22.3 SECURE SOCKETS LAYER (SSL) AND TRANSPORT LAYER SECURITY (TLS)

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 2246. 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 discusses SSLv3. Only minor changes are found in TLS.

SSL Architecture

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 22.4.

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.

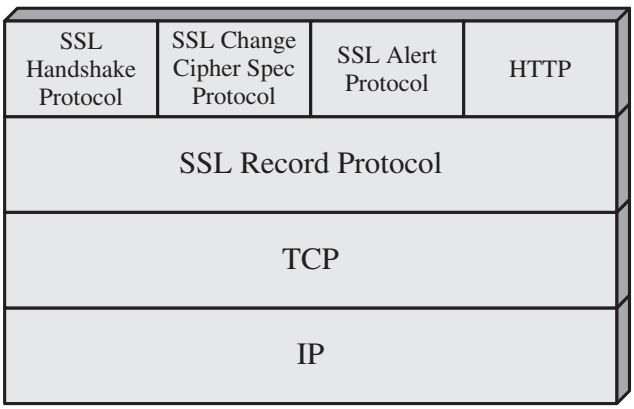

Figure 22.4 SSL Protocol Stack

- **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.

SSL Record Protocol

The SSL Record Protocol provides two services for SSL connections:

- **Confidentiality**: The Handshake Protocol defines a shared secret key that is used for symmetric encryption of SSL payloads.
- **Message integrity**: The Handshake Protocol also defines a shared secret key that is used to form a message authentication code (MAC).

Figure 22.5 indicates the overall operation of the SSL Record Protocol. The first step is **fragmentation**. Each upper-layer message is fragmented into blocks of 214 bytes (16,384 bytes) or less. Next, **compression** is optionally applied. The next step in processing is to compute a **message authentication code** over the compressed data. Next, the compressed message plus the MAC are **encrypted** using symmetric encryption.

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

- **Content Type (8 bits)**: The higher-layer protocol used to process the enclosed fragment.

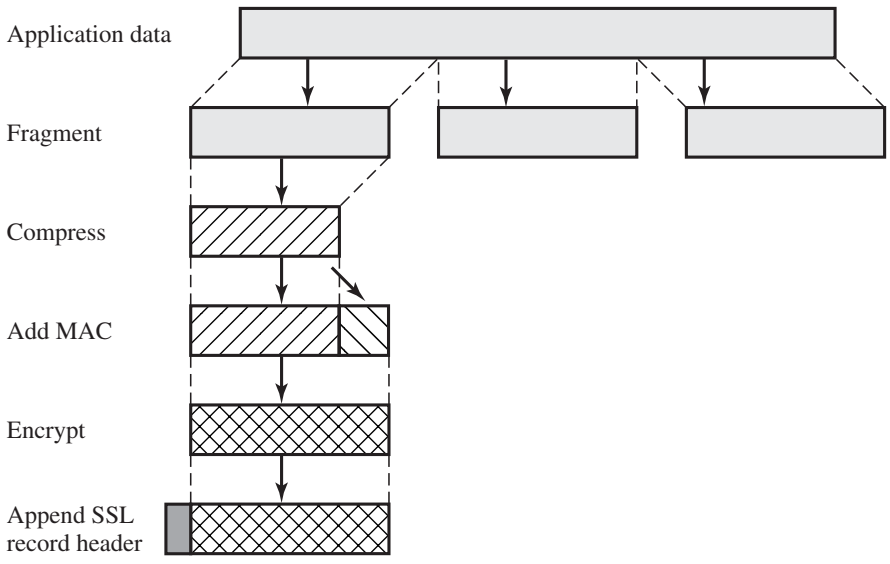

Figure 22.5 SSL Record Protocol Operation

- **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.

The Record Protocol then transmits the resulting unit in a TCP segment. Received data are decrypted, verified, decompressed, and reassembled, and then delivered to higher-level users.

Change Cipher Spec Protocol

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, 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.

Alert Protocol

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. 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. An example of a fatal alert is an incorrect MAC. An example of a nonfatal alert is a close_notify message, which notifies the recipient that the sender will not send any more messages on this connection.

Handshake Protocol

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 are transmitted.

The Handshake Protocol consists of a series of messages exchanged by client and server. Figure 22.6 shows the initial exchange needed to establish a logical connection between client and server. The exchange can be viewed as having four phases.

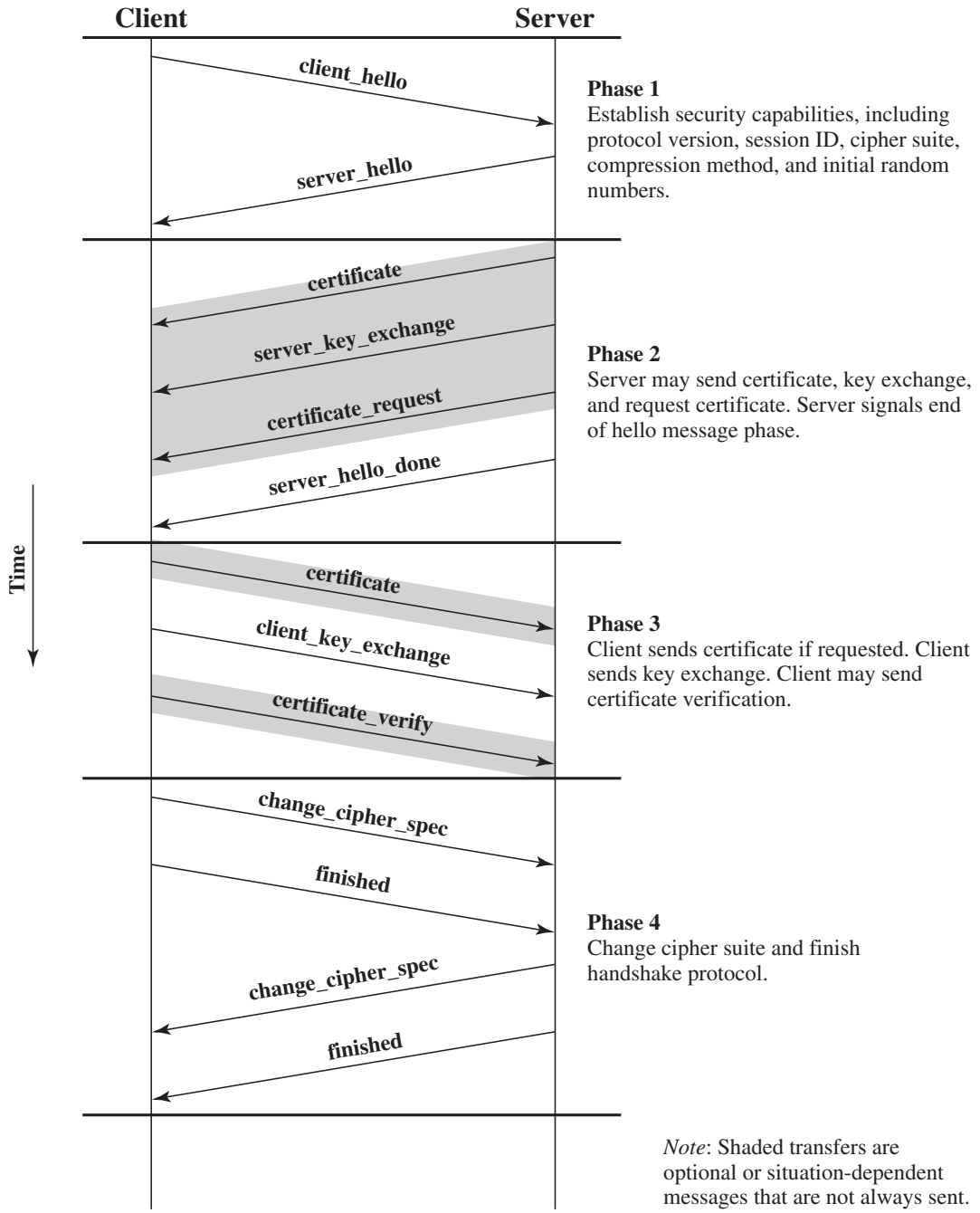

Figure 22.6 **Handshake Protocol Action**

Phase 1 is used to initiate a logical connection and to establish the security capabilities that will be associated with it. The exchange is initiated by the client, which sends a client_hello message 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 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 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.
- **Compression method**: This is a list of the compression methods the client supports.

After sending the client_hello message, the client waits for the server_hello message, which contains the same parameters as the client_hello message.

The details of **phase 2** depend on the underlying public-key encryption scheme that is used. In some cases, the server passes a certificate to the client, possibly additional key information, and a request for a certificate from the client.

The final message in phase 2, and one that is always required, is the server_done message, which is sent by the server to indicate the end of the server hello and associated messages. After sending this message, the server will wait for a client response.

In **phase 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, depending on the underlying public-key scheme.

Phase 4 completes the setting up of a secure connection. The client sends a change_cipher_spec message and copies the pending CipherSpec into the current CipherSpec. Note that this message is not considered part of the Handshake Protocol but is sent using the Change Cipher Spec Protocol. The client then immediately sends the finished message under the new algorithms, keys, and secrets. The finished message verifies that the key exchange and authentication processes were successful.

In response to these two messages, the server sends its own change_cipher_spec message, transfers the pending to the current CipherSpec, and sends its finished message. At this point, the handshake is complete and the client and server may begin to exchange application layer data.

22.4 HTTPS

HTTPS (HTTP over SSL) refers to the combination of HTTP and SSL to implement secure communication between a Web browser and a Web server. The HTTPS capability is built into all modern Web browsers. Its use depends on the Web server supporting HTTPS communication. For example, search engines do not support HTTPS.

The principal difference seen by a user of a Web browser is that URL (uniform resource locator) addresses begin with https:// rather than http://. A normal HTTP connection uses port 80. If HTTPS is specified, port 443 is used, which invokes SSL.