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.

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.

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.

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.)

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

Allocate a Session Identifier

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.

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.

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)

TCP SEGMENT+ACK

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

TCP ACK

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

Certificate Exchange and Encryption Start

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

Server: Setup Elliptic Curve Cryptography

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.

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

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

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

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 (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

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.
- The client proceeds only if the MAC integrity check passes.
- 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.

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

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

The client sends additional SPDY packets, each encrypted with the Master Key.

The server also sends SPDY packets, each encrypted with the Master Key.

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

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

The server also releases the TCP connection.
TCP ACK
TCP Segment Len: 0,
Sequence number: 16866 (relative sequence number)

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

EXPLORE MORE
SSL Sequence Diagram http://www.eventhelix.com/realtimenamtra/networking/SSL.pdf
LTE http://www.eventhelix.com/lte/