Secure Socket Layer

What is SSL?

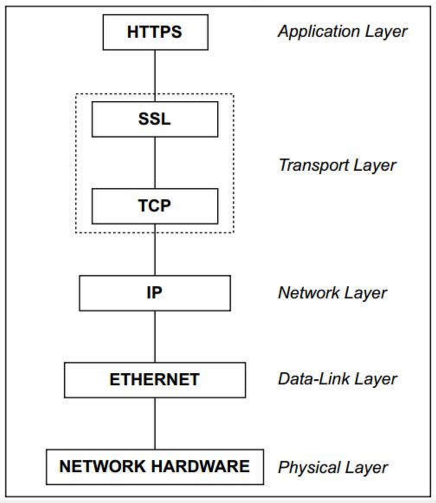
SSL is the ubiquitous security protocol used in almost 100% of secure Internet transactions. Essentially, SSL transforms a typical reliable transport protocol (such as TCP) into a secure communications channel suitable for conducting sensitive transactions.[i] The SSL protocol defines the methods by which a secure communications channel can be established—it does not indicate which cryptographic algorithms need to be used. SSL supports many different algorithms, and serves as a framework whereby cryptography can be used in a convenient and distributed manner.

Uses for SSL

The uses for SSL are endless. Any application that needs to transmit data over an unsecured network such as the Internet or a company intranet is a potential candidate for SSL. SSL provides security, and more importantly, peace of mind. When using SSL, you can be fairly sure that your data are safe from eavesdroppers and tampering.

• The Internet-enabled vending machine can now become a reality—SSL makes tampering with communications almost impossible.
• Home automation systems can be Internet-enabled—forgot to turn off the oven? Just log into your house from your computer at work and turn it off. SSL provides a secure means of protecting your home from hackers.
• Readings from medical devices can be sent over a standard network—SSL protects your privacy.
• Make a telephone switch Web-configurable—SSL encrypts all data, so no one monitoring the network can read your information. Since Web-based access means that your data will likely be travelling over a competitor's network, SSL makes a lot of sense.
• Remote-entry configuration—change the passcode on all the doors of a building simultaneously. SSL protects the passcode, allowing the doors to be connected to a standard corporate network, no need for expensive proprietary hardware!

Figure 1. How SSL Fits Into the 5-Layer TCP/IP Reference Model



SSL Walkthrough

This walkthrough explains the setup and execution of a simple HTTPS server on a Rabbit-based device. There are six steps, which are listed here.

1. Create a digital certificate.
2. Import the certificate.
3. Set up TCP/IP for the sample application.
4. Set up the application to use SSL.
5. Set up the Web browser.
6. Run the application.

SSL Basics
Although SSL is simple in theory (keys are exchanged using public-key cryptography, communication is done using symmetric-key cryptography), the actual implementation is quite complex. This section briefly covers the details of establishing an SSL connection and communicating using that connection.

SSL Alerts
One of the most important components of SSL is its error-handling system. SSL errors are called *alerts*, and represent possible attacks. Alerts are messages sent across the SSL communication channel, and may be encrypted. The SSL specification details about 20 different alerts and gives guidelines on how to handle them when received, and when to generate and send them. Error handling is implementation-specific, and will be covered later in this document.

The Handshake
SSL communication takes place in an SSL *session*. The session is established using a handshake process similar to the TCP 3-way handshake. The entire handshake, including the establishing the TCP/IP socket. The TCP/IP connection is established first, and then the SSL handshake begins. The SSL session is established when both the client and server are communicating using the negotiated parameters and ciphers. The SSL session ends when either side is done transmitting application data and notifies the other machine that it is done sending data.

The Client Hello and Public-Key Operation
All SSL sessions begin with a *Client Hello* message. This message is sent by the client to the server it wishes to communicate with. This message contains the client's version of SSL, a random number used later in key derivation, as well as a collection of ciphersuite *offers*. The offers are identifiers that specify the ciphers and hashing algorithms the client is willing to use.
When establishing the initial connection, the server chooses an offer it is willing to use, and communicates that offer back to the client along with its certificate and a random value of its own. The client then verifies the server using the certificate and extracts the server's public key. Using the public key, the client encrypts the *pre-master secret*, a random value that will be used to generate the symmetric keys independently, and sends the encrypted message to the server, which decrypts the message using its private key.

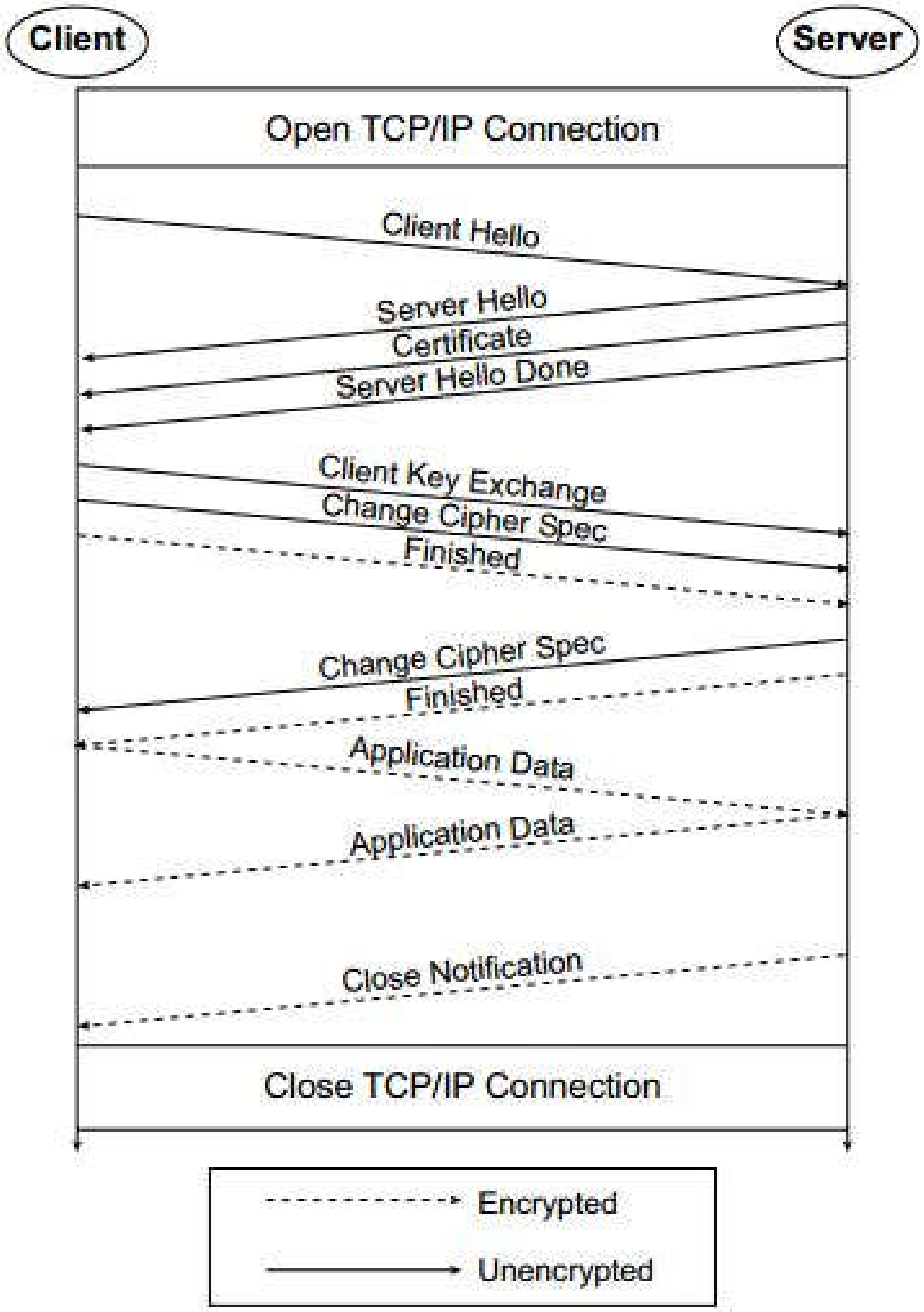


Figure 2. SSL Handshake

Symmetric-Key Derivation

Once the server receives the pre-master secret from the client, both the server and the client generate the *same* symmetric keys using the pre-master secret and the random numbers exchanged above using the TLS pseudo-random function (PRF), which expands a secret and some data into a block of arbitrary length.i This way, only the small pre-master secret is encrypted using public-key cryptography, limiting the impact of the expensive operation on performance.

Handshake Finish

As soon as the keys are generated, the client and server exchange *change cipher spec* messages to indicate that they each now have symmetric keys and all further communications will be conducted using the symmetric algorithm chosen in the initial stages of the handshake. At this point, the server and client take all the handshake messages received and sent, and generate a block of data used to verify that the handshake was not tampered with. These data, generated using the TLS PRF, are sent in a final handshake message, *Finish*. If the data in the finish message do not match the locally generated finish data, then the connection is terminated by whoever failed the finish verification test.

SSL Session

Once the handshake is finished, the server and client begin to communicate over the newly established secure channel. Each message is hashed, encrypted, and sent. If at anytime there is a failure, either in the decryption, encryption, hashing, verification, or communication, an SSL alert is sent (using the symmetric encryption) by the entity experiencing the failure. Most alerts are fatal, causing the communication to stop immediately.

Ending the Session

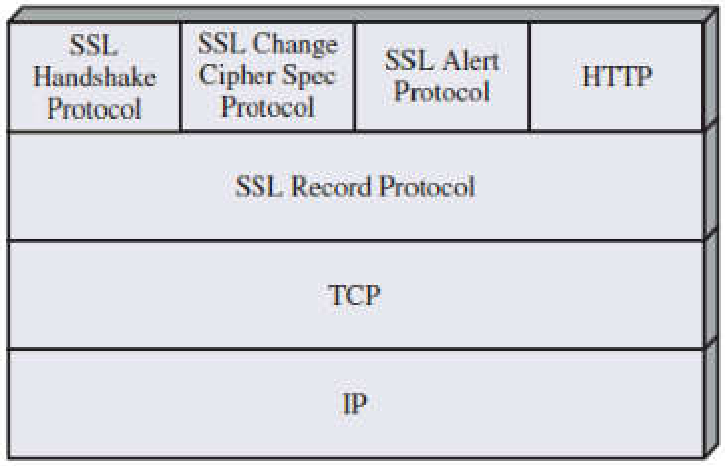
When the client or server is done communicating, a special alert, *close_notify*, is sent to ensure that all communications have ceased and the connection can be closed. This alert prevents an adversary from performing a *truncation attack*, fooling the server or client into thinking that all the data to be exchanged have been sent, when actually there are some data left (this can be a problem in situations such as banking transactions, where it is necessary for *all* information to be received).

SSL Security

The elaborate and expensive establishment of an SSL session is the result of many years of studying the various attacks performed against SSL and other secure protocols. Some attacks target the cryptography implementations, others target the PRNG. Some attacks can use knowledge of the information being sent to derive secret information. Some attacks even use the timing of certain algorithms to derive secrets. SSL and TLS address many of these issues in its protocol design, but even more importantly, implementations must follow the protocol for those safeguards to work. Beyond that, implementations must also not be subject to other forms of attack, such as buffer overflow.

SSL Protocol

- SSL is designed to make use of TCP to provide a reliable end-to-end secure service.
- SSL is not a single protocol but rather two layers of protocols, as illustrated in Figure below.



- The SSL Record Protocol provides basic security services to various higher layer Protocols.
- In particular, the Hypertext Transfer Protocol (HTTP), which providesthe transfer service for Web client/server interaction, can operate on top ofSSL.
- Three higher-layer protocols are defined as part of SSL:

1. the Handshake Protocol,
2. The Change Cipher Spec Protocol,
3. The Alert Protocol.

- These SSL-specificprotocols are used in the management of SSL exchanges and are examinedlater in this section.
- Two important SSL concepts are the SSL session and the SSL connection,which are defined in the specification as follows.

  1. Connection:

  - A connection is a transport (in the OSI layering model definition that provides a suitable type of service.
  - For SSL, such connections arepeer-to-peer relationships. The connections are transient. Every connection isassociated with one session.

  2. Session:

  - An SSL session is an association between a client and a server.
  - Sessionsare created by the Handshake Protocol. Sessions define a set of cryptographicsecurity parameters which can be shared among multiple connections.
  - Sessions are used to avoid the expensive negotiation of new security parameters for each connection.

Handshake Protocol

- This protocol allows the server and client to authenticate each other and to negotiate an encryption and MAC algorithm and cryptographic keys to be used to protect data sent in an SSL record.

- The Handshake Protocol is used before any application data is transmitted. The Handshake Protocol consists of a series of messages exchanged by client and server.

  Type (1 byte): Indicates one of 10 messages.

  Length (3 bytes): The length of the message in bytes.

  Content (bytes): The parameters associated with this message

Phase 1

Establishing Security Capabilities

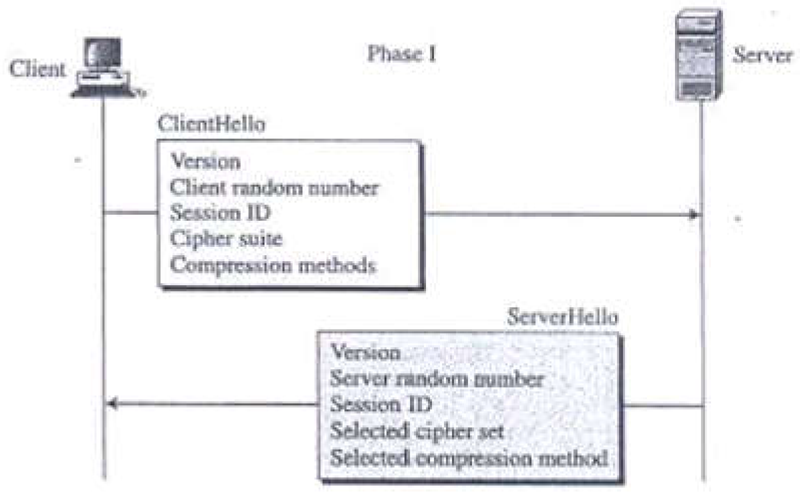
- The exchange is initiated by the client, which sends a client_hellomessage with the following parameters:

  Version: The highest SSL version understood by the client.

Random: A client-generated random structure consisting of a 32-bit timestamp and 28 bytes generated by a secure random number generator. These values serve as nonces and are used during key exchange to prevent replay attacks.

Session ID: A variable-length session identifier. A nonzero value indicates that the client wishes to update the parameters of an existing connection or to create a new connection on this session. A zero value indicates that the client wishes to establish a new connection on a new session.

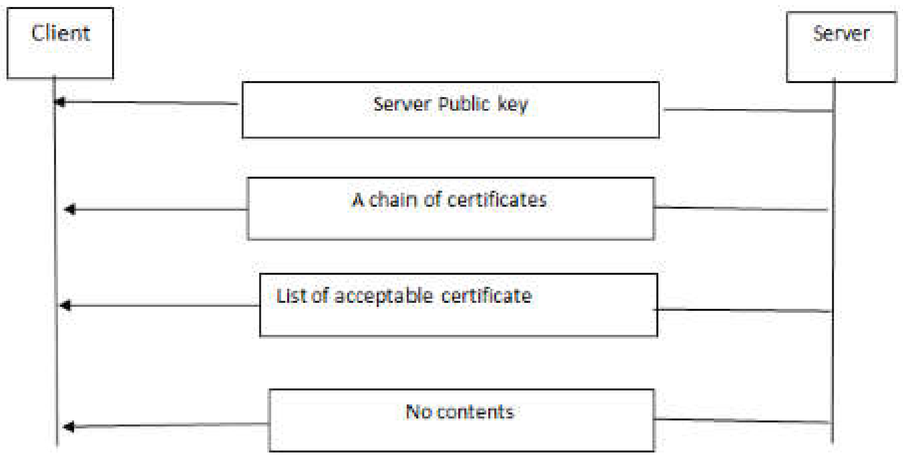
CipherSuite: This is a list that contains the combinations of cryptographic algorithms supported by the client, in decreasing order of preference. Each element of the list (each cipher suite) defines both a key exchange algorithm and a CipherSpec



Phase 2.

Server Authentication and Key Exchange

- In this phase the server authenticates itself if needed.
- The sender may sends its certificate, its public key and may also request certificate from client.
- At the end server announces that the server Hello process is been done.



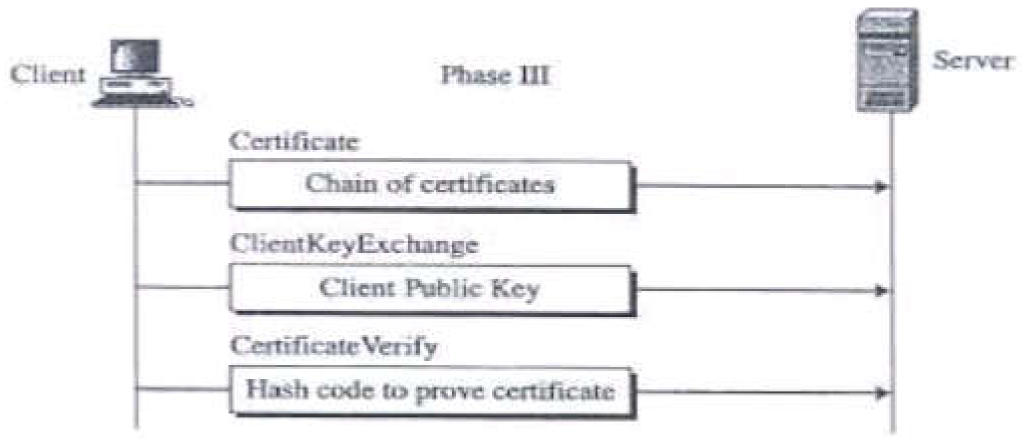
After phase 2

- The server is authenticated to the client.
- The client knows the public key of the server if required.

Phase 3

Client Key Exchange and Authentication

Phase 3 is designed to authenticate the client upto 3 messages are send from client to server.



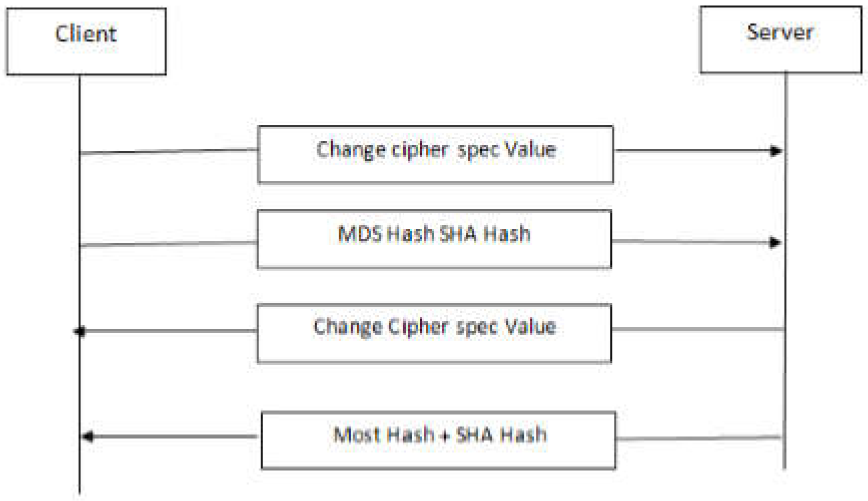
After phase 3

- The client is authenticated for the server
- Both client and server knows pre- master secret.

Phase 4

Finalizing And Finishing

- In phase 4 client and server send messages to change cipher specification and to finish the handshake protocol
- Four messages are exchanged in this phase as shown below:


Client
Server
Change cipher spec Value
MDS Hash SHA Hash
Change Cipher spec Value
Most Hash + SHA Hash