Client

Server

HTTPS SSL/TLS Session for SPDY

This sequence diagram covers the establishment of a SSL/TLS connection for sending Google SPDY data. The protocol flow covers:

(1) SSL/TLS initial cryptographic parameter negotiation.

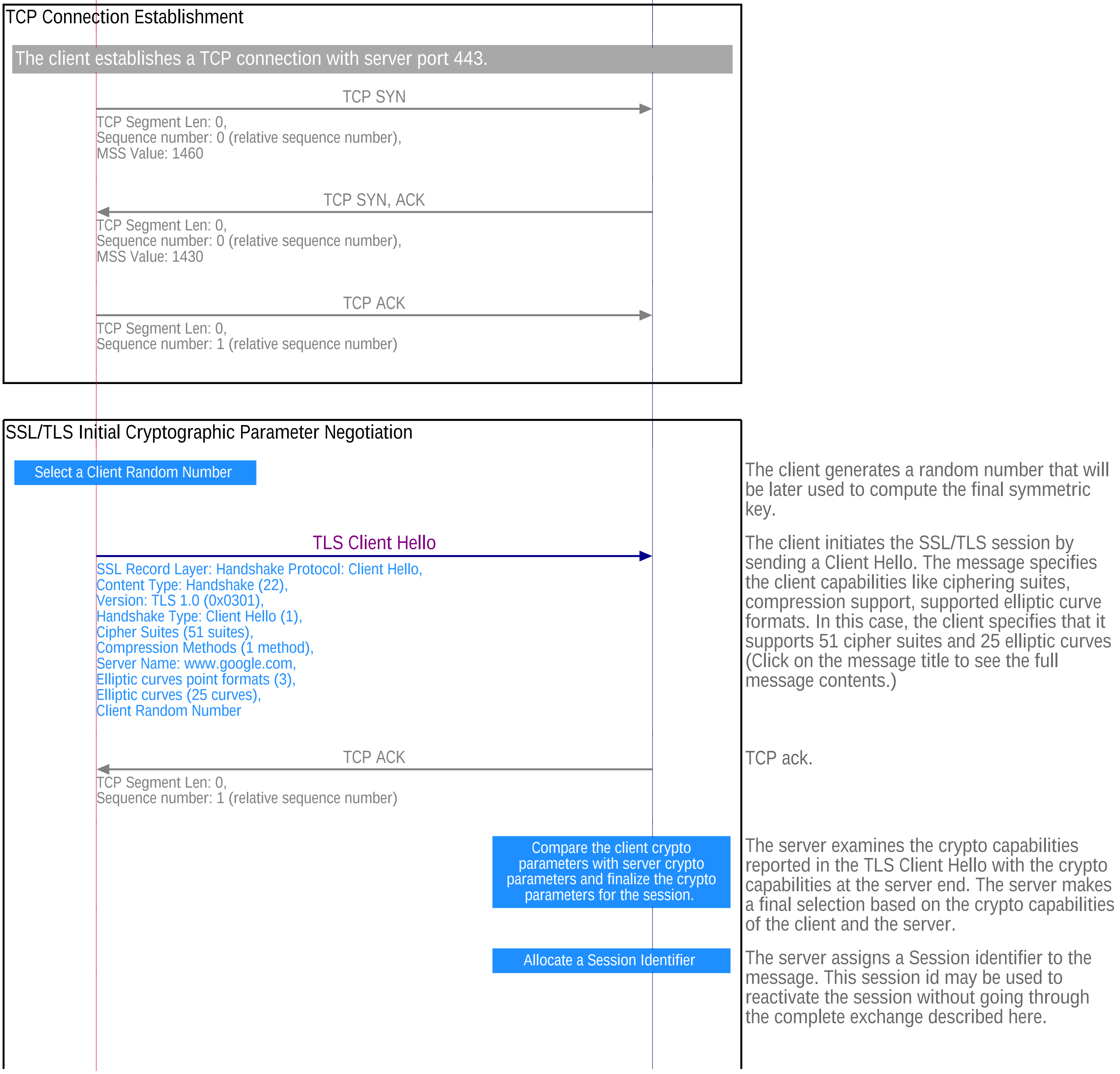
(2) Certificate exchange and encryption start with elliptic curve Diffie Hellman key exchange.

(3) Master key generation and encrypted data transfer.

(4) SSL/TLS session release.

Generated with EventStudio (http://www.eventhelix.com/eventstudio/) and VisualEther (http://www.eventhelix.com/visualether/)

Note: You can click on any message title in this flow to examine the message structure and fields.



TCP Connection Establishment

The client establishes a TCP connection with server port 443.

TCP SYN

TCP Segment Len: 0,
Sequence number: 0 (relative sequence number),
MSS Value: 1460

TCP SYN, ACK

TCP Segment Len: 0,
Sequence number: 0 (relative sequence number),
MSS Value: 1430

TCP ACK

TCP Segment Len: 0,
Sequence number: 1 (relative sequence number)

SSL/TLS Initial Cryptographic Parameter Negotiation

Select a Client Random Number

The client generates a random number that will be later used to compute the final symmetric key.

TLS Client Hello

SSL Record Layer: Handshake Protocol: Client Hello,
Content Type: Handshake (22),
Version: TLS 1.0 (0x0301),
Handshake Type: Client Hello (1),
Cipher Suites (51 suites),
Compression Methods (1 method),
Server Name: www.google.com,
Elliptic curves point formats (3),
Elliptic curves (25 curves),
Client Random Number

The client initiates the SSL/TLS session by sending a Client Hello. The message specifies the client capabilities like ciphering suites, compression support, supported elliptic curve formats. In this case, the client specifies that it supports 51 cipher suites and 25 elliptic curves (Click on the message title to see the full message contents.)

TCP ACK

TCP Segment Len: 0,
Sequence number: 1 (relative sequence number)

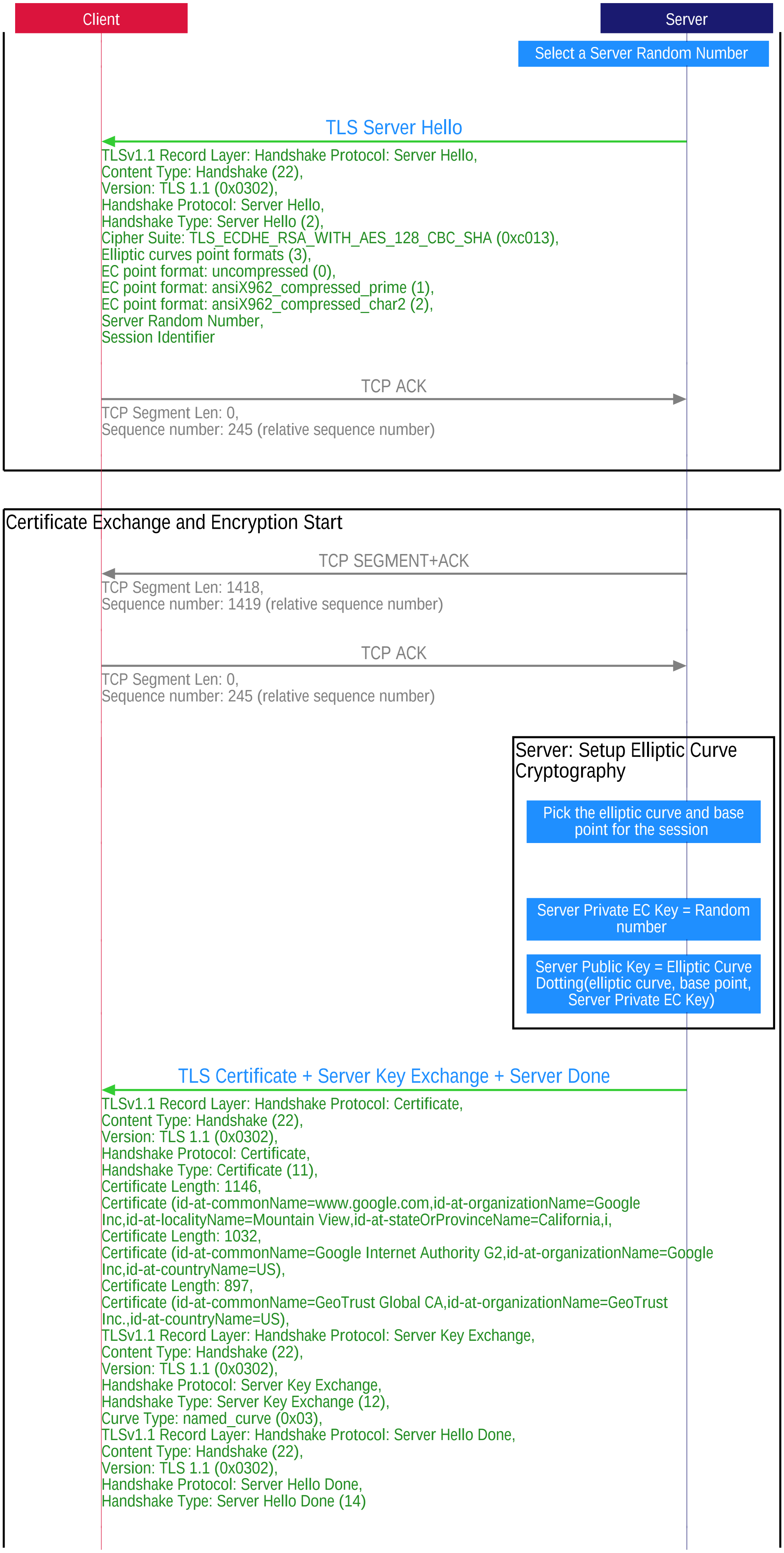
TCP ack.

Compare the client crypto parameters with server crypto parameters and finalize the crypto parameters for the session.

The server examines the crypto capabilities reported in the TLS Client Hello with the crypto capabilities at the server end. The server makes a final selection based on the crypto capabilities of the client and the server.

Allocate a Session Identifier

The server assigns a Session identifier to the message. This session id may be used to reactivate the session without going through the complete exchange described here.



Select a Server Random Number

The server generates a random number that will be later used to compute the final symmetric key.

TLS Server Hello

TLSv1.1 Record Layer: Handshake Protocol: Server Hello,
Content Type: Handshake (22),
Version: TLS 1.1 (0x0302),
Handshake Protocol: Server Hello,
Handshake Type: Server Hello (2),
Cipher Suite: TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA (0xc013),
Elliptic curves point formats (3),
EC point format: uncompressed (0),
EC point format: ansiX962_compressed_prime (1),
EC point format: ansiX962_compressed_char2 (2),
Server Random Number,
Session Identifier

The server makes a final selection based on the crypto capabilities of the client and the server. In this case, the server has selected:
- RSA for Certification
- Elliptic Curve based Diffie Hellman
- AES 128 Encryption for the data

TCP ACK

TCP Segment Len: 0,
Sequence number: 245 (relative sequence number)

Certificate Exchange and Encryption Start

TCP SEGMENT+ACK

TCP Segment Len: 1418,
Sequence number: 1419 (relative sequence number)

A segment of the "TLS Certificate + Server Key Exchange + Server Done" message. The message is split into two IP segments.

TCP ACK

TCP Segment Len: 0,
Sequence number: 245 (relative sequence number)

Server: Setup Elliptic Curve Cryptography

Pick the elliptic curve and base point for the session

Select the elliptic curve and the base point that will be used for the Diffie-Hellman key exchange. Click on the action box to learn more about elliptic curve cryptography.

Server Private EC Key = Random number

A random number is generated to be used as the server's private key.

Server Public Key = Elliptic Curve Dotting(elliptic curve, base point, Server Private EC Key)

Derive the public key that will be sent to the client.

TLS Certificate + Server Key Exchange + Server Done

TLSv1.1 Record Layer: Handshake Protocol: Certificate,
Content Type: Handshake (22),
Version: TLS 1.1 (0x0302),
Handshake Protocol: Certificate,
Handshake Type: Certificate (11),
Certificate Length: 1146,
Certificate (id-at-commonName=www.google.com,id-at-organizationName=Google Inc,id-at-localityName=Mountain View,id-at-stateOrProvinceName=California,i,
Certificate Length: 1032,
Certificate (id-at-commonName=Google Internet Authority G2,id-at-organizationName=Google Inc,id-at-countryName=US),
Certificate Length: 897,
Certificate (id-at-commonName=GeoTrust Global CA,id-at-organizationName=GeoTrust Inc.,id-at-countryName=US),
TLSv1.1 Record Layer: Handshake Protocol: Server Key Exchange,
Content Type: Handshake (22),
Version: TLS 1.1 (0x0302),
Handshake Protocol: Server Key Exchange,
Handshake Type: Server Key Exchange (12),
Curve Type: named_curve (0x03),
TLSv1.1 Record Layer: Handshake Protocol: Server Hello Done,
Content Type: Handshake (22),
Version: TLS 1.1 (0x0302),
Handshake Protocol: Server Hello Done,
Handshake Type: Server Hello Done (14)

The server sends a compound message that contains the following:

X.509 Certificates

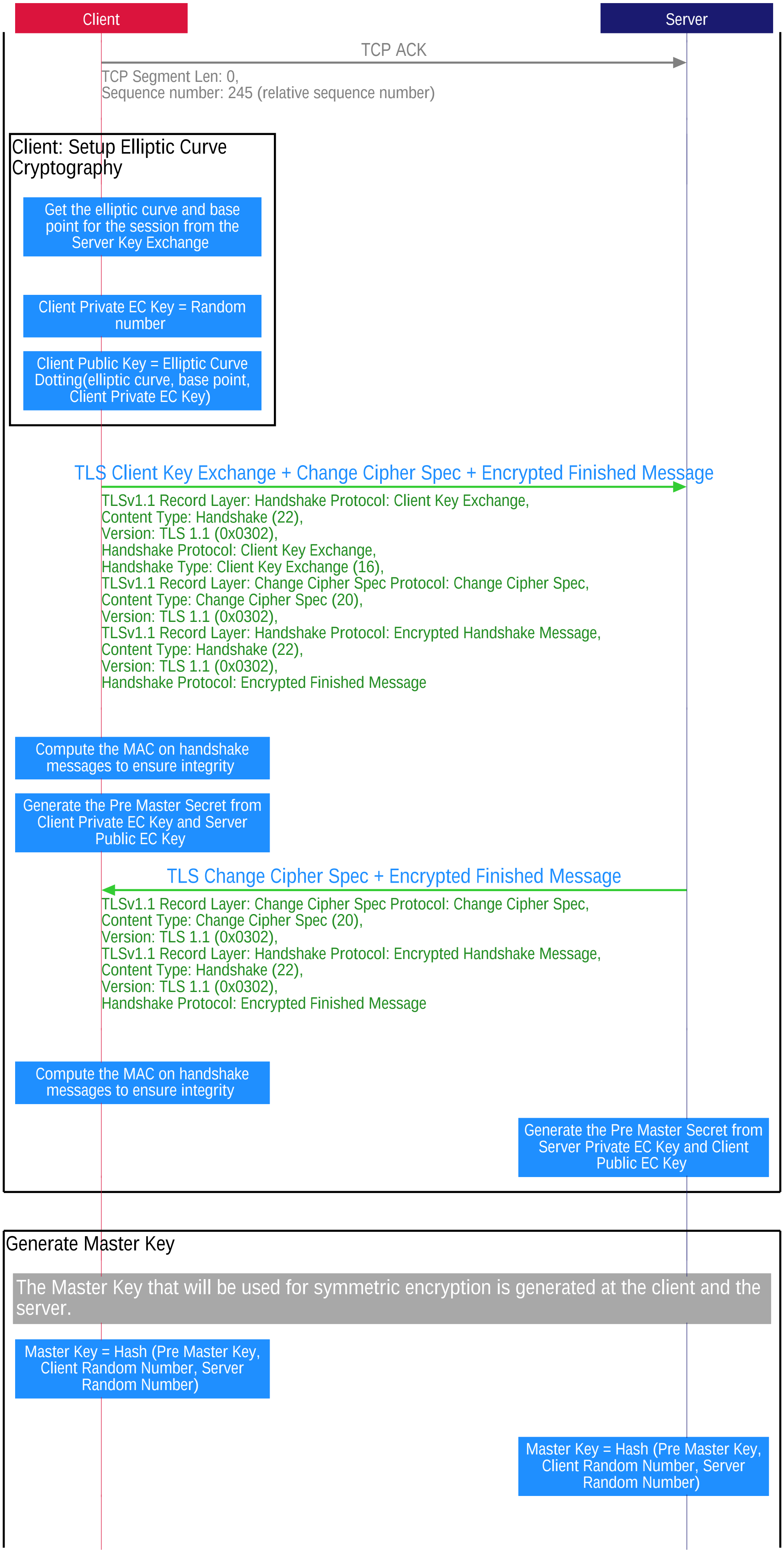
A cascade of three certificates to authenticate that the Google Server:
(1) Google server certificate (issued and signed by Google Intermediate CA)
(2) Google Intermediate CA certificate (issued and signed by GeoTrust CA)
(3) GeoTrust CA certificate. (issued and signed by Equifax Root CA)

Server Key Exchange

The Google server is using Elliptic Curve cryptography so it sends a EC Diffie-Hellman public key and signature.

Server Done

Signals that the complete cryptographic information has been sent from the server.



Client

Server

TCP ACK

TCP Segment Len: 0,
Sequence number: 245 (relative sequence number)

Client: Setup Elliptic Curve Cryptography

Get the elliptic curve and base point for the session from the Server Key Exchange

Select the elliptic curve and the base point that will be used for the Diffie-Hellman key exchange. Click on the action box to learn more about elliptic curve cryptography.

Client Private EC Key = Random number

A random number is generated to be used as the client's private key.

Client Public Key = Elliptic Curve Dotting(elliptic curve, base point, Client Private EC Key)

Derive the public key that will be sent to the server.

TLS Client Key Exchange + Change Cipher Spec + Encrypted Finished Message

TLSv1.1 Record Layer: Handshake Protocol: Client Key Exchange,
Content Type: Handshake (22),
Version: TLS 1.1 (0x0302),
Handshake Protocol: Client Key Exchange,
Handshake Type: Client Key Exchange (16),
TLSv1.1 Record Layer: Change Cipher Spec Protocol: Change Cipher Spec,
Content Type: Change Cipher Spec (20),
Version: TLS 1.1 (0x0302),
TLSv1.1 Record Layer: Handshake Protocol: Encrypted Handshake Message,
Content Type: Handshake (22),
Version: TLS 1.1 (0x0302),
Handshake Protocol: Encrypted Finished Message

Client Key Exchange
The client sends EC Diffie-Hellman public key and signature.

Change Cipher Spec
Client signals that is initiating encryption from the next record.

Encrypted Finished Message
This message contains the MAC of the handshake messages. The MAC ensures that the handshake messages that were sent in the clear have not been modified by a third party.

Compute the MAC on handshake messages to ensure integrity

The client proceeds only if the MAC integrity check passes.

Generate the Pre Master Secret from Client Private EC Key and Server Public EC Key

The shared secret is derived as a result of the Diffie-Hellman key exchange.

TLS Change Cipher Spec + Encrypted Finished Message

TLSv1.1 Record Layer: Change Cipher Spec Protocol: Change Cipher Spec,
Content Type: Change Cipher Spec (20),
Version: TLS 1.1 (0x0302),
TLSv1.1 Record Layer: Handshake Protocol: Encrypted Handshake Message,
Content Type: Handshake (22),
Version: TLS 1.1 (0x0302),
Handshake Protocol: Encrypted Finished Message

Change Cipher Spec
Server signals that is initiating encryption from the next record.

Encrypted Finished Message
This message contains the MAC of the handshake messages. The MAC ensures that the handshake messages that were sent in the clear have not been modified by a third party.

Compute the MAC on handshake messages to ensure integrity

The client proceeds only if the MAC integrity check passes.

Generate the Pre Master Secret from Server Private EC Key and Client Public EC Key

The shared secret is derived as a result of the Diffie-Hellman key exchange.

Generate Master Key

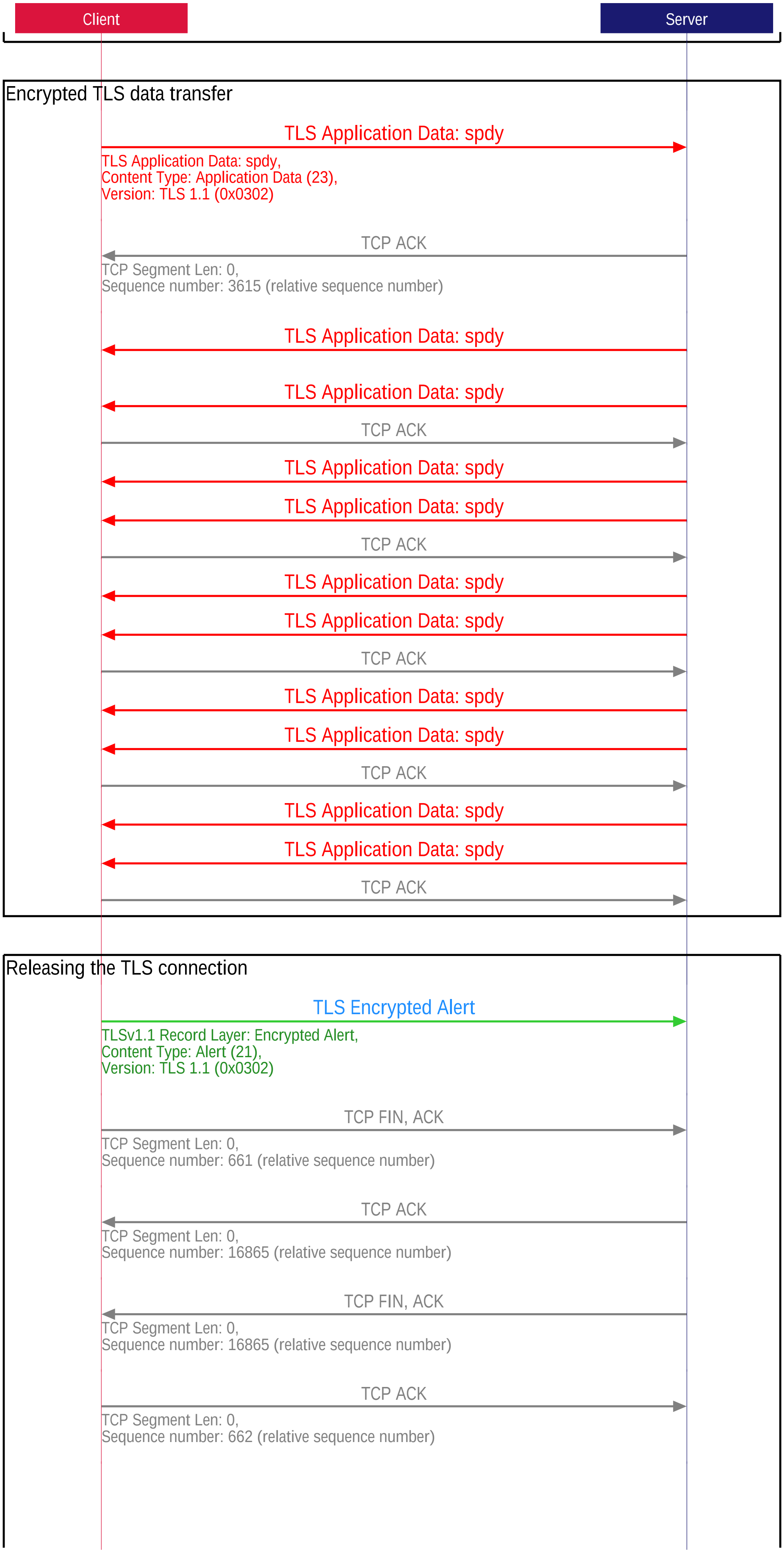
The Master Key that will be used for symmetric encryption is generated at the client and the server.

Master Key = Hash (Pre Master Key, Client Random Number, Server Random Number)

The client generates the Master Key that depends on the Pre Master Key and the client and server random numbers. This protects the session from replay attacks.

Master Key = Hash (Pre Master Key, Client Random Number, Server Random Number)

The server also generates the Master Key.



Client

Server

Encrypted TLS data transfer

TLS Application Data: spdy

TLS Application Data: spdy,
Content Type: Application Data (23),
Version: TLS 1.1 (0x0302)

The client sends a SPDY packet encrypted with the Master Key.

TCP ACK

TCP Segment Len: 0,
Sequence number: 3615 (relative sequence number)

TLS Application Data: spdy

The server sends a SPDY packet encrypted with the Master Key.

TLS Application Data: spdy

TCP ACK

TLS Application Data: spdy

TLS Application Data: spdy

TCP ACK

TLS Application Data: spdy

TLS Application Data: spdy

TCP ACK

TLS Application Data: spdy

TLS Application Data: spdy

TCP ACK

TLS Application Data: spdy

TLS Application Data: spdy

TCP ACK

Releasing the TLS connection

TLS Encrypted Alert

TLSv1.1 Record Layer: Encrypted Alert,
Content Type: Alert (21),
Version: TLS 1.1 (0x0302)

The client sends an Alert (Close) to release the TLS connection.

TCP FIN, ACK

TCP Segment Len: 0,
Sequence number: 661 (relative sequence number)

The client also initiates the release of the TCP connection with a FIN.

TCP ACK

TCP Segment Len: 0,
Sequence number: 16865 (relative sequence number)

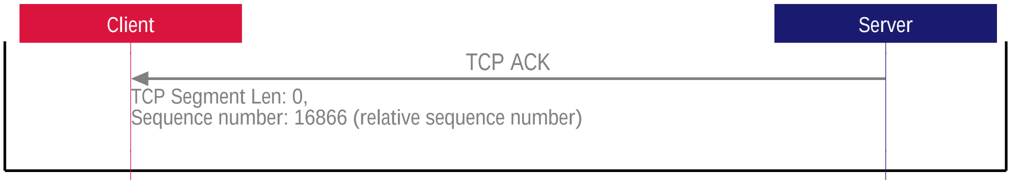
TCP FIN, ACK

TCP Segment Len: 0,
Sequence number: 16865 (relative sequence number)

The server also releases the TCP connection.

TCP ACK

TCP Segment Len: 0,
Sequence number: 662 (relative sequence number)



EXPLORE MORE

SSL Sequence Diagram http://www.eventhelix.com/realtimemantra/networking/SSL.pdf

Networking Protocol Flows http://www.eventhelix.com/realtimemantra/networking/

LTE http://www.eventhelix.com/lte/