SSL/TLS: Handshake

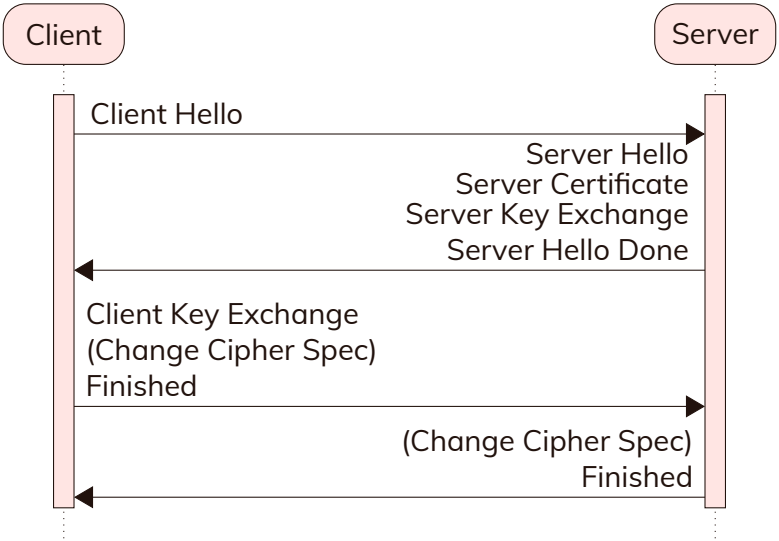
Michael Purcell

01 June 2021

Australian National University

Software Innovation Institute

TLS 1.2 Handshake



TLS Handshake Message Format

Offset	Offset $+0$	Offset $+1$	Offset $+2$	Offset $+3$
0x00	Content Type			
	0x16			
0x01	Legacy Version		Length	
	(major)	(minor)		
0x05	Message Type	Handshake Message Data Length		
0x09	Handshake Message Data			
⋮	Handshake Message Data (Continued)			

Version Numbers

Version numbers for SSL/TLS are a mess. Due to some convoluted historical reasoning we have:

Version Name	Version Number
SSL 2.0	0x0200
SSL 3.0	0x0300
TLS 1.0	0x0301
TLS 1.1	0x0302
TLS 1.2	0x0303
TLS 1.3	0x0304

Remark: TLS 1.3 does not use the ”Legacy Version” field in the TLS Handshake message format to specify the version of the protocol which the client is requesting. Rather, this is done via the ”Supported Versions” extension.

Client Hello Message

The SSL/TLS handshake begins with a ”Client Hello” message in which the client sends the following information to the server:

- The highest protocol version number that the client supports
- A 32-byte random number
- A session ID
- A list of supported cipher suites. Each cipher suite defines three cryptographic algorithms:
  - A key negotiation algorithm
  - An encryption algorithms
  - A MAC (message authentication code) algorithm
- Supported compression methods
- A list of protocol extensions that the client supports

Server Hello Message

The server responds with a ”Server Hello” message in which the server sends the following information to the client:

- The selected protocol version number
- A 32-byte random number
- A Session ID
- The selected cipher suite
- The selected compression method
- A list of the requested protocol extensions that the server will provide

Server Certificate Message

The server then sends a ”Server Certificate” message conveys the server’s certificate chain to the client.

This message includes:

- The server’s `X.509v3` certificate
- Any intermediate certificates that the client will need to authenticate the server’s identity

Server Key Exchange Message

The server then sends the ”Server Key Exchange” message which conveys cryptographic information to allow the client to communicate the premaster secret.

Some key exchange methods (e.g. `RSA`, `DH_DSS`, `DH_RSA`) do not require the server to send any cryptographic information to the client. In such cases, this message is omitted.

If ephemeral Diffie-Hellman is the selected key exchange method, (e.g. `DHE_DSS`, `DHE_RSA`) then this message includes:

- The prime modulus used for the Diffie-Hellman operation
- The generator used for the Diffie-Hellman operation
- The server’s Diffie-Hellman public value

Server Hello Done Message

The server concludes this portion of the TLS handshake by sending the ”Server Hello Done” message which tells the client to proceed to the next phase of the protocol.

This message is needed because there are several variations of the protocol in which some of the server’s messages can be omitted.

This message includes no message data. The value for the ”Message Type” field identifies the message as a ”Server Hello Done” message and no additional content is required.

Server Hello Done Details

Offset	Offset +0	Offset +1	Offset +2	Offset +3
0x00	Content Type			
	0x16			
0x01	Legacy Version		Length	
	(major)	(minor)	0x00	0x04
0x05	Message Type	Handshake Message Data Length		
	0x0e	0x00	0x00	0x00

Client Key Exchange Message

The client responds with the ”Client Key Exchange” message which conveys cryptographic information to allow the server to compute the premaster secret.

Some key exchange methods require the client to compute the premaster secret and send the encrypted result to the server. In such cases, this message includes:

- A 48-byte premaster secret encrypted using the server’s public key

If Diffie-Hellman is the selected key exchange method, then this message includes:

- The client’s Diffie-Hellman public value

Change Cipher Spec Message

Both the client and server send a ”Change Cipher Spec” (CCS) message immediately before sending their respective ”Finished” messages.

These messages indicate to the receiving parties that subsequent records will be protected under the newly negotiated Cipher Spec and keys.

The TLS Change Cipher Spec protocol from the TLS Handshake protocol. This ensures that the Change Cipher Spec message interrupts the stream of TLS Handshake message and ensures that the subsequent ”Finished” messages will be properly encrypted.

CCS Message Format

Offset	Offset $+0$	Offset $+1$	Offset $+2$	Offset $+3$
0x00	Content Type			
	0x14			
0x01	Legacy Version		Length	
	(major)	(minor)	0x00	0x01
0x05	Protocol Type			
	0x01			

Finished Messages

Both parties conclude the TLS Handshake by sending ”Finished” messages which allow them to verify that they can successfully decrypt and authenticate records sent by the other party.

These messages are encrypted and authenticated using the negotiated Cipher Spec.

These messages include:

- A `verify_data` field which is computed using all previous TLS Handshake messages
- A MAC of the record header and message data
- Any padding necessary to facilitate encryption by a block cipher (e.g. AES)

References

- The website ”The Illustrated TLS Connection” by Michael Driscoll et al (https://tls.ulfheim.net)
- The book ”SSL/TLS Under Lock and Key: A Guide to Understanding SSL/TLS” Cryptography by Paul Baka and Jeremy Schatten
- The blog ”Command Line Fanatic” by Jeremy Davies
  - ”The TLS Handshake at a High Level”
  - ”A walk-through of an SSL handshake”
  - ”SSL Key Exchange example”
  - ”SSL Certificate Exchange”
  - ”SSL OCSP Exchange”
- RFC 5246 - ”The Transport Layer Security (TLS) Protocol Version 1.2”
  - Section 7 - ”The TLS Handshaking Protocols”