TLS/SSL Handshake

Analysis in Wireshark

Lesley Nuttall

17 April 2019

Version 1

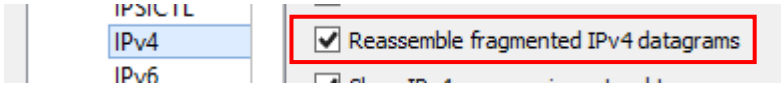
SSL Connections

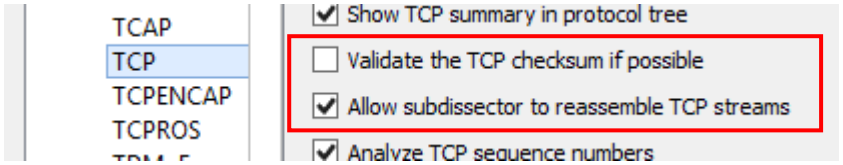
SSL/TLS is between the transport and the application layer and is protocol independent. Many IBM products make use of GSKit to establish a secure connection. The following graphic shows an example flow of API calls on a secure client using the GSKit APIs.

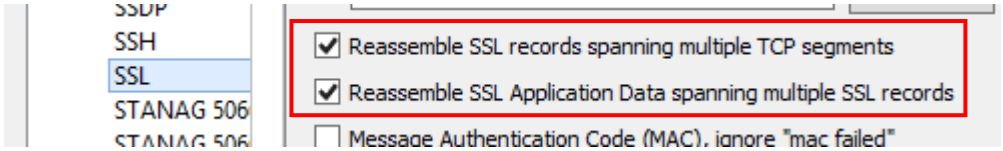
As you can see, the secure socket is opened after a connect() has been completed. In TCP conversations, you would see the SSL/TLS protocol traffic after a socket has been opened and a TCP connection successfully ESTABLISHED (as seen in netstat).

Wireshark Setup

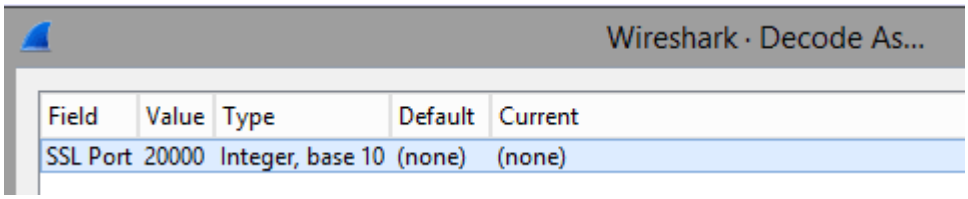
Edit -> Preferences -> Protocols



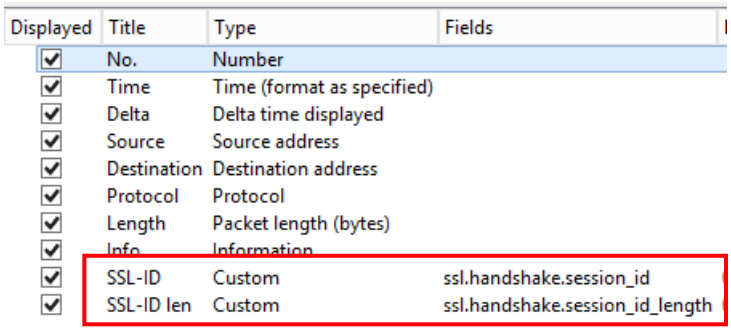




If none SSL/TLS ports are used and Wireshark does not recognise packets as SSL/TLS, you may need to define the port to be decoded as SSL. Can add this to 'Analyze -> Decode as':

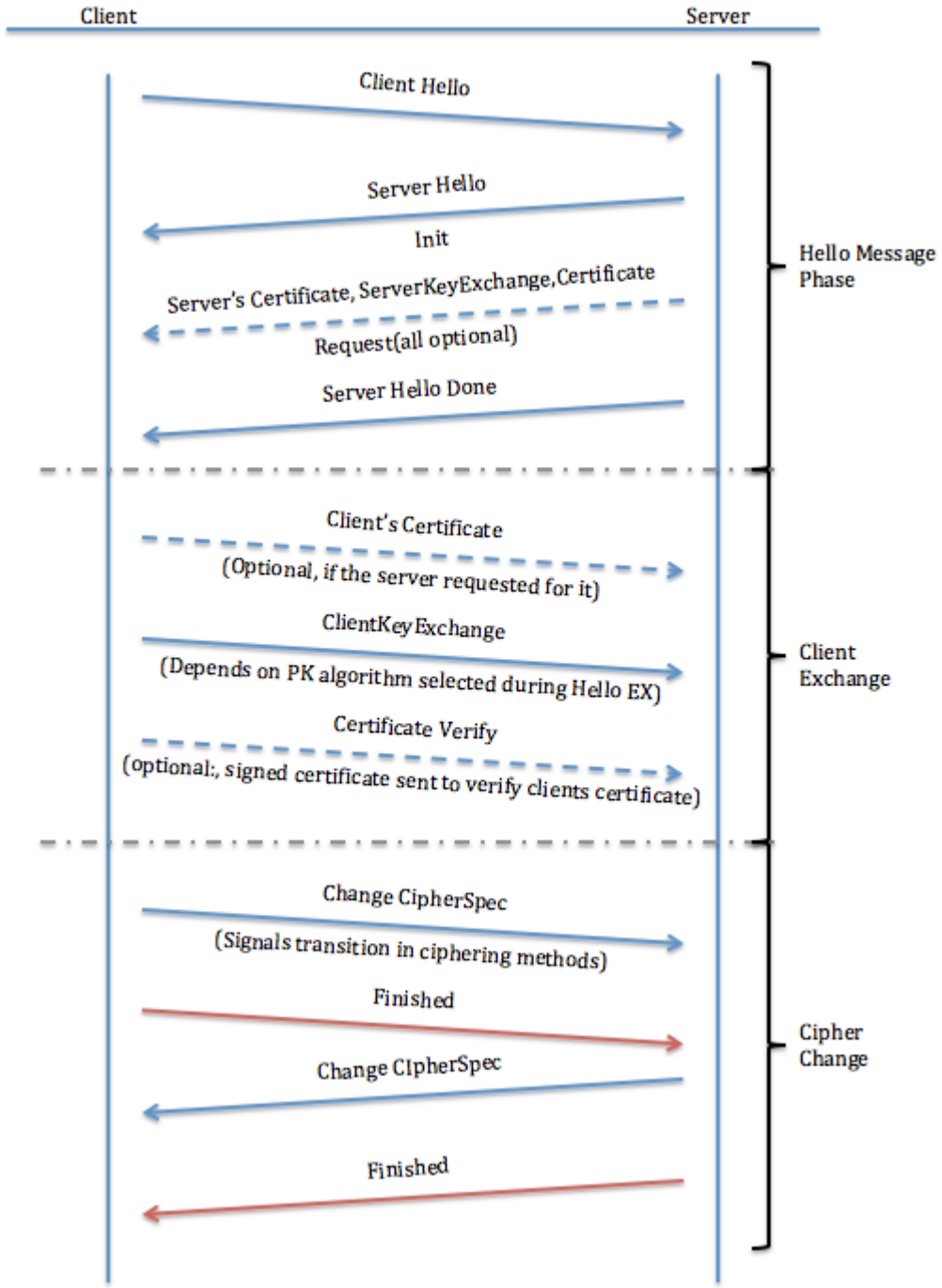


To keep track of SSL ID and SSL ID length, you can add two columns via Edit -> Preferences -> Columns, as follows:



SSL transaction flow

Source: ww.cisco.com



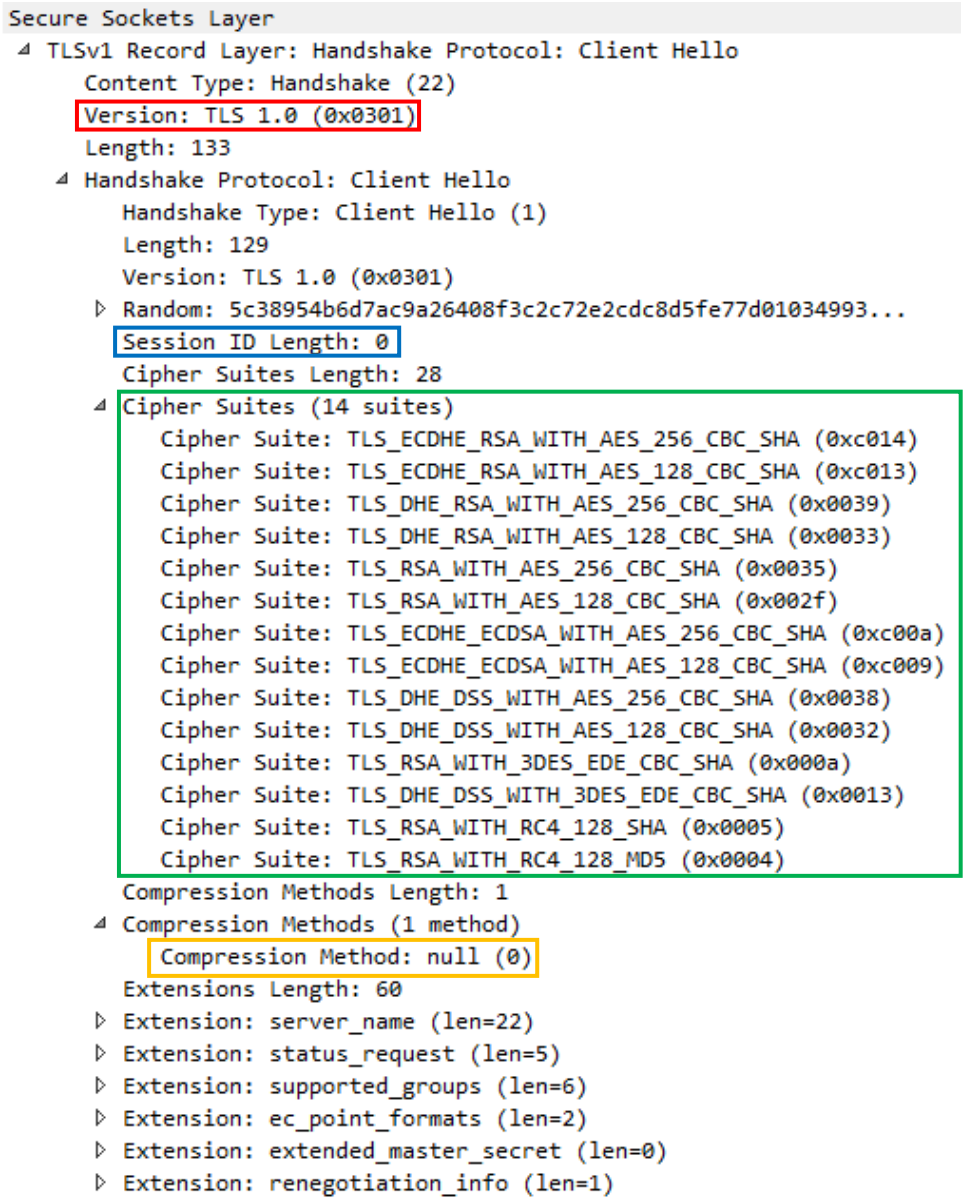
The Hello Exchange

When an SSL client and server begin to communicate, they agree on a protocol version, select cryptographic algorithms, optionally authenticate each other, and use public key encryption techniques to generate shared secrets. These processes are performed in the handshake protocol. In summary, the client sends a Client Hello message to the server, which must respond with a Server Hello message or a fatal error occurs, and the connection fails.

Interpreting a Client Hello

The Client Hello sends these attributes to the server:

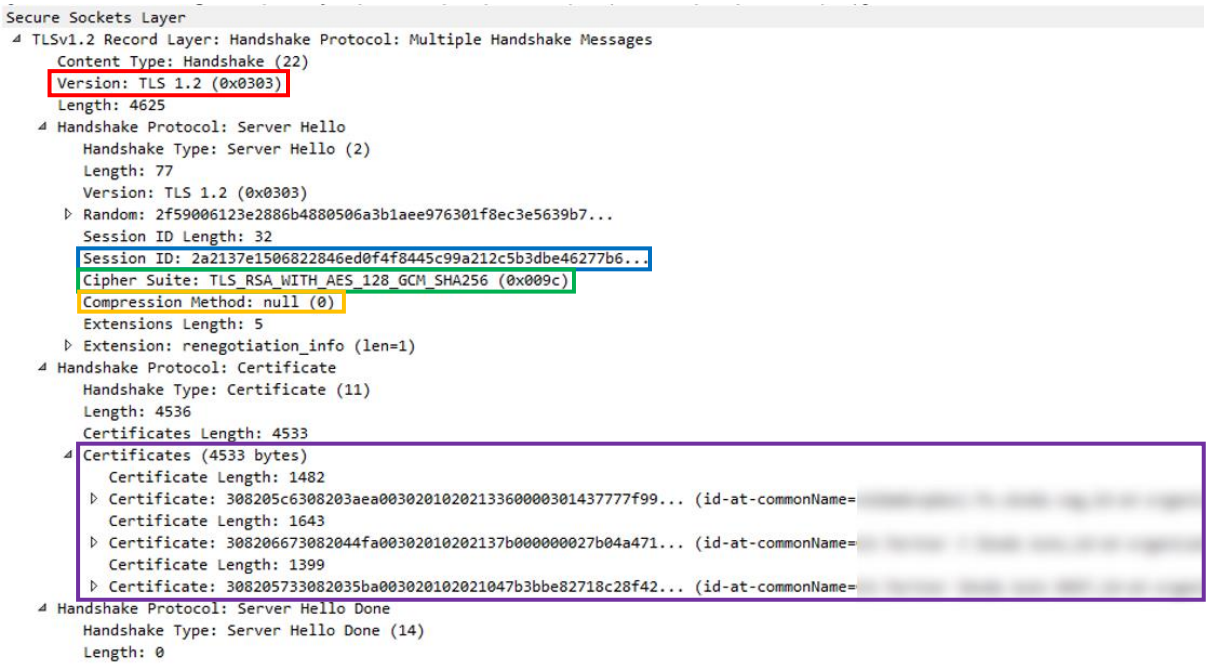
- **Protocol Version:** The version of the SSL protocol by which the client wishes to communicate during this session.
- **Session ID:** The ID of a session the client wishes to use for this connection. In the first Client Hello of the exchange, the session ID is empty (as in the example below).
- **Cipher Suite:** The combinations of cryptographic algorithms supported by the client in order of the client's preference (first choice first). Each cipher suite defines both a key exchange algorithm and a cipher spec. The server selects a cipher suite or, if no acceptable choices are presented, returns a handshake failure alert and closes the connection.
- **Compression Method:** Includes a list of compression algorithms supported by the client. If the server does not support any method sent by the client, the connection fails. The compression method can also be null.



Interpreting a Server Hello

You can view information about the certificates presented during an SSL handshake.
In the example Server Hello below, you can see the following:

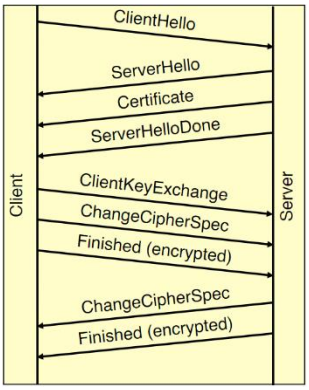
- **Protocol Version:** The chosen version of the SSL protocol that the client supports.
- **Session ID:** This is the identity of the session that corresponds to this connection. If the session ID sent by the client in the Client Hello is not empty, the server looks in the session cache for a match. If a match is found and the server is willing to establish the new connection using the specified session state, the server responds with the same value that was supplied by the client. This indicates a resumed session and dictates that the parties must proceed directly to the finished messages. Otherwise, this field contains a different value that identifies the new session. The server might return an empty **session_id** to indicate that the session will not be cached, and therefore cannot be resumed.
- **Cipher Suite:** As selected by the server from the list that was sent from the client.
- **Compression Method:** As selected by the server from the list that was sent from the client.
- **Certificate Request:** The server sends the client a list of all the certificates that are configured on it and allows the client to select which certificate it wants to use for authentication.



New SSL connection

This is a fresh connection from a client and it will not have any reference to any previous SSL session id, so will normally set the 'Session ID length' field to 0. The server will establish a new session, by performing a full SSL handshake and negotiating keys with the client. The server will assign an SSL session ID that the client will remember. The server application would also normally have an SSL session cache that it maintains, with a list of all SSL session ids.

Protocol	Length	Info
TCP	74	22454 → 4000 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=2617
TCP	74	4000 → 22454 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=1
TCP	66	22454 → 4000 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=2617138731 TSecr=16
TLSv1.2	258	Client Hello
TCP	66	4000 → 22454 [ACK] Seq=1 Ack=193 Win=30080 Len=0 TSval=160563734 TSecr=2
TLSv1.2	4704	Server Hello, Certificate, Server Hello Done
TCP	66	22454 → 4000 [ACK] Seq=193 Ack=4639 Win=38528 Len=0 TSval=2617138895 TSe
TLSv1.2	384	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
TCP	66	4000 → 22454 [ACK] Seq=4639 Ack=511 Win=31104 Len=0 TSval=160563903 TSec
TLSv1.2	72	Change Cipher Spec
TLSv1.2	111	Encrypted Handshake Message



In the first Client Hello of an exchange, the session ID is empty. You can search for these packets using the display filter "ssl.handshake.type == 1 and ssl.handshake.session_id_length == 0"

ssl.handshake.type == 1 and ssl.handshake.session_id_length == 0

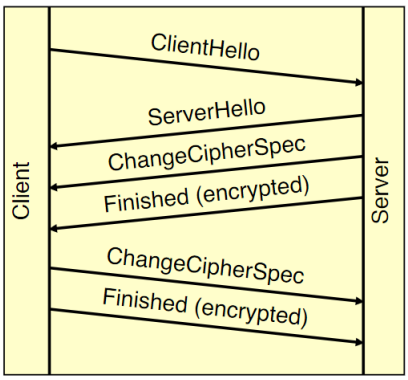
No.	Time	Delta	Source	Destination	Protocol	Length	Info
49752	7.205934	0.000666	[redacted]	[redacted]	TLSv1.2	170	Client Hello
49831	7.238793	0.032859	[redacted]	[redacted]	TLSv1.2	170	Client Hello
50129	7.284068	0.045275	[redacted]	[redacted]	TLSv1.2	170	Client Hello
50250	7.305478	0.021410	[redacted]	[redacted]	TLSv1.2	170	Client Hello
50319	7.311263	0.005785	[redacted]	[redacted]	TLSv1.2	170	Client Hello
51857	7.418130	0.106867	[redacted]	[redacted]	TLSv1	184	Client Hello
52287	7.452057	0.033927	[redacted]	[redacted]	TLSv1.2	170	Client Hello
53481	7.564377	0.112320	[redacted]	[redacted]	TLSv1.2	170	Client Hello
53612	7.575901	0.011524	[redacted]	[redacted]	TLSv1.2	170	Client Hello
53712	7.584647	0.008746	[redacted]	[redacted]	TLSv1.2	170	Client Hello
54277	7.636559	0.051912	[redacted]	[redacted]	TLSv1.2	170	Client Hello
54979	7.735826	0.099267	[redacted]	[redacted]	TLSv1.2	170	Client Hello

```
▲ TLSv1.2 Record Layer: Handshake Protocol: Client Hello
    Content Type: Handshake (22)
    Version: TLS 1.2 (0x0303)
    Length: 99
  ▲ Handshake Protocol: Client Hello
      Handshake Type: Client Hello (1)
      Length: 95
      Version: TLS 1.2 (0x0303)
    ▷ Random: 4dbbe3d4c242d0a086906f5760864e8c088860fd990d3b48...
      Session ID Length: 0
      Cipher Suites Length: 36
    ▷ Cipher Suites (18 suites)
      Compression Methods Length: 1
    ▷ Compression Methods (1 method)
      Extensions Length: 18
    ▷ Extension: supported_groups (len=8)
    ▷ Extension: ec_point_formats (len=2)
```

Resumed SSL connection

If the Client Hello presents a session id that the Server recognises, the session is resumed – meaning the session reuses previously negotiated keys. A full SSL handshake is not required. This is quicker and less 'expensive' than full key negotiation.

Protocol	Length	Info
TCP	74	21012 → 7135 [SYN] Seq=0 Win=29
TCP	74	7135 → 21012 [SYN, ACK] Seq=0 A
TCP	66	21012 → 7135 [ACK] Seq=1 Ack=1
TLSv1.2	200	Client Hello
TCP	66	7135 → 21012 [ACK] Seq=1 Ack=13
TLSv1.2	158	Server Hello
TCP	66	21012 → 7135 [ACK] Seq=135 Ack=
TLSv1.2	72	Change Cipher Spec
TCP	66	21012 → 7135 [ACK] Seq=135 Ack=
TLSv1.2	72	Change Cipher Spec
TLSv1.2	111	Encrypted Handshake Message



You can search for these packets using the display filter "ssl.resume". Wireshark expert info also tells you the session reuses previously negotiated keys.

```
▲ Secure Sockets Layer
  ▲ TLSv1.2 Record Layer: Handshake Protocol: Client Hello
      Content Type: Handshake (22)
      Version: TLS 1.2 (0x0303)
      Length: 219
    ▲ Handshake Protocol: Client Hello
        Handshake Type: Client Hello (1)
        Length: 215
        Version: TLS 1.2 (0x0303)
      ▷ Random: 5c38954d85177492431f3ee9fdcedccaef8ade9457ca9cb0...
        Session ID Length: 32
        Session ID: b6afcefa33613eed905ba7a739d75ba5b301639527190eb0...
        Cipher Suites Length: 42
      ▷ Cipher Suites (21 suites)
        Compression Methods Length: 1
      ▷ Compression Methods (1 method)
```

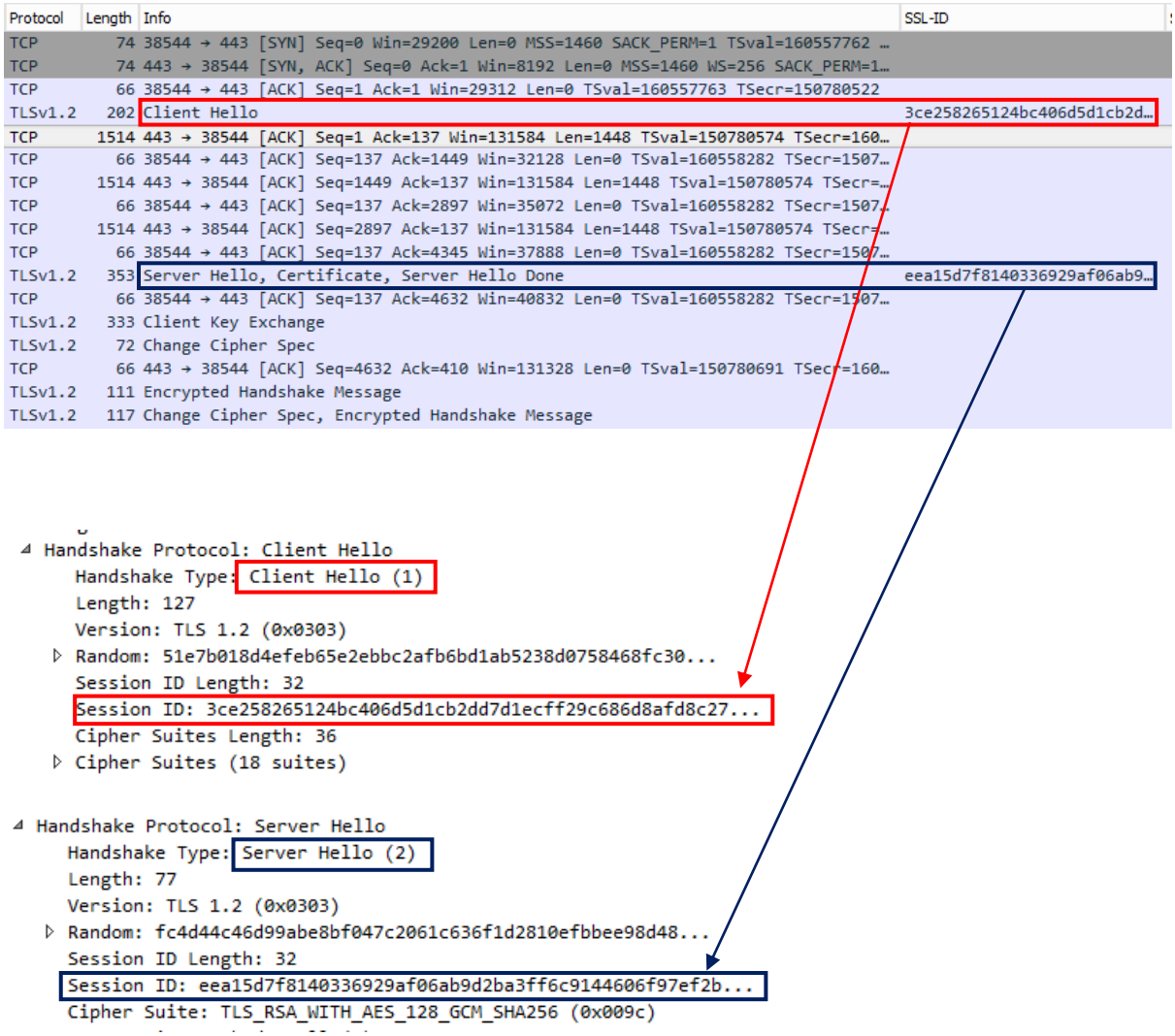
```
▲ Secure Sockets Layer
  ▲ TLSv1.2 Record Layer: Handshake Protocol: Server Hello
      Content Type: Handshake (22)
      Version: TLS 1.2 (0x0303)
      Length: 85
    ▲ Handshake Protocol: Server Hello
        Handshake Type: Server Hello (2)
        Length: 81
        Version: TLS 1.2 (0x0303)
      ▷ Random: d9b36e5ad82a37a68dcc8dcef7e7b841144b7abb9987c77d...
        Session ID Length: 32
        Session ID: b6afcefa33613eed905ba7a739d75ba5b301639527190eb0...
        Cipher Suite: TLS_RSA_WITH_AES_128_GCM_SHA256 (0x009c)
        Compression Method: null (0)
        Extensions Length: 9
      ▷ Extension: extended_master_secret (len=0)
```

```
▲ Secure Sockets Layer
  ▲ 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
      ▲ [Expert Info (Note/Sequence): This session reuses previously negotiated keys (Session resumption)]
          [This session reuses previously negotiated keys (Session resumption)]
          [Severity level: Note]
          [Group: Sequence]
```

Renegotiated SSL connection

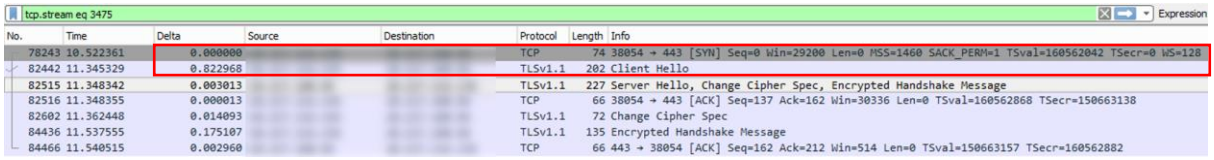
The client thinks it is re-using an SSL session, so it will present the previously used SSL session ID to the server. If the server does not recognise the session ID, a full SSL handshake is required in order to negotiate keys again. As a new session must be negotiated, the Server Hello response will contain a new SSL ID that does not match the one presented in the Client Hello. This is resource intensive (similar to a new ssl session).

The server may not recognise the SSL id for various reasons - the server ssl cache is full, the server ssl cache timeout has triggered, the ssl session id has been invalidated in the cache, in a cluster/load balanced env the connection has switched to a new server, etc.

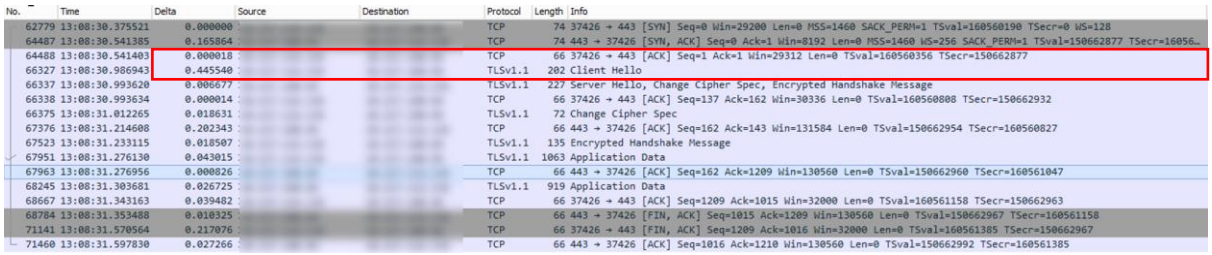


Example delay from establishing TCP connection to SSL handshake

0.82 second delay from establishing connection to sending ssl handshake



0.45 second delay from establishing connection to sending ssl handshake



This could indicate that there is a performance issue on the server acting as the SSL client. Key negotiation is resource expensive, so check to see if sessions are not being resumed where you would expect them to be (i.e. there are many unnecessary renegotiated SSL handshakes).