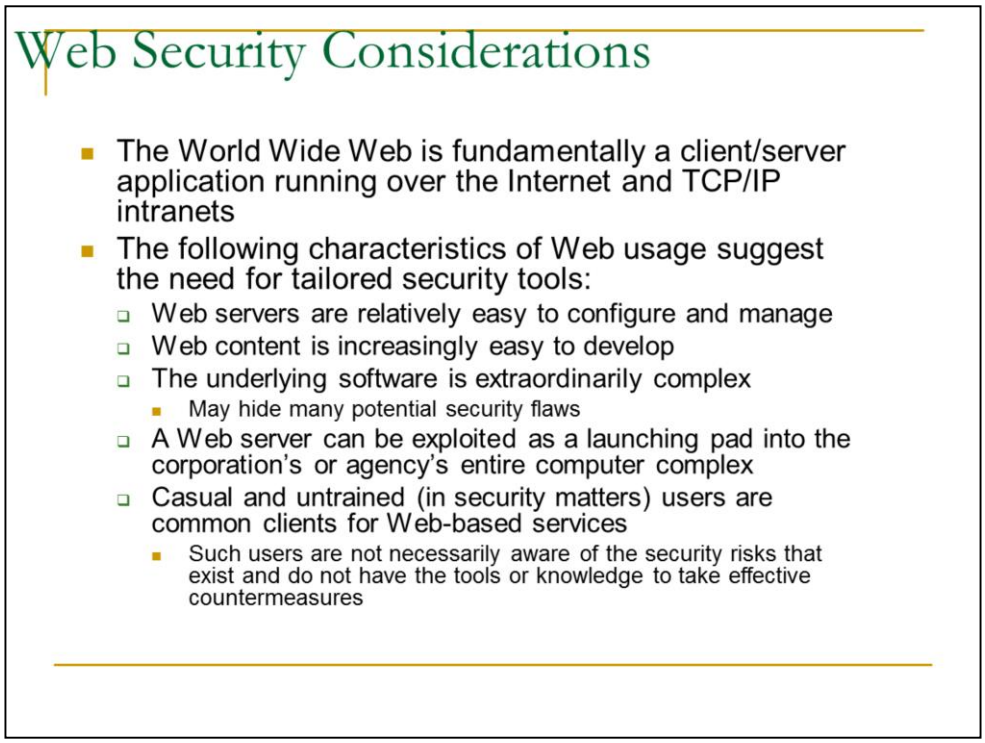
Plan of Talk

- **Transport Layer Security**
- **SSL**



The World Wide Web is fundamentally a client/server application running over the

Internet and TCP/IP intranets. As such, the security tools and approaches discussed

so far in this book are relevant to the issue of Web security. However, the following

characteristics of Web usage suggest the need for tailored security tools:

• Although Web browsers are very easy to use, Web servers are relatively easy

to configure and manage, and Web content is increasingly easy to develop, the

underlying software is extraordinarily complex. This complex software may

hide many potential security flaws. The short history of the Web is filled with

examples of new and upgraded systems, properly installed, that are vulnerable

to a variety of security attacks.

• A Web server can be exploited as a launching pad into the corporation's or agency's entire computer complex. Once the Web server is subverted, an attacker may be able to gain access to data and systems not part of the Web itself but connected to the server at the local site.

• Casual and untrained (in security matters) users are common clients for Web-based services. Such users are not necessarily aware of the security risks that exist and do not have the tools or knowledge to take effective countermeasures.

	Threats	Consequences	Countermeasures
Integrity	•Modification of user data •Trojan horse browser •Modification of memory •Modification of message traffic in transit	•Loss of information •Compromise of machine •Vulnerabilty to all other threats	Cryptographic checksums
Confidentiality	•Eavesdropping on the net •Theft of info from server •Theft of data from client •Info about network configuration •Info about which client talks to server	•Loss of information •Loss of privacy	Encryption, Web proxies
Denial of Service	•Killing of user threads •Flooding machine with bogus requests •Filling up disk or memory •Isolating machine by DNS attacks	•Disruptive •Annoying •Prevent user from getting work done	Difficult to prevent
Authentication	•Impersonation of legitimate users •Data forgery	•Misrepresentation of user •Belief that false information is valid	Cryptographic techniques

Table 17.1 A Comparison of Threats on the Web

Table 17.1 provides a summary of the types of security threats faced when using

the Web. One way to group these threats is in terms of passive and active attacks.

Passive attacks include eavesdropping on network traffic between browser and

server and gaining access to information on a Web site that is supposed to be restricted.

Active attacks include impersonating another user, altering messages in

transit between client and server, and altering information on a Web site.

Another way to classify Web security threats is in terms of the location of the

threat: Web server, Web browser, and network traffic between browser and server.

Issues of server and browser security fall into the category of computer system security;

Part Six of this book addresses the issue of system security in general but is also

applicable to Web system security. Issues of traffic security fall into the

category of
network security and are addressed in this chapter.

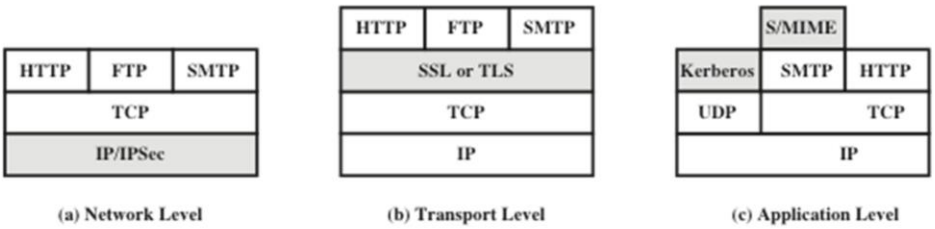


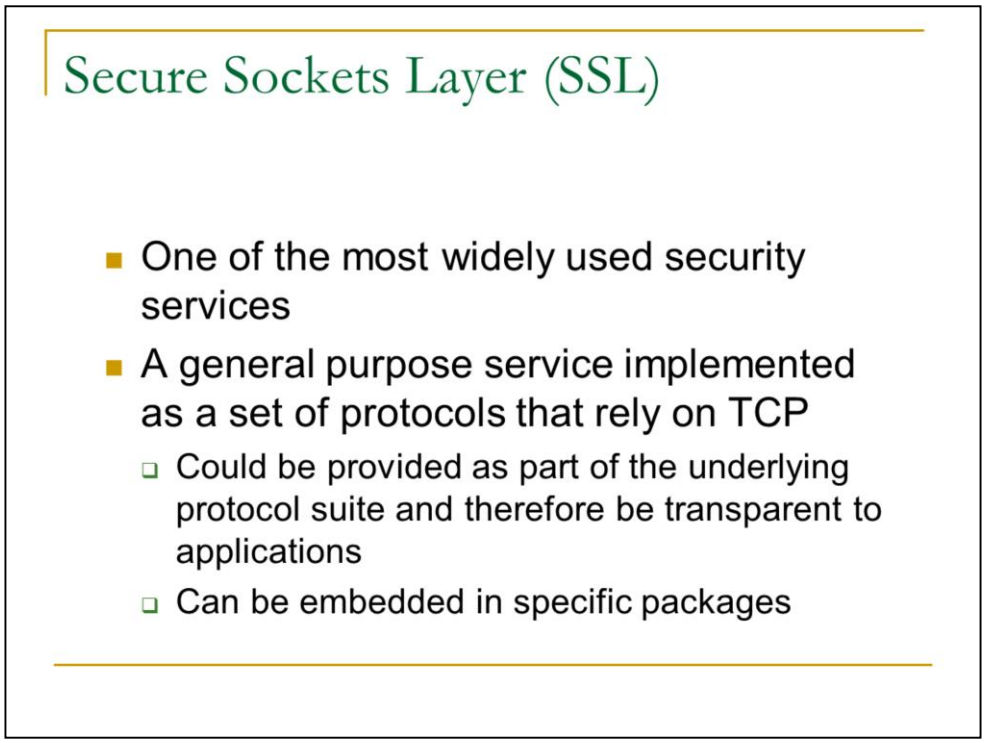
Figure 17.1 Relative Location of Security Facilities in the TCP/IP Protocol Stack

A number of approaches to providing Web security are possible. The various approaches
that have been considered are similar in the services they provide and, to some extent, in the mechanisms that they use, but they differ with respect to their scope of applicability and their relative location within the TCP/IP protocol stack.

Figure 17.1 illustrates this difference. One way to provide Web security is to use IP security (IPsec) (Figure 17.1a). The advantage of using IPsec is that it is transparent to end users and applications and provides a general-purpose solution. Furthermore, IPsec includes a filtering capability so that only selected traffic need incur the overhead of IPsec processing.

Another relatively general-purpose solution is to implement security just above TCP (Figure 17.1b). The foremost example of this approach is the Secure Sockets Layer (SSL) and the follow-on Internet standard known as Transport Layer Security (TLS). At this level, there are two implementation choices. For full generality, SSL (or TLS) could be provided as part of the underlying protocol suite and therefore be transparent to applications. Alternatively, SSL can be embedded in specific packages. For example, Netscape and Microsoft Explorer browsers come equipped with SSL, and most Web servers have implemented the protocol.

Application-specific security services are embedded within the particular application. Figure 17.1c shows examples of this architecture. The advantage of this approach is that the service can be tailored to the specific needs of a given application.



One of the most widely used security services is the Secure Sockets Layer (SSL) and the follow-on Internet standard known as Transport Layer Security (TLS), the latter defined in RFC 5246. SSL is a general-purpose service implemented as a set of protocols that rely on TCP. At this level, there are two implementation choices. For full generality, SSL (or TLS) could be provided as part of the underlying protocol suite and therefore be transparent to applications. Alternatively, SSL can be embedded in specific packages. For example, most browsers come equipped with SSL, and most Web servers have implemented the protocol.

This section is devoted to a discussion of SSLv3, and next section describes the principal differences between SSLv3 and TLS.

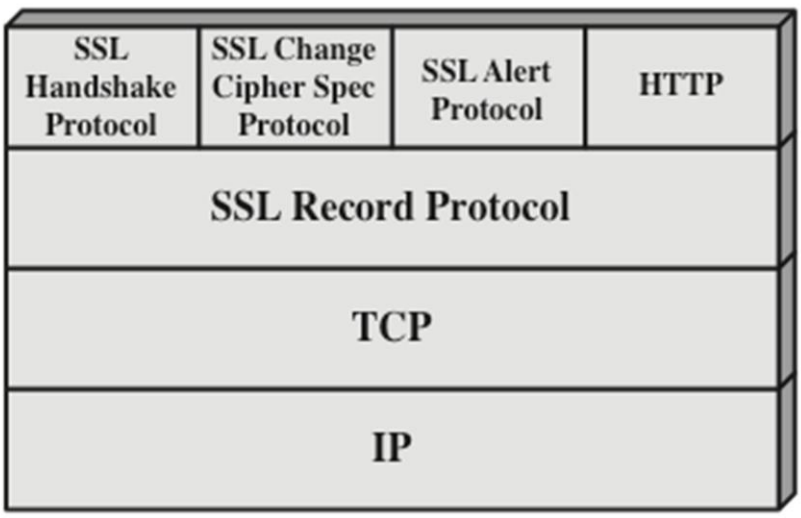


Figure 17.2 SSL Protocol Stack

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 17.2.

The SSL Record Protocol provides basic security services to various higher layer

protocols. In particular, the Hypertext Transfer Protocol (HTTP), which

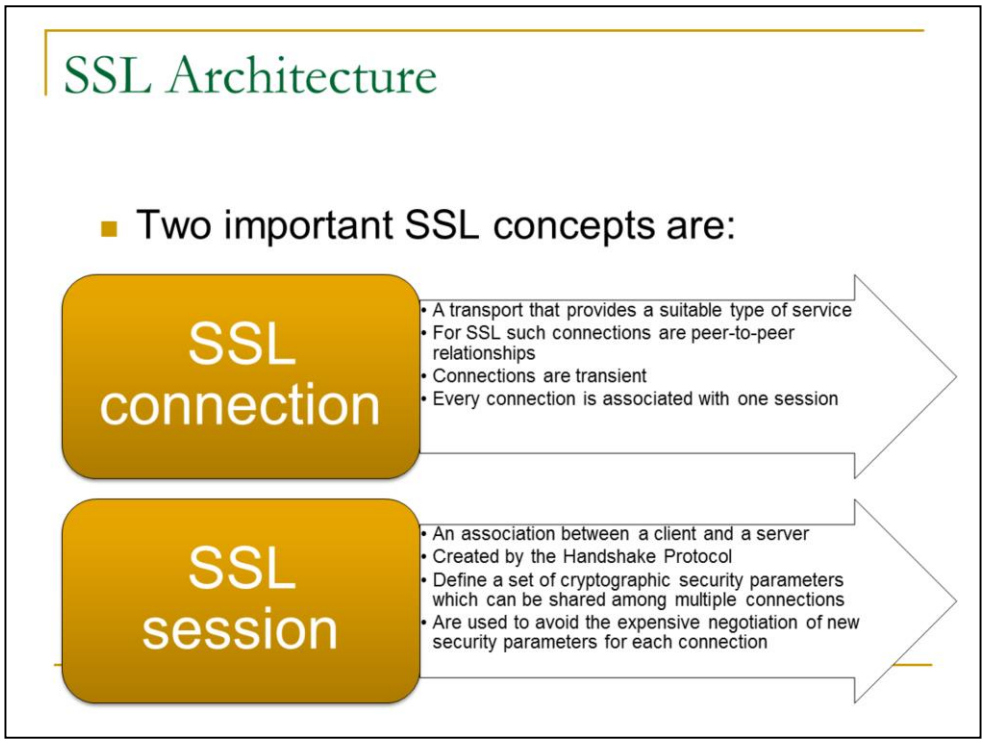
provides the transfer service for Web client/server interaction, can operate on top

of SSL. Three higher-layer protocols are defined as part of SSL: the Handshake

Protocol, The Change Cipher Spec Protocol, and the Alert Protocol. These

SSL-specific protocols are used in the management of SSL exchanges and are

examined later in this section.



Two important SSL concepts are the SSL session and the SSL connection,

which are defined in the specification as follows.

• Connection: A connection is a transport (in the OSI layering model definition)

that provides a suitable type of service. For SSL, such connections are

peer-to-peer relationships. The connections are transient. Every connection is

associated with one session.

• Session: An SSL session is an association between a client and a server.

Sessions are created by the Handshake Protocol. Sessions define a set of cryptographic

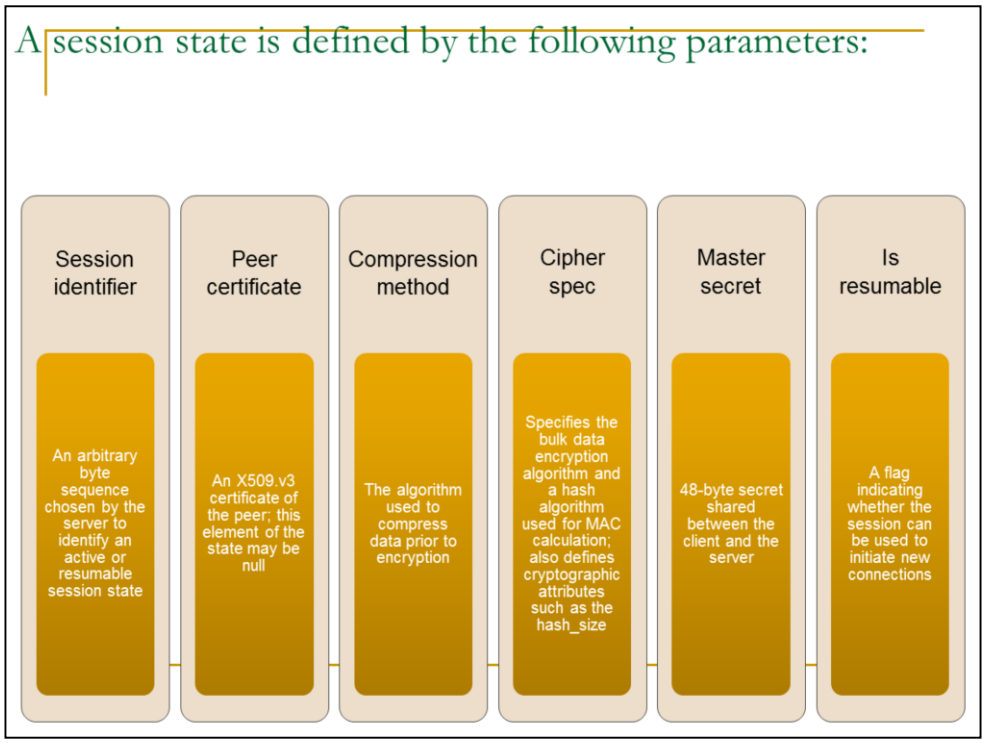
security parameters which can be shared among multiple connections.

Sessions are used to avoid the expensive negotiation of new security

parameters for each connection.

Between any pair of parties (applications such as HTTP on client and server), there may be multiple secure connections. In theory, there may also be multiple simultaneous sessions between parties, but this feature is not used in practice.

There are a number of states associated with each session. Once a session is established,
there is a current operating state for both read and write (i.e., receive and send). In addition, during the Handshake Protocol, pending read and write states are created. Upon successful conclusion of the Handshake Protocol, the pending states become the current states.



A session state is defined by the following parameters.

• Session identifier: An arbitrary byte sequence chosen by the server to identify an active or resumable session state.

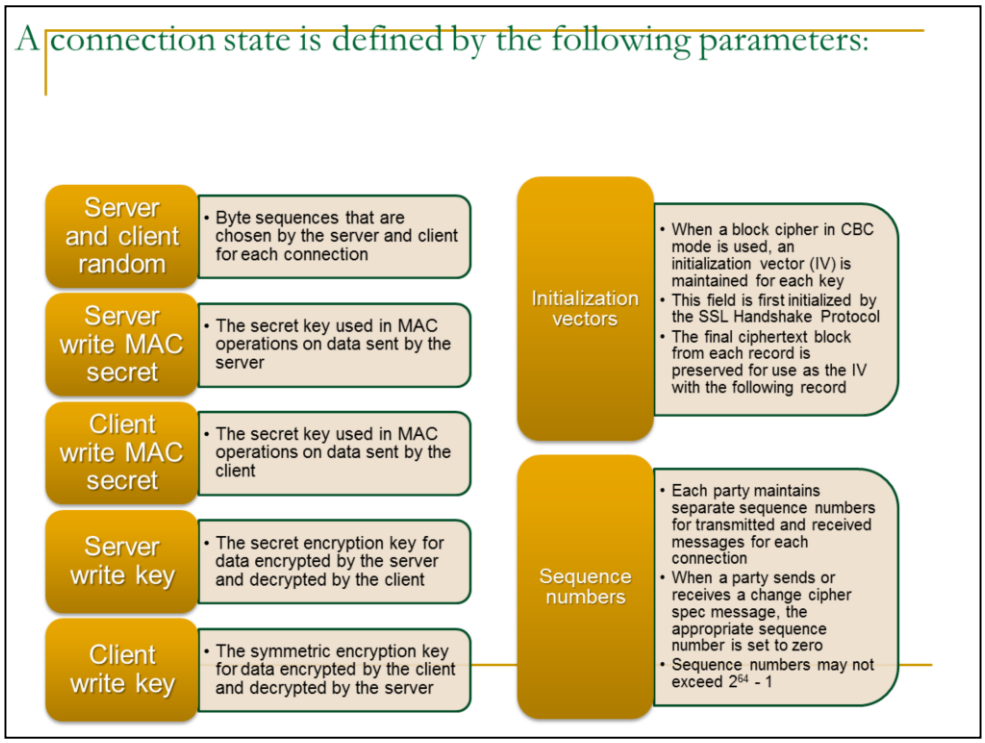
• Peer certificate: An X509.v3 certificate of the peer. This element of the state may be null.

• Compression method: The algorithm used to compress data prior to encryption.

• Cipher spec: Specifies the bulk data encryption algorithm (such as null, AES, etc.) and a hash algorithm (such as MD5 or SHA-1) used for MAC calcula tion. It also defines cryptographic attributes such as the hash_size.

• Master secret: 48-byte secret shared between the client and the server.

• Is resumable: A flag indicating whether the session can be used to initiate new
connections.



A connection state is defined by the following parameters.

• Server and client random: Byte sequences that are chosen by the server and client for each connection.

• Server write MAC secret: The secret key used in MAC operations on data sent by the server.

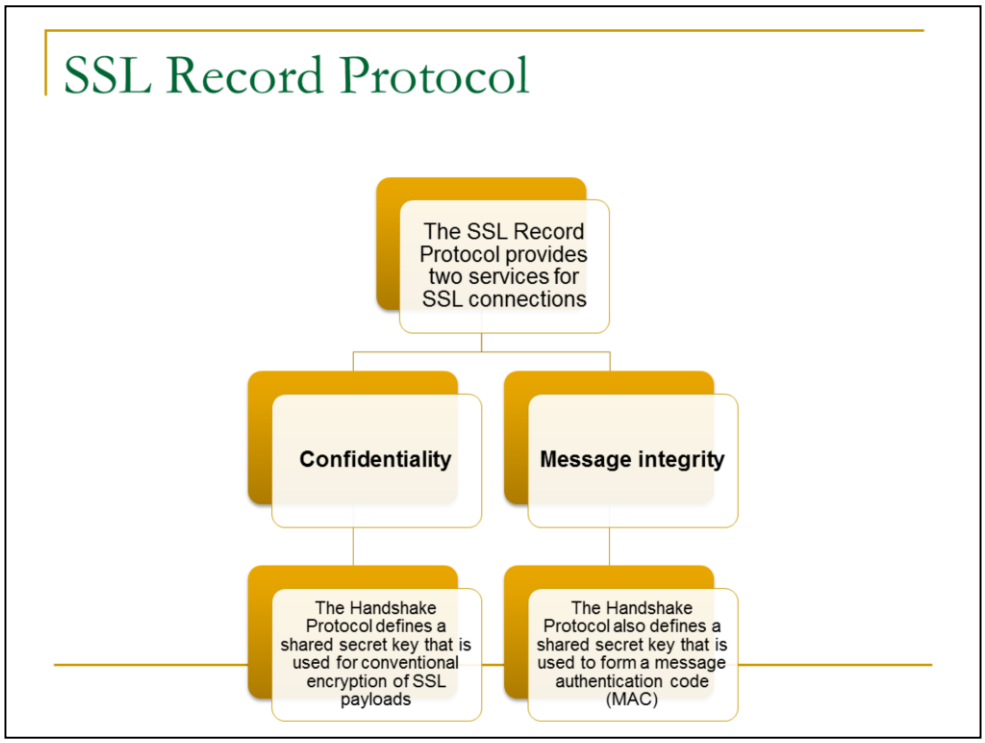
• Client write MAC secret: The secret key used in MAC operations on data sent by the client.

• Server write key: The secret encryption key for data encrypted by the server and decrypted by the client.

• Client write key: The symmetric encryption key for data encrypted by the client and decrypted by the server.

• Initialization vectors: When a block cipher in CBC mode is used, an initialization vector (IV) is maintained for each key. This field is first initialized by the SSL Handshake Protocol. Thereafter, the final ciphertext block from each record is preserved for use as the IV with the following record.

• Sequence numbers: Each party maintains separate sequence numbers for transmitted and received messages for each connection. When a party sends or receives a change cipher spec message, the appropriate sequence number is set to zero. Sequence numbers may not exceed $2^{64} - 1$.



SSL Record Protocol defines two services for SSL connections:

• Confidentiality: The Handshake Protocol defines a shared secret key that is used for conventional encryption of SSL payloads. The message is compressed before being concatenated with the MAC and encrypted, with a range of ciphers being supported as shown.

• Message Integrity: The Handshake Protocol also defines a shared secret key that is used to form a message authentication code (MAC), which is similar to HMAC

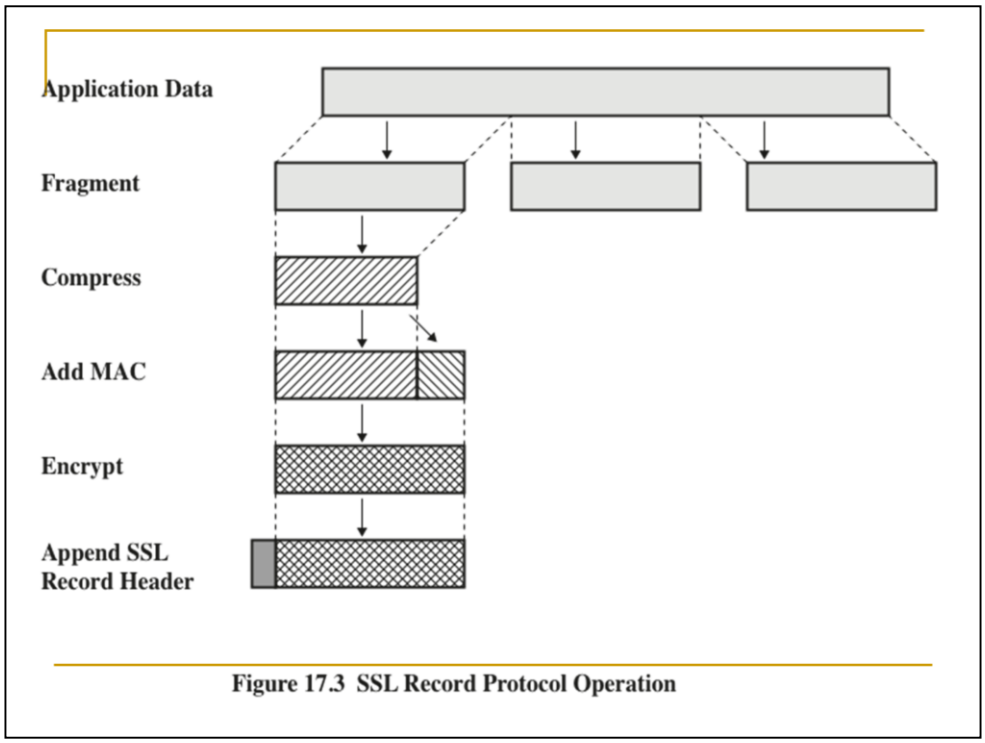


Figure 17.3 SSL Record Protocol Operation

Figure 17.3 indicates the overall operation of the SSL Record Protocol. The

Record Protocol takes an application message to be transmitted, fragments the data

into manageable blocks, optionally compresses the data, applies a MAC, encrypts,

adds a header, and transmits the resulting unit in a TCP segment. Received data

are decrypted, verified, decompressed, and reassembled before being delivered to

higher-level users.

The first step is fragmentation. Each upper-layer message is fragmented into

blocks of 2^{14} bytes (16384 bytes) or less. Next, compression is optionally applied.

Compression must be lossless and may not increase the content length by more than

1024 bytes. In SSLv3 (as well as the current version of TLS), no

compression algorithm
is specified, so the default compression algorithm is null.

The next step in processing is to compute a message authentication code over the compressed data. For this purpose, a shared secret key is used.

Next, the compressed message plus the MAC are 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.

For stream encryption, the compressed message plus the MAC are encrypted. Note that the MAC is computed before encryption takes place and that the MAC is
then encrypted along with the plaintext or compressed plaintext.

For block encryption, padding may be added after the MAC prior to encryption. The padding is in the form of a number of padding bytes followed by a onebyte indication of the length of the padding. The total amount of padding is the smallest amount such that the total size of the data to be encrypted (plaintext plus
MAC plus padding) is a multiple of the cipher's block length. An example is a plaintext
(or compressed text if compression is used) of 58 bytes, with a MAC of 20 bytes (using SHA-1), that is encrypted using a block length of 8 bytes (e.g., DES). With the padding-length byte, this yields a total of 79 bytes. To make the total an integer
multiple of 8, one byte of padding is added.

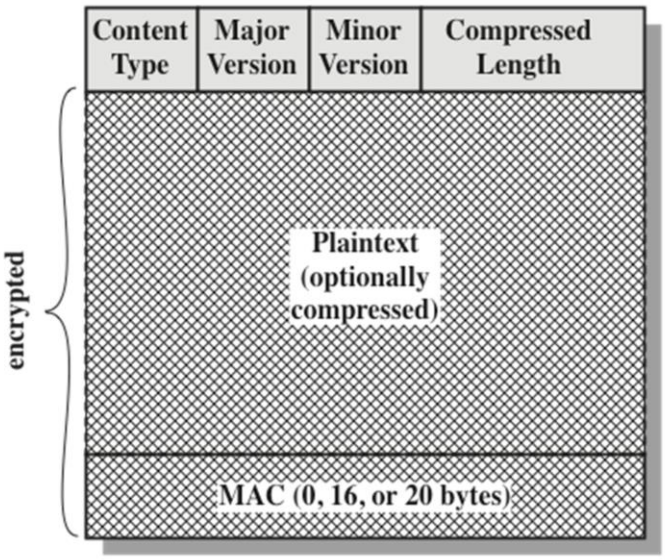


Figure 17.4 SSL Record Format

The final step of SSL Record Protocol processing is to prepare a header consisting of the following fields:

• 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 plaintext fragment (or compressed fragment if compression is used). The maximum value is 2^{14} + 2048.

The content types that have been defined are change_cipher_spec , alert , handshake , and application_data . The first three are the SSL-specific protocols, discussed next. Note that no distinction is made among the various applications
(e.g., HTTP) that might use SSL; the content of the data created by such applications is opaque to SSL.

Figure 17.4 illustrates the SSL record format.

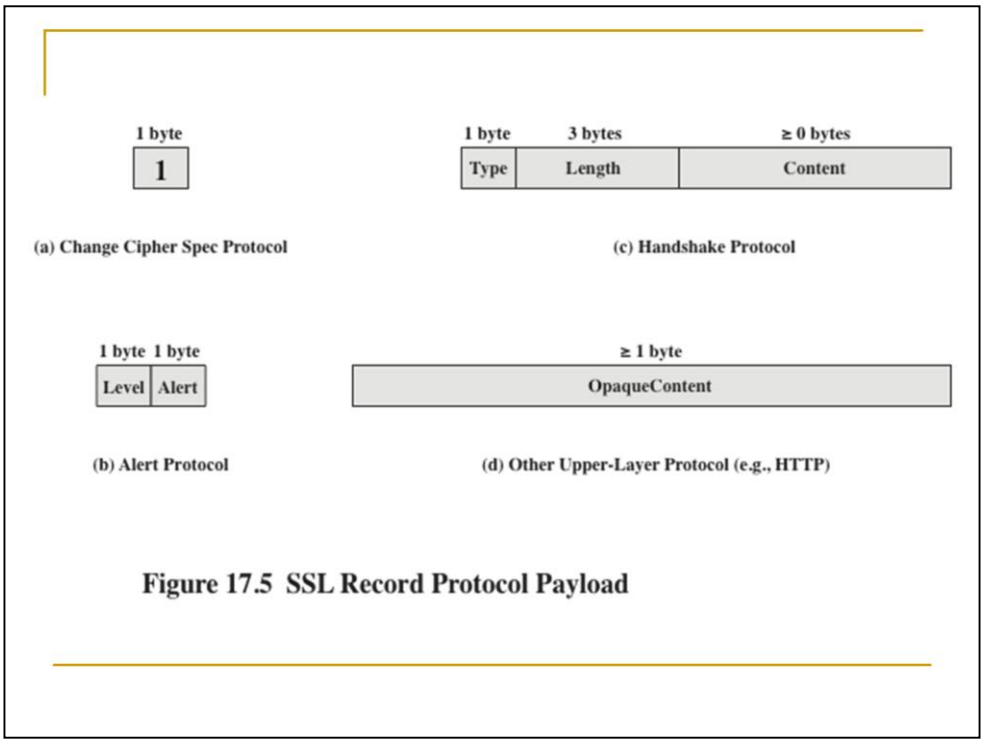


Figure 17.5 SSL Record Protocol Payload

The Change Cipher Spec Protocol is one of the three SSL-specific protocols that use the SSL Record Protocol, and it is the simplest. This protocol consists of a single message (Figure 17.5a), which consists of a single byte with the value 1. The sole purpose of this message is to cause the pending state to be copied into the current state, which updates the cipher suite to be used on this connection.

The Alert Protocol is used to convey SSL-related alerts to the peer entity. As with other applications that use SSL, alert messages are compressed and encrypted, as specified by the current state.

Each message in this protocol consists of two bytes (Figure 17.5b). The first byte takes the value warning (1) or fatal (2) to convey the severity of the

message.

If the level is fatal, SSL immediately terminates the connection. Other connections on the same session may continue, but no new connections on this session may be established. The second byte contains a code that indicates the specific alert.

The most complex part of SSL is the 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. All of these have the format shown in Figure 17.5c.

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

Table 17.2 SSL Handshake Protocol Message Types

Each message has

three fields:

• Type (1 byte): Indicates one of ten messages. Table 17.2 lists the defined

message types.

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

• Content (# 0 bytes): The parameters associated with this message; these are

listed in Table 17.2.

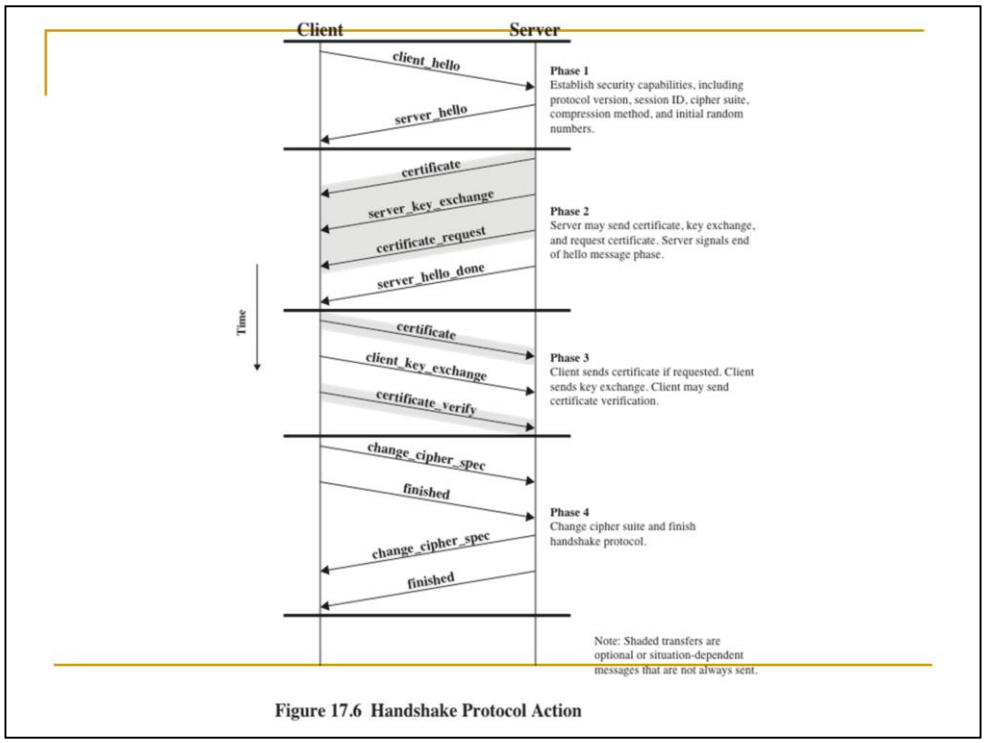
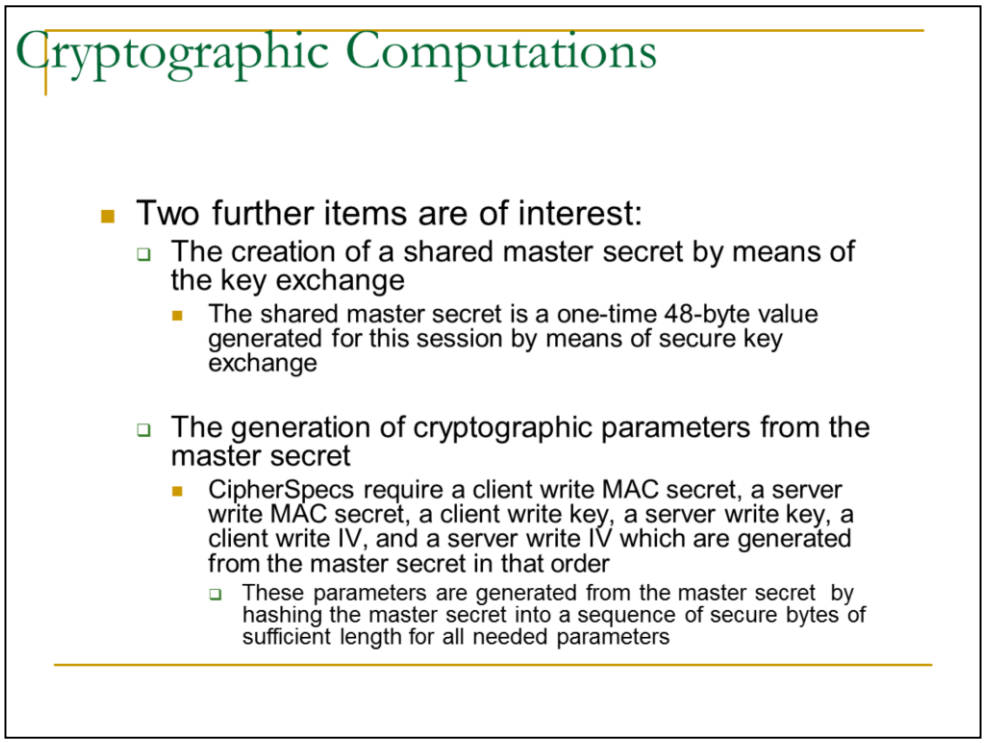


Figure 17.6 Handshake Protocol Action

Figure 17.6 shows the initial exchange needed to establish a logical connection

between client and server. The exchange can be viewed as having four phases.



Two further items are of interest: (1) the creation of a shared master secret by means

of the key exchange and (2) the generation of cryptographic parameters from the

master secret.

The shared master secret is a one-time 48-byte value

(384 bits) generated for this session by means of secure key exchange. The creation

is in two stages. First, a pre_master_secret is exchanged. Second, the master_

secret is calculated by both parties. For pre_master_secret exchange, there

are two possibilities.

• RSA: A 48-byte pre_master_secret is generated by the client, encrypted

with the server's public RSA key, and sent to the server. The server decrypts

the ciphertext using its private key to recover the pre_master_secret.

• Diffie-Hellman: Both client and server generate a Diffie-Hellman public key. After these are exchanged, each side performs

Summary

- Web Security Considerations
- SSL
  - Other topics in the book (Not examinable)
    - Transport Layer Security
    - HTTPS
    - SSH