Chapter 11

The Secure Sockets Layer (SSL)

Due to the fact that nearly all businesses have websites (as well as government agencies and individuals) a large enthusiasm exists for setting up facilities on the Web for electronic commerce. Of course there are major security issues involved here that need to be addressed. Nobody wants to send their credit card number over the Internet unless they have a guarantee that *only* the intended recipient will receive it. As businesses begin to see the threats of the Internet to electronic commerce, the demand for secure web pages grows.

A number of approaches to providing Web security are possible. The various approaches are similar in many ways but may differ with respect to their scope of applicability and relative location within the TCP/IP protocol stack. For example we can have security at the IP level making it transparent to end users and applications. However another relatively general-purpose solution is to implement security just above TCP. The foremost example of this approach is the Secure Sockets Layer (SSL) and the follow-on Internet standard known as Transport Layer Security (TLS). This chapter looks at SSL which was originated by Netscape. The Internet standard, TLS, can be viewed essentially as SSLv3.1 and is very close to and backward compatible with SSLv3. We will mainly be interested in SSLv3 at present.

11.1 Overview

As mentioned, the Secure Sockets Layer (SSL) is a method for providing security for web based applications. It 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 11.1. It can be seen that one layer makes use of TCP directly. This layer is known as the **SSL Record Protocol** and it provides basic security services to various higher layer protocols. An independent protocol that makes use of the record protocol is the Hypertext Markup Language (HTTP) protocol. Another three higher level protocols that also make use of this layer are part of the SSL stack. They are used in the management of SSL exchanges and are as follows:

1. Handshake Protocol.
2. Change Cipher Spec Protocol.
3. Alert Protocol.

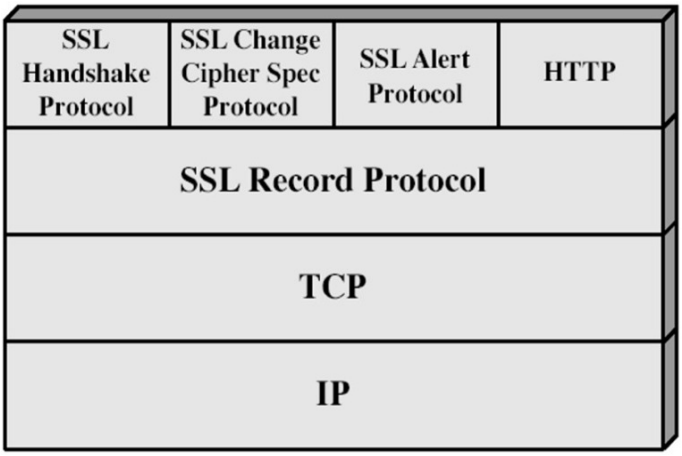

Figure 11.1: SSL protocol stack.

The **SSL record protocol**, which is at a lower layer and offers services to these three higher level protocols, is discussed first.

11.2 SSL Record Protocol

This protocol provides two services for SSL connections:

1. Confidentiality - using conventional encryption.
2. Message Integrity - using a Message Authentication Code (MAC).

In order to operate on data the protocol performs the following actions (see figure 11.2):

- It takes an application message to be transmitted and fragments it into manageable blocks. These blocks are $2^{14} = 16,384$ bytes or less.
- These blocks are then optionally compressed which must be lossless and may not increase the content length by more than 1024 bytes.
- A message authentication code is then computed over the compressed data using a shared secret key. This is then appended to the compressed (or plaintext) block.
- The compressed message plus MAC are then encrypted using symmetric encryption. Encryption may not increase the content length by more than 1024 bytes, so that the total length may not exceed $2^{14} + 2048$. A number of different encryption algorithms are permitted.
- The final step is to prepend a header, consisting of the following fields:

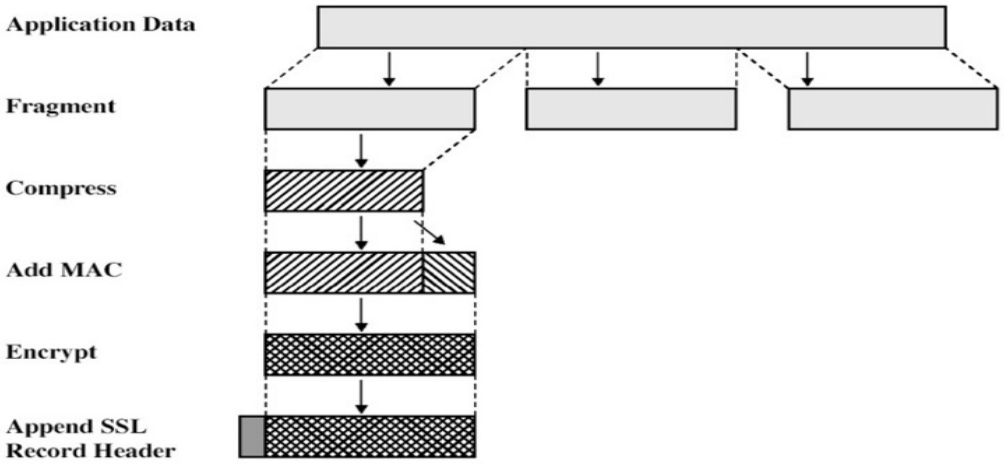

Figure 11.2: SSL Record Protocol Operation.

- Content type (8 bits) - The higher layer protocol used to process the enclosed fragment.
- Major Version (8 bits) - Indicates major version of SSL in use. For SSLv3, the value is 3.
- Minor Version (8 bits) - Indicates minor version in use. For SSLv3, the value is 0.
- Compressed Length (16 bits) - The length in bytes of the compressed (or plaintext) fragment.

The overall format is shown in figure 11.3.

The "content type" above is one of four types; the three higher level protocols given above that make use of the SSL record, and a fourth known as "application_data". The first three are described next as they are SSL specific protocols.

11.3 Change Cipher Spec Protocol

This consists of a single message which consists of a single byte with the value 1. This is used to cause the pending state to be copied into the current state which updates the cipher suite to be used on this connection.

11.4 Alert Protocol

This protocol is used to convey SSL-related alerts to the peer entity. It consists of two bytes the first of which takes the values 1 (warning) or 2 (fatal). If the level is fatal SSL immediately terminates the connection. The second byte contains a code that indicates

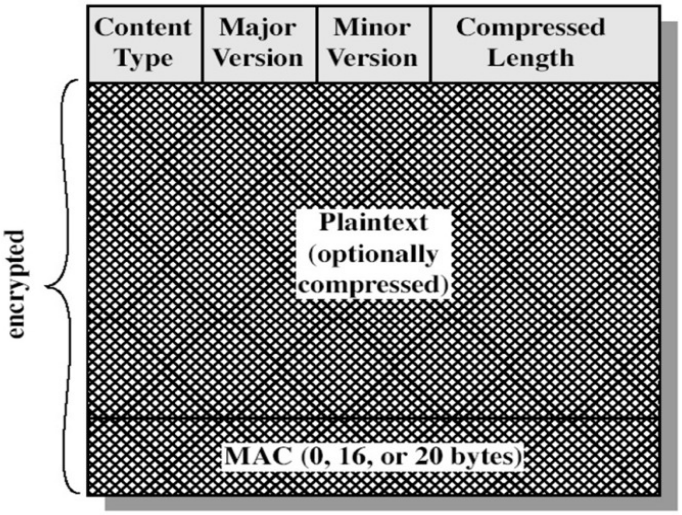

Figure 11.3: SSL record format.

the specific alert.

11.5 Handshake Protocol

This is the most complex part of SSL and 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. This protocol is used before any application data is sent. It consists of a series of messages exchanged by the client and server, all of which have the format shown in figure 11.5. Each message has three fields:

1. Type (1 byte): Indicates one of 10 messages such as "hello_request" (see figure 11.4).
2. Length (3 bytes): The length of the message in bytes.
3. Content(≥ 0 byte): The parameters associated with this message such version of SSL being used.

The Handshake Protocol is shown in figure 11.6. This consists of four phases:

1. Establish security capabilities including protocol version, session ID, cipher suite, compression method and initial random numbers. This phase consists of the client_hello and server_hello messages which contain the following (this is for the client however it's a little different for the server):
   - Version: The highest SSL version understood by client

Message Type	Parameters
hello_request	null
client_hello	version, random, session id, cipher suite, compression method
server_hello	version, random, session id, cipher suite, compression method
certificate	chain of X.509v3 certificates
server_key_exchange	parameters, signature
certificate_request	type, authorities
server_done	null
certificate_verify	signature
client_key_exchange	parameters, signature
finished	hash value

Figure 11.4: SSL Handshake protocol message types.

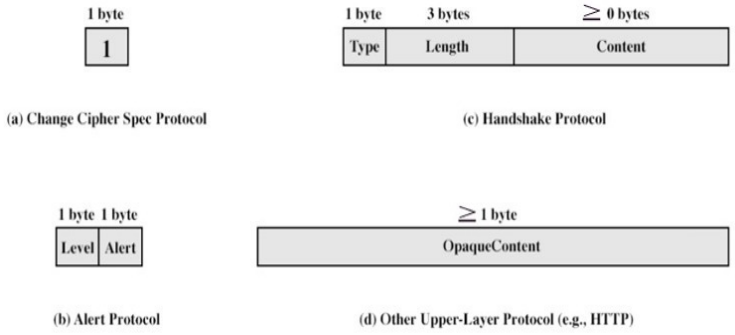

Figure 11.5: SSL record protocol payload.

- Random: 32-bit timestamp and 28 byte nonce.
- Session ID: A variable length session identifier.
- CipherSuite: List of cryptoalgorithms supported by client in decreasing order of preference. Both key exchange and CipherSpec (this includes fields such as CipherAlgorithm, MacAlgorithm, CipherType, HashSize, Key Material and IV Size) are defined.
- Compression Method: List of methods supported by client.

2. Server may send certificate, key exchange, and request certificate it also signals end of hello message phase. The certificate sent is one of a chain of X.509 certificates discussed earlier in the course. The server_key exchange is sent only if required. A certificate may be requested from the client if needs be by certificate_request.

3. Upon receipt of the server_done message, the client should verify that the server provided a valid certificate, if required, and check that the server_hello parameters are acceptable. If all is satisfactory, the client sends one or more messages

back to the server. The client sends certificate if requested (if none available then it sends a no_certificate alert instead). Next the client sends client_key_exchange message . Finally, the client may send certificate verification.

4. Change cipher suite and finish handshake protocol. The secure connection is now setup and the client and server may begin to exchange application layer data.

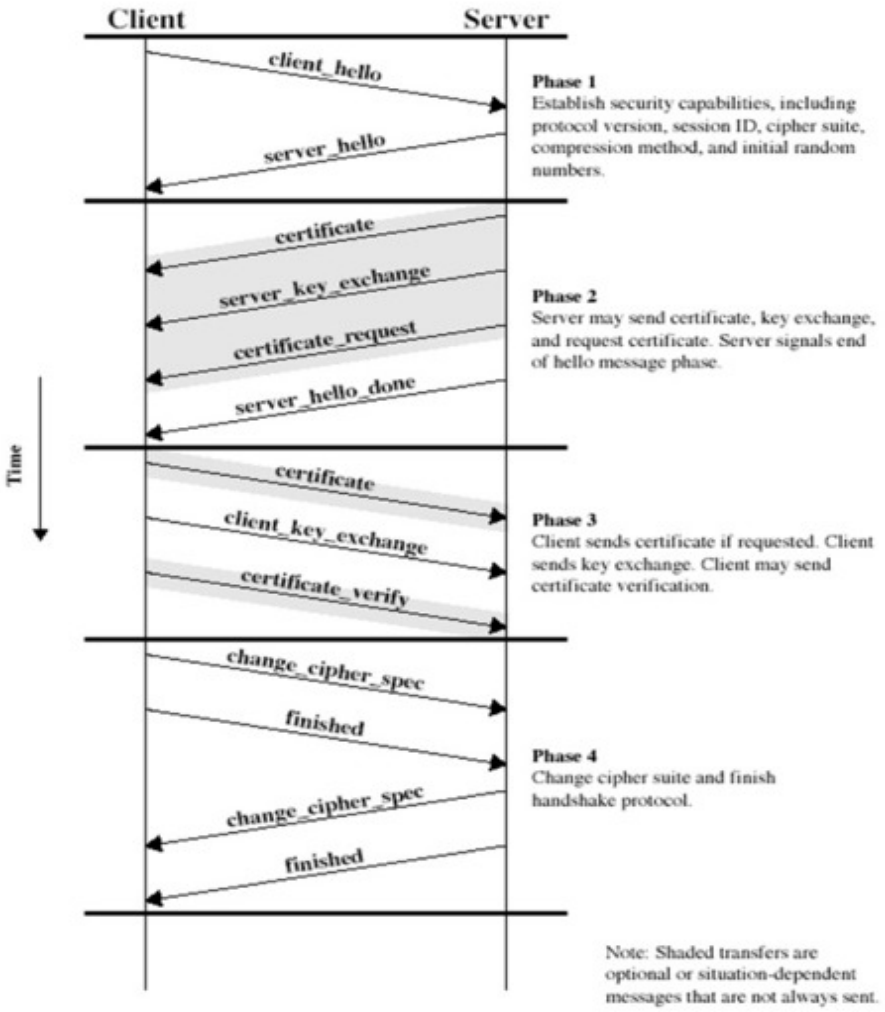

Figure 11.6: Handshake protocol action.