SSL Introduction with Sample Transaction and Packet Exchange

Contents

Introduction

This document describes the basic concepts of Secure Sockets Layer (SSL) protocol, and provides a sample transaction and packet capture.

SSL Record Overview

The basic unit of data in SSL is a record. Each record consists of a five-byte record header, followed by data.

Record Format

- **Type**: uint8 - values listed
- **Version**: uint16
- **Length**: uint16

Type	Version		Length	
T	VH	VL	LH	LL

Record Type

There are four record types in SSL:

- **Handshake** (22, 0x16)
- **Change Cipher Spec** (20, 0x14)
- **Alert** (21, 0x15)
- **Application Data** (23, 0x17)

Record Version

The record version is a 16-bits value and is formatted in network order.

> **Note**: For SSL Version 3 (SSLv3), the version is 0x0300. For Transport Layer Security Version 1 (TLSv1), the version is 0x0301. The Cisco Adaptive Security Appliance (ASA) does not support SSL Version 2 (SSLv2), which uses version 0x0002, or any version of TLS greater than TLSv1.

Record Length

The record length is a 16-byte value and is formatted in network order.

In theory, this means that a single record can be up to 65,535 (2^16 -1) bytes in length. The TLSv1 RFC2246 states that the maximum length is 16,383 (2^14 -1) bytes. Microsoft products (Microsoft Internet Explorer and Internet Information Services) are known to exceed these limits.

Types of Records

This section describes the four types of SSL records.

Handshake Records

Handshake records contain a set of messages that are used in order to handshake. These are the messages and their values:

- **Hello Request** (0, 0x00)
- **Client Hello** (1, 0x01)
- **Server Hello** (2, 0x02)
- **Certificate** (11, 0x0B)
- **Server Key Exchange** (12, 0x0C)
- **Certificate Request** (13, 0x0D)
- **Server Hello Done** (14, 0x0E)
- **Certificate Verify** (15, 0x0F)
- **Client Key Exchange** (16, 0x10)
- **Finished** (20, 0x14)

In the simple case, handshake records are not encrypted. However, a handshake record that contains a finished message is always encrypted, as it always occurs after a Change Cipher Spec (CCS) record.

CCS Records

CCS records are used in order to indicate a change in cryptographic ciphers. Immediately after the CCS record, all data is encrypted with the new cipher. CCS records might or might not be encrypted; in a simple connection with a single handshake, the CCS record is not encrypted.

Alert Records

Alert records are used in order to indicate to the peer that a condition has occured. Some alerts are warnings, while others are fatal and cause the connection to fail. Alerts might or might not be encrypted, and might occur during a handshake or during data transfer. There are two types of alerts:

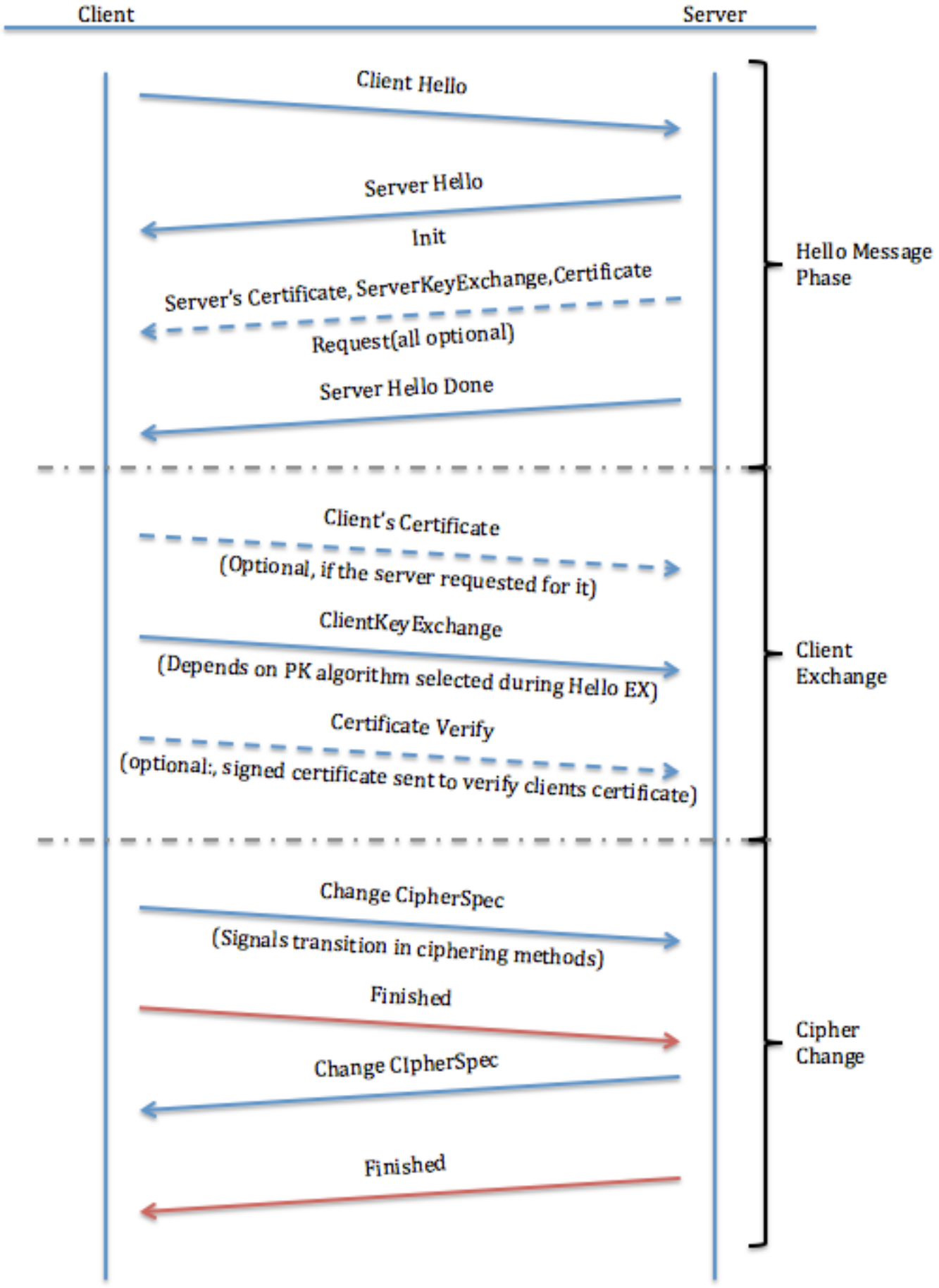
- **Closure Alerts:** The connection between the client and the server must be properly closed in order to avoid any kind of truncation attacks. A **close_notify** message is sent that indicates to the recipient that the sender will not send anymore messages on that connection.
- **Error Alerts:** When an error is detected, the detecting party sends a message to the other party. Upon transmission or receipt of a fatal alert message, both parties immediately close the connection. Some examples of error alerts are:
  - **unexpected_message** (fatal)
  - **decompression_failure**
  - **handshake_failure**

Application Data Record

These records contain the actual application data. These messages are carried by the record layer and are fragmented, compressed, and encrypted, based on the current connection state.

Sample Transaction

This section describes a sample transaction between the client and server.



The Hello Exchange

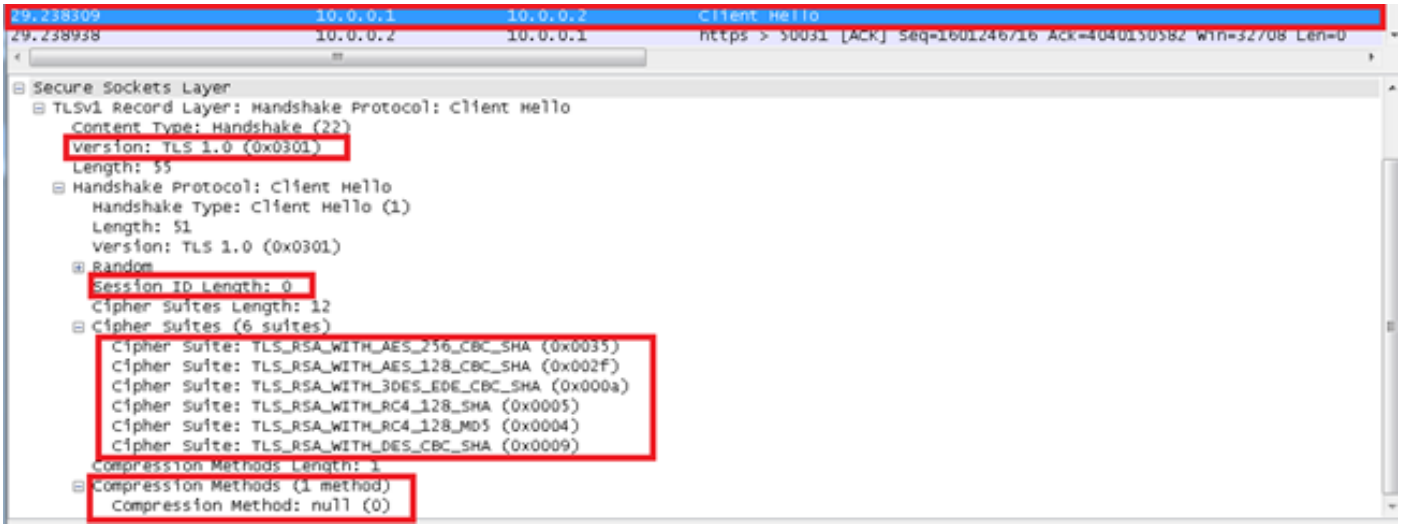
When an SSL client and server begin to communicate, they agree on a protocol version, select cryptographic algorithms, optionally authenticate each other, and use public key encryption techniques in order to generate shared secrets. These processes are performed in the handshake protocol. In summary, 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. The Client Hello and Server Hello are used to establish security enhancement capabilities between the client and server.

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 (refer to the packet capture screen shot after the note).
- **Cipher Suite:** This is passed from the client to the server in the Client Hello message. It contains 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.

Note: The server IP address in the captures is 10.0.0.2 and the client IP address is 10.0.0.1.



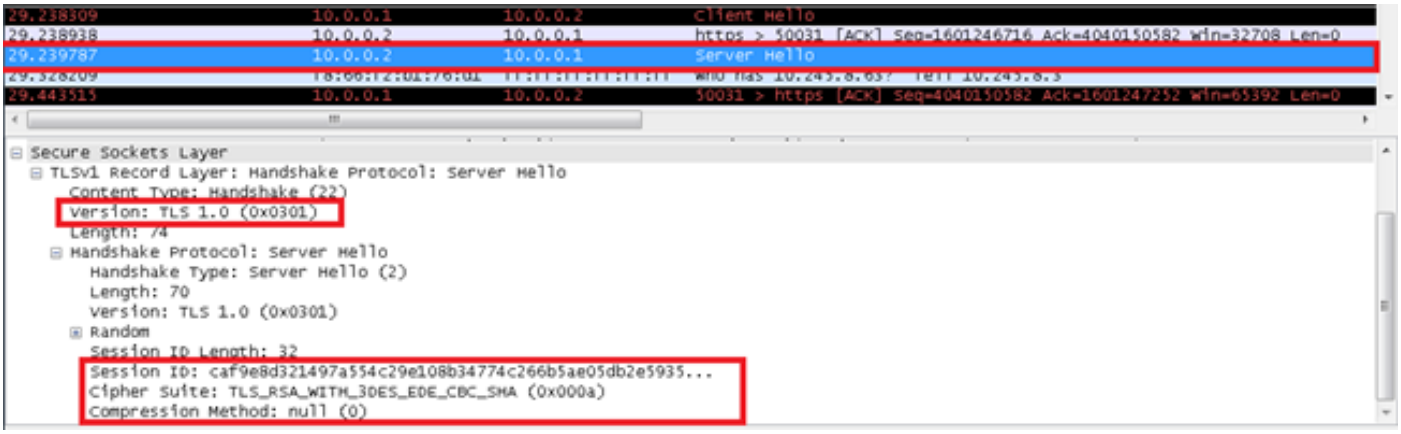
Server Hello

The server sends back these attributes to the client:

- **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** in order 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.

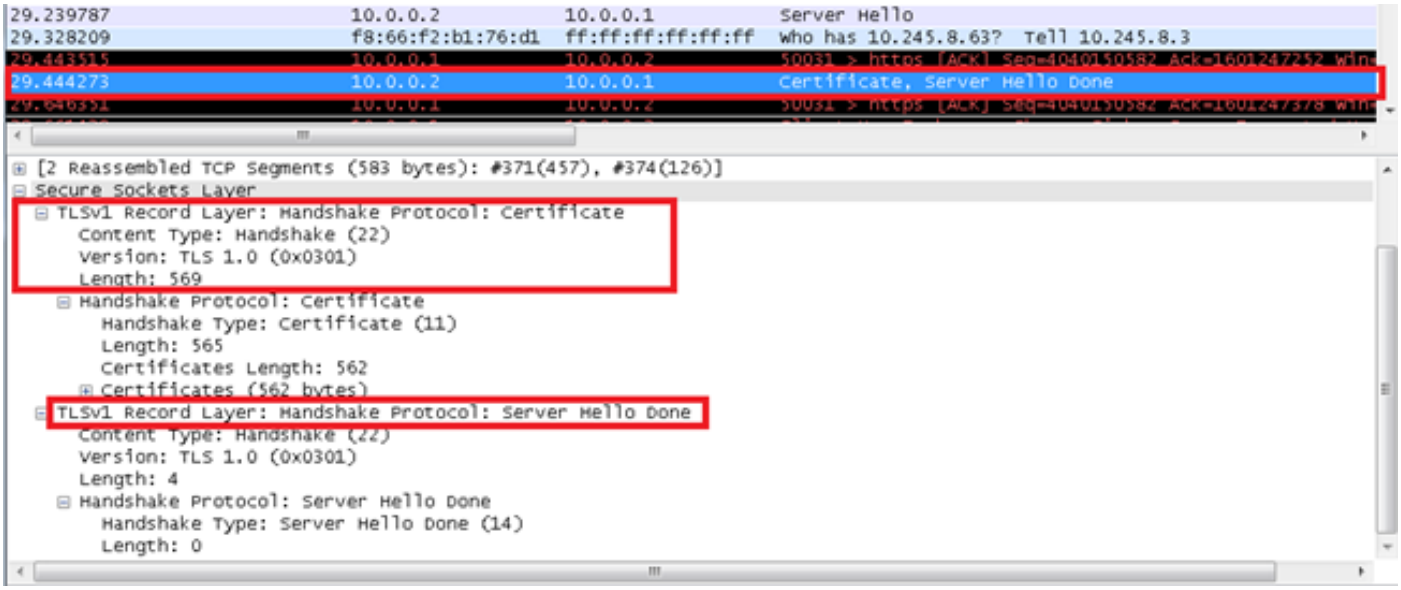


For SSL session resumption requests:

- The server can send a Hello request to the client as well. This is only to remind the client that it should start the renegotiation with a Client Hello request when convenient. The client ignores the Hello request from the server if the handshake process is already underway.
- The handshake messages have more precedence over the transmission of application data. The renegotiation must begin in no more than one or two times the transmission time of a maximum-length application data message.

Server Hello Done

The Server Hello Done message is sent by the server in order to indicate the end of the server hello and associated messages. After it sends this message, the server waits for a client response. Upon receipt of the Server Hello Done message, the client verifies that the server provided a valid certificate, if required, and checks that the Server Hello parameters are acceptable.



Server Certificate, Server Key Exchange, and Certificate Request (Optional)

- **Server Certificate:** If the server must be authenticated (which is generally the case), the server sends its certificate immediately after the Server Hello message. The certificate type must be appropriate for the selected cipher suite key exchange algorithm, and is generally an X.509.v3 certificate.
- **Server Key Exchange:** The Server Key Exchange message is sent by the server if it has no certificate. If the Diffie–Hellman (DH) parameters are included with the server certificate, this message is not used.
- **Certificate Request:** A server can optionally request a certificate from the client, if appropriate for the selected cipher suite.

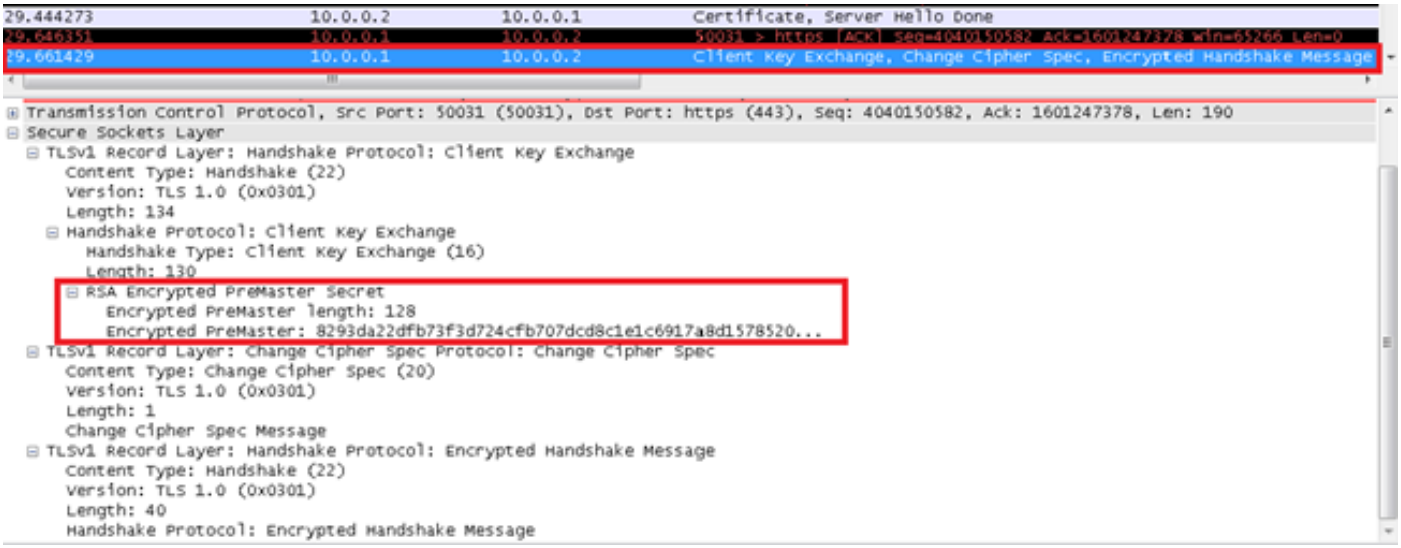
Client Exchange

Client Certificate (Optional)

This is the first message that the client sends after he/she receives a Server Hello Done message. This message is only sent if the server requests a certificate. If no suitable certificate is available, the client sends a **no_certificate** alert instead. This alert is only a warning; however, the server might respond with a fatal handshake failure alert if client authentication is required. Client DH certificates must match the server specified DH parameters.

Client Key Exchange

The content of this message depends on the public key algorithm selected between the Client Hello and the Server Hello messages. The client uses either a premaster key encrypted by the Rivest-Shamir-Addleman (RSA) algorithm or DH for key agreement and authentication. When RSA is used for server authentication and key exchange, a 48-byte **pre_master_secret** is generated by the client, encrypted under the server public key, and sent to the server. The server uses the private key in order to decrypt the **pre_master_secret**. Both parties then convert the **pre_master_secret** into the **master_secret**.



Certificate Verify (Optional)

If the client sends a certificate with signing ability, a digitally-signed Certificate Verify message is sent in order to explicitly verify the certificate.

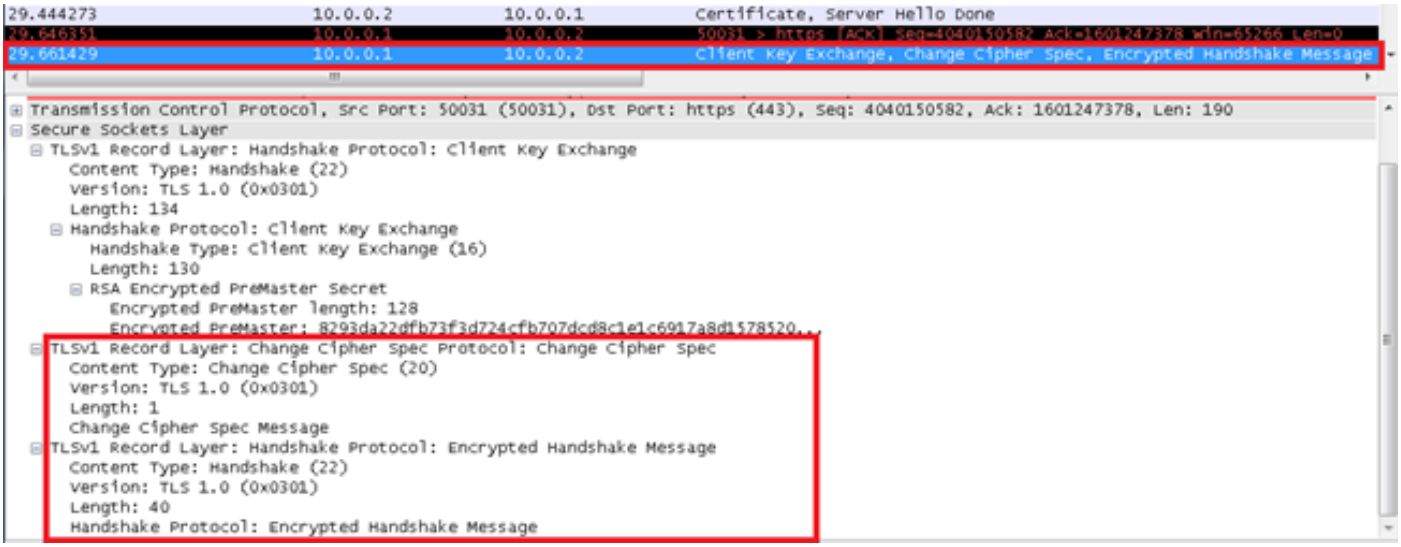
Cipher Change

Change Cipher Spec Messages

The Change Cipher Spec message is sent by the client, and the client copies the pending Cipher Spec (the new one) into the current Cipher Spec (the one that was previously used). Change Cipher Spec protocol exists in order to signal transitions in ciphering strategies. The protocol consists of a single message, which is encrypted and compressed under the current (not the pending) Cipher Spec. The message is sent by both the client and server in order to notify the receiving party that subsequent records are protected under the most recently negotiated Cipher Spec and keys. Reception of this message causes the receiver to copy the read pending state into the read current state. The client sends a Change Cipher Spec message after the handshake key exchange and Certificate Verify messages (if any), and the server sends one after it successfully processes the key exchange message it received from the client. When a previous session is resumed, the Change Cipher Spec message is sent after the Hello messages. In the captures, the Client Exchange, Change Cipher, and Finished messages are sent as a single message from the client.

Finished Messages

A Finished message is always sent immediately after a Change Cipher Spec message in order to verify that the key exchange and authentication processes were successful. The Finished message is the first protected packet with the most recently negotiated algorithms, keys, and secrets. No acknowledgment of the Finished message is required; parties can begin to send encrypted data immediately after they send the Finished message. Recipients of Finished messages must verify that the contents are correct.



Related Information

- [RFC 6101 - The Secure Sockets Layer Protocol Version 3.0]
- [Wireshark SSL wiki] - decrypt SSL packets with Wireshark
- [Technical Support & Documentation - Cisco Systems]