Extensions Length: 21
Extension: renegotiation_info (len=1)
Type: renegotiation_info (65281)
Length: 1
Renegotiation Info extension
Extension: ec_point_formats (len=4)
Type: ec_point_formats (11)
Length: 4

Figure 6: Server Hello

Of 29 cipher suites we offered, Monash didn't pick our first choice, instead offered to use TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256. So we will be using:

- **ECDHE**: Key exchange with Ephemeral Elliptic Curve Diffie-Hellman
- **RSA**: Signature with RSA (Rivest–Shamir–Adleman)
- **AES_128**: 128-bit Encryption with Advanced Encryption Standard
- **GCM**: AES Mode of Operation with Galois/Counter Mode (GCM)
- **SHA256**: SHA256 hash function to verify the contents of the message

2. Certificate Message

The "Server Hello" is not yet done. Find "Certificate" packet which is also part of TLS handshake.

No.	Time	Source	Destination	Protocol	Length	Info
19	10.989024	10.10.5.50	202.9.95.188	TCP	54	39110 → 443 [ACK] Seq=361 Ack=2841 Win=63900 Len=0
20	10.993403	202.9.95.188	10.10.5.50	TLSv1.2	1474	Certificate [TCP segment of a reassembled PDU]
21	10.994213	10.10.5.50	202.9.95.188	TCP	54	39110 → 443 [ACK] Seq=361 Ack=4261 Win=63900 Len=0
22	10.995056	202.9.95.188	10.10.5.50	TLSv1.2	1474	Certificate Status [TCP segment of a reassembled PDU]

Frame 20: 1474 bytes on wire (11792 bits), 1474 bytes captured (11792 bits) on interface -, id 0
Ethernet II, Src: 0c:97:81:3c:f2:00 (0c:97:81:3c:f2:00), Dst: 46:b2:4c:85:c3:cd (46:b2:4c:85:c3:cd)
Internet Protocol Version 4, Src: 202.9.95.188, Dst: 10.10.5.50
Transmission Control Protocol, Src Port: 443, Dst Port: 39110, Seq: 2841, Ack: 361, Len: 1420
[3 Reassembled TCP Segments (3555 bytes): #16(1350), #17(1420), #20(785)]
Transport Layer Security
TLSv1.2 Record Layer: Handshake Protocol: Certificate
Content Type: Handshake (22)
Version: TLS 1.2 (0x0303)
Length: 3550
Handshake Protocol: Certificate
Handshake Type: Certificate (11)
Length: 3546
Certificates Length: 3543
Certificates (3543 bytes)
Certificate Length: 1826
Certificate: 3082071e30820506a00302010202143ba4347a03e1983015... (id-at-commonName=monash.edu,id-at-organizationName=Monash University, id-at-commonName=QuoVadis Global SSL ICA G3,id-at-organizationName=QuoVadis)
Certificate Length: 1711
Certificate: 308206ab30820493a00302010202142d2c802018b7907c4d... (id-at-commonName=QuoVadis Global SSL ICA G3,id-at-organizationName=QuoVadis)

Figure 7: Certificate

Notice that the certificate size is 3543 bytes, but the packet size is 1474 bytes. The certificate is

taking more than one packet and Wireshark is showing us the certificate after re-assembling TCP segments (3 segments in this case).

3. Certificate Status

Find "Certificate Status" packet. This message validates whether the server's X.509 digital certificate is revoked or not, it is ascertained by contacting a designated OSCP (Online Certificate Status Protocol) server. The OSCP response, which is dated and signed, contains the certificate status. The client can ask the server to send the "certificate status" message which contains the OSCP response. This approach is known as OSCP Stapling. The process saves bandwidth on constrained networks as it prevents OSCP servers from getting overwhelmed with too many client requests.

No.	Time	Source	Destination	Protocol	Length	Info
17	10.988772	202.9.95.188	10.10.5.50	TCP	1474	443 → 39110 [ACK] Seq
18	10.989001	10.10.5.50	202.9.95.188	TCP	54	39110 → 443 [ACK] Seq
19	10.989024	10.10.5.50	202.9.95.188	TCP	54	39110 → 443 [ACK] Seq
20	10.993463	202.9.95.188	10.10.5.50	TLSv1.2	1474	Certificate [TCP segr
21	10.994213	10.10.5.50	202.9.95.188	TCP	54	39110 → 443 [ACK] Seq
22	10.995056	202.9.95.188	10.10.5.50	TLSv1.2	1474	Certificate Status [
23	10.995265	10.10.5.50	202.9.95.188	TCP	54	39110 → 443 [ACK] Seq
24	10.996899	202.9.95.188	10.10.5.50	TLSv1.2	117	Server Key Exchange,
25	11.024795	10.10.5.50	202.9.95.188	TCP	54	39110 → 443 [ACK] Seq

Frame 22: 1474 bytes on wire (11792 bits), 1474 bytes captured (11792 bits) on interface -, id 0

Ethernet II, Src: 0c:97:81:3c:f2:00 (0c:97:81:3c:f2:00), Dst: 46:b2:4c:85:c3:cd (46:b2:4c:85:c3:cd)

Internet Protocol Version 4, Src: 202.9.95.188, Dst: 10.10.5.50

Transmission Control Protocol, Src Port: 443, Dst Port: 39110, Seq: 4261, Ack: 361, Len: 1420

[2 Reassembled TCP Segments (1771 bytes): #20(635), #22(1136)]

Transport Layer Security

- ▼ TLSv1.2 Record Layer: Handshake Protocol: Certificate Status
 - Content Type: Handshake (22)
 - Version: TLS 1.2 (0x0303)
 - Length: 1766
 - ▼ Handshake Protocol: Certificate Status
 - Handshake Type: Certificate Status (22)
 - Length: 1762
 - Certificate Status Type: OSCP (1)
 - OCSP Response Length: 1758
 - ▼ OCSP Response
 - responseStatus: successful (0)
 - responseBytes

Figure 8: Certificate Status

4. Server Key Exchange

Find packet "Server Key Exchange, Server Hello Done". The "Server Key Exchange" message is optional and sent when the public key present in the server's certificate is not suitable for key exchange or if the cipher suite places a restriction requiring a temporary key. This key is used by the client to encrypt Client Key Exchange later in the process.

No.	Time	Source	Destination	Protocol	Length	Info
17	10.988772	202.9.95.188	10.10.5.50	TCP	1474	443 → 39110 [ACK] Seq=1421 Ack=361 Win=
18	10.989001	10.10.5.50	202.9.95.188	TCP	54	39110 → 443 [ACK] Seq=361 Ack=1421 Win=
19	10.989024	10.10.5.50	202.9.95.188	TCP	54	39110 → 443 [ACK] Seq=361 Ack=2841 Win=
20	10.993463	202.9.95.188	10.10.5.50	TLSv1.2	1474	Certificate [TCP segment of a reasemb
21	10.994213	10.10.5.50	202.9.95.188	TCP	54	39110 → 443 [ACK] Seq=361 Ack=4261 Win=
22	10.995056	202.9.95.188	10.10.5.50	TLSv1.2	1474	Certificate Status [TCP segment of a r
23	10.995265	10.10.5.50	202.9.95.188	TCP	54	39110 → 443 [ACK] Seq=361 Ack=5681 Win=
24	10.996899	202.9.95.188	10.10.5.50	TLSv1.2	117	Server Key Exchange, Server Hello Done
25	11.024795	10.10.5.50	202.9.95.188	TCP	54	39110 → 443 [ACK] Seq=361 Ack=5744 Win=

Frame 24: 117 bytes on wire (936 bits), 117 bytes captured (936 bits) on interface -, id 0
Ethernet II, Src: 0c:97:81:3c:f2:00 (0c:97:81:3c:f2:00), Dst: 46:b2:4c:85:c3:cd (46:b2:4c:85:c3:cd)
Internet Protocol Version 4, Src: 202.9.95.188, Dst: 10.10.5.50
Transmission Control Protocol, Src Port: 443, Dst Port: 39110, Seq: 5681, Ack: 361, Len: 63
[2 Reassembled TCP Segments (338 bytes): #22(284), #24(54)]
Transport Layer Security
TLSv1.2 Record Layer: Handshake Protocol: Server Key Exchange
Content Type: Handshake (22)
Version: TLS 1.2 (0x0303)
Length: 333
Handshake Protocol: Server Key Exchange
Handshake Type: Server Key Exchange (12)
Length: 329
EC Diffie-Hellman Server Params
Curve Type: named_curve (0x03)
Named Curve: secp256r1 (0x0017)
Pubkey Length: 65
Pubkey: 043dbb4e0636ab61d58831889d90d72b23e948c246e6f1ce...
Signature Algorithm: rsa_pkcs1_sha512 (0x0601)
Signature Hash Algorithm Hash: SHA512 (6)
Signature Hash Algorithm Signature: RSA (1)
Signature Length: 256
Signature: 29624fe0e78f37f4503d1d3f1bfb67c9c05a53e70e13c683...
Transport Layer Security

Figure 9: Server Key Exchange

5. Server Hello Done

This is a zero byte message that tells the client that it's done with the "Hello" process and indicate that the server won't be asking the client for a certificate.

23	10.995265	10.10.5.50	202.9.95.188	TCP	54	39110 → 443 [ACK] Seq=361 Ack=5681 Win=
24	10.996899	202.9.95.188	10.10.5.50	TLSv1.2	117	Server Key Exchange, Server Hello Done
25	11.024795	10.10.5.50	202.9.95.188	TCP	54	39110 → 443 [ACK] Seq=361 Ack=5744 Win=

Frame 24: 117 bytes on wire (936 bits), 117 bytes captured (936 bits) on interface -, id 0
Ethernet II, Src: 0c:97:81:3c:f2:00 (0c:97:81:3c:f2:00), Dst: 46:b2:4c:85:c3:cd (46:b2:4c:85:c3:cd)
Internet Protocol Version 4, Src: 202.9.95.188, Dst: 10.10.5.50
Transmission Control Protocol, Src Port: 443, Dst Port: 39110, Seq: 5681, Ack: 361, Len: 63
[2 Reassembled TCP Segments (338 bytes): #22(284), #24(54)]
Transport Layer Security
Transport Layer Security
0000 46 b2 4c 85 c3 cd 0c 97 81 3c f2 00 08 00 45 00 F.L.....<...E.
0010 00 67 0f 26 00 00 3d 06 35 6a ca 09 5f bc 0a 0a g&...= 5j.....
0020 05 32 01 bb 98 c6 00 4e 36 32 7d 96 43 d1 50 18 .2.....N 62}.C.P.
0030 ff ff a7 fb 00 00 d8 42 9e 52 fc ed e6 f6 88 e1B .R.....
0040 d3 92 4f c8 4c 0e ad f7 a6 19 81 47 2f c6 c2 c9 .O.L.....G/...
0050 da 5b c3 2c a2 ed 7f 4e dc e5 44 4e ac 0b 7b fd .[,...N .DN.({
0060 3d 03 f9 00 63 29 ec 87 91 15 e4 9a 16 03 03 00 =...c)... ..
0070 04 0e 00 00 00

Figure 10: Server Hello Done

The browser will verify the Certificate Authority's signature on the server's certificate, and if all goes well it will continue to the next step, otherwise the browser will complain about the validity of the certificate and let the user decide if they still want to continue.

• Client response to server

Find "Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message" packet.

1. Client Key Exchange

The "Client Key Exchange" message contains: (a) the protocol version of the client which the server verifies if it matches with the original client hello message, (b) pre-master secret – a random number generated by the client and encrypted with the server's public key.

No.	Time	Source	Destination	Protocol	Length	Info
26	11.029931	10.10.5.50	202.9.95.188	TLSv1.2	180	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
27	11.037641	202.9.95.188	10.10.5.50	TCP	54	443 → 39110 [ACK] Seq=5744 Ack=487 Win=65535 Len=0
28	11.054761	202.9.95.188	10.10.5.50	TLSv1.2	312	New Session Ticket, Change Cipher Spec, Encrypted Handshake Message
29	11.059286	10.10.5.50	202.9.95.188	TCP	54	39110 → 443 [ACK] Seq=487 Ack=6002 Win=63900 Len=0

Frame 26: 180 bytes on wire (1440 bits), 180 bytes captured (1440 bits) on interface 0, id 0
 Ethernet II, Src: 46:b2:4c:85:c3:cd (46:b2:4c:85:c3:cd), Dst: 0c:97:81:3c:f2:00 (0c:97:81:3c:f2:00)
 Internet Protocol Version 4, Src: 10.10.5.50, Dst: 202.9.95.188
 Transmission Control Protocol, Src Port: 39110, Dst Port: 443, Seq: 361, Ack: 5744, Len: 126
 Transport Layer Security
 TLSv1.2 Record Layer: Handshake Protocol: Client Key Exchange
 Content Type: Handshake (22)
 Version: TLS 1.2 (0x0303)
 Length: 70
 Handshake Protocol: Client Key Exchange
 Handshake Type: Client Key Exchange (16)
 Length: 66
 EC Diffie-Hellman Client Params
 Pubkey Length: 65
 Pubkey: 041623c6d221333e91b7e8614e5d96ac13b91bb64977b4e5...
 TLSv1.2 Record Layer: Change Cipher Spec Protocol: Change Cipher Spec
 Content Type: Change Cipher Spec (20)
 Version: TLS 1.2 (0x0303)
 Length: 1
 Change Cipher Spec Message
 TLSv1.2 Record Layer: Handshake Protocol: Encrypted Handshake Message
 Content Type: Handshake (22)
 Version: TLS 1.2 (0x0303)
 Length: 40
 Handshake Protocol: Encrypted Handshake Message

Figure 11: Client Response

If the server can decrypt the message using the private key and can create the master secret locally, then the client is assured that the server has authenticated itself.

2. Change Cipher Spec

This message notifies the server that all the future messages will be encrypted using the algorithm and keys that were just negotiated.

3. Finished (Encrypted Handshake)

The Finished message is complicated as it is a hash of all the messages exchanged previously along with a label ("client finished"). This message indicates that the TLS negotiation is completed for the client.

• Master secret

If we've done everything correctly, both sides (and only those sides) now know the pre-master secret. There's a slight trust issue here from Monash's perspective: the pre-master secret just has bits that were generated by the client, they don't take anything into account from the server or anything we said earlier. We'll fix that by computing the "master secret." This is done by calculating:

$$\text{master_secret} = \text{PRF}(\text{pre_master_secret}, \text{"mastersecret"}, \text{ClientHello.random} + \text{ServerHello.random})[0..47];$$

The two random values `ClientHello.random` and `ServerHello.random`, sometimes called "nonces", are randomly generated and sent during the Hello of each parties (Remember?). This is to bound the soon-to-be master key to this session. PRF stands for Pseudo-random function, basically some concrete construction that emulates a random oracle: given an input will produce an output computationally indistinguishable from a truly random sequence.

• Server response to Client

The server informs the client that the messages will be encrypted with the existing algorithms and keys. The record layer now changes its state to use the symmetric key encryption.

No.	Time	Source	Destination	Protocol	Length	Info
37	22.282840	202.9.95.188	10.10.5.50	TLSv1.2	312	New Session Ticket, Change Cipher Spec, Encrypted Handshake Message
38	22.286000	10.10.5.50	202.9.95.188	TCP	54	36986 → 443 [ACK] Seq=487 Ack=6002 Win=63900 Len=0
39	22.288508	10.10.5.50	202.9.95.188	TLSv1.2	319	Application Data
40	22.293866	202.9.95.188	10.10.5.50	TCP	54	443 → 36986 [ACK] Seq=6002 Ack=752 Win=65535 Len=0

Frame 37: 312 bytes on wire (2496 bits), 312 bytes captured (2496 bits) on interface 0, id 0
 Ethernet II, Src: 0c:97:81:3c:f2:00 (0c:97:81:3c:f2:00), Dst: 8a:91:6f:a6:07:1e (8a:91:6f:a6:07:1e)
 Internet Protocol Version 4, Src: 202.9.95.188, Dst: 10.10.5.50
 Transmission Control Protocol, Src Port: 443, Dst Port: 36986, Seq: 5744, Ack: 487, Len: 258
 Transport Layer Security
 TLSv1.2 Record Layer: Handshake Protocol: New Session Ticket
 TLSv1.2 Record Layer: Change Cipher Spec Protocol: Change Cipher Spec
 TLSv1.2 Record Layer: Handshake Protocol: Encrypted Handshake Message

Figure 12: Server Response

• Application Data Flow

The master secret will be used for generating encryption keys (AES_GCM as agreed by both parties), MAC secrets, and IVs. The key generation algorithms are not the scope of this lab, refer to the RFC 5246 (<https://datatracker.ietf.org/doc/html/rfc5246/>) for more details.

Once the entire TLS Handshake is successfully completed and the peers validated, the applications on the peers can begin communicating with each other.

No.	Time	Source	Destination	Protocol	Length	Info
43	22.327263	202.9.95.188	10.10.5.50	TLSv1.2	333	Application Data
44	22.328839	10.10.5.50	202.9.95.188	TCP	54	36986 → 443 [ACK] Seq=752 Ack=7701 Win=63900 Len=0
45	22.333356	202.9.95.188	10.10.5.50	TCP	1474	443 → 36986 [ACK] Seq=7701 Ack=752 Win=65535 Len=1420 [TCP segment of a reassembled PDU]
46	22.333486	10.10.5.50	202.9.95.188	TCP	54	36986 → 443 [ACK] Seq=752 Ack=9121 Win=63900 Len=0
47	22.334348	202.9.95.188	10.10.5.50	TCP	94	443 → 36986 [PSH, ACK] Seq=9121 Ack=752 Win=65535 Len=40 [TCP segment of a reassembled PDU]
48	22.334564	10.10.5.50	202.9.95.188	TCP	54	36986 → 443 [ACK] Seq=752 Ack=9161 Win=63900 Len=0
49	22.334819	202.9.95.188	10.10.5.50	TLSv1.2	1474	Application Data [TCP segment of a reassembled PDU]
50	22.334913	10.10.5.50	202.9.95.188	TCP	54	36986 → 443 [ACK] Seq=752 Ack=10581 Win=63900 Len=0

Figure 13: Encrypted Application Data

• TLS 1.0/1.1

There are currently four versions of the TLS protocol in use today: TLS 1.0, 1.1, 1.2 and 1.3.

In March 2021 IETF officially deprecated TLS 1.0 and TLS 1.1.

TLS 1.0 was released in 1999, making it a nearly two-decade-old protocol. It has been known to be vulnerable to attacks—such as BEAST and POODLE, in addition to supporting weak cryptography, which doesn't keep modern-day connections sufficiently secure.

TLS 1.1 doesn't have any known protocol vulnerabilities, though it does share support for bad cryptography like TLS1.0.

According to Shodan search engine thousands of website around the world still use TLS1.0/1.1.

Monash does not support TLS1.0/1.1, let's enable TLS1.0 in openssl and try connecting with Monash.

Open console in **Internal-Client**, and copy the following in the beginning of the file `/etc/ssl/openssl.cnf`:

```
openssl_conf = default_conf
[default_conf]
ssl_conf = ssl_sect
[ssl_sect]
system_default = ssl_default_sect
[ssl_default_sect]
MinProtocol = TLSv1
CipherString = DEFAULT:@SECLEVEL=1
```

The above configuration will set the minimum TLS protocol to TLS1.0.

Now browse to Monash using TLS1.0 protocol, we can use `s_client` from openssl to do this:

```
openssl s_client -connect monash.edu:443 -tls1
```

Notice that Monash did not perform handshake with us. Let's try a website which supports TLS1.0, before executing the below command, start Wireshark on the link between **Internal-Client** and **Switch3**:

```
openssl s_client -connect www.sutherlandshire.nsw.gov.au:443 -tls1
```

Analyse the handshake traffic in Wireshark and compare it with TLS1.2 (especially compare the cipher suites).

Acknowledgment

Parts of this lab is taken from: <http://www.moserware.com/2009/06/first-few-milliseconds-of-https.html>

1.2 Additional Review Questions

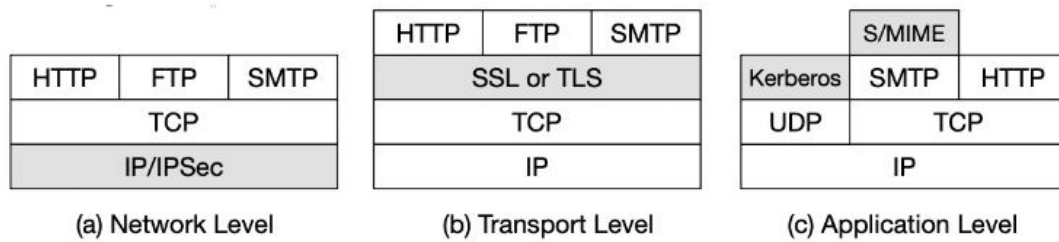


Figure 14: Relative Location of Security Facilities in the TCP/IP Protocol Stack

- What are the advantages of each of the three approaches shown in above figure?
- What services are provided by the SSL/TLS record protocol?
- What steps are involved in the SSL/TLS record protocol transmission?
- What is the purpose of HTTPS?