SSL/TLS for Secure Web Services

- SSL (Secure Socket Layer) was developed originally by Netscape to provide secure and authenticated connections between browsers and servers in 1995.

- SSL provides **transport layer** security. Recall from Slide 18 that the transport layer is where the TCP and UDP protocols reside in the TCP/IP stack. [Since SSL sits immediately above TCP in the protocol stack, a more precise way of stating this would be that SSL provides **Session Layer** security in the OSI model of the internet protocols. See the comments on Slide 18.]

- IETF (Internet Engineering Task Force, the body in charge of the core internet protocols, including the TCP/IP protocol) made **SSL Version 3** an open standard in 1999 and called it **TLS** (Transport Layer Security) **Version 1**.

- SSL/TLS allows for either server-only authentication or server-client authentication. In server-only authentication, the client receives the server's certificate. The client verifies the server's certificate and generates a secret key that it then encrypts with the server's public key. The client sends the encrypted secret key to the server; the server decrypts it with its own private key and

subsequently uses the client-generated secret key to encrypt the messages meant for the client.

- In the server-client authentication, in addition to the secret key, the client also sends to the server its certificate that the server uses for authenticating the client. **OpenSSL is an implementation of the SSL and the TLS protocols.**

- SSL is actually not a single protocol, or even a single protocol layer. SSL is composed of **four protocols** in **two layers**, as shown in the figure.

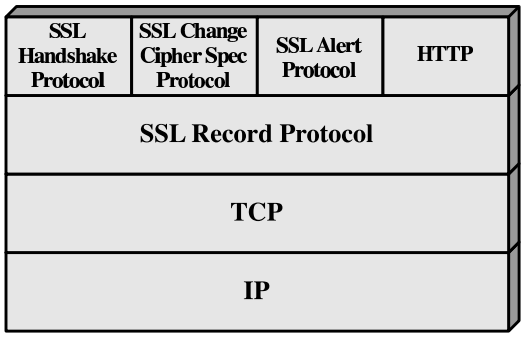

Figure 17.2 SSL Protocol Stack

This figure is from Chapter 17 of Stallings: "Cryptography and Network Security", Fourth Edition

The Twin Concepts of "SSL Connection" and "SSL Session"

- In the SSL family of protocols, a **connection** is a **one-time transport** of information between two nodes in a communication network.
  - A connection constitutes a **peer-to-peer** relationship between the two nodes.
  - Being one-time, connections are transient.
  - Every connection is associated with a **session**.
- A **session** is an enduring association between a **client** and a **server**.
  - A session is created by the **SSL Handshaking Protocol**.
  - A session can consist of multiple connections.
  - A session is characterized by a set of security parameters that apply to all the connections in the session.
  - The concept of a session eliminates the need for negotiating the security parameters for each separate connection.

SSL Session State

An SSL **session state** is characterized 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 the data prior to encryption.
- **Cipher Spec:** Specifics of the bulk data encryption algorithm and the hash algorithm used for MAC calculations.
- **Master Secret:** A 48-byte secret shared between the client and the server.
- **IsResumable:** A flag indicating whether the session is allowed to initiate new connections.

SSL Connection State

An SSL **connection state** is characterized by the following **parameters**:

- **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 symmetric-key encryption key for data encrypted by the server and decrypted by the client.

- **Client Write Key:** The symmetric-key encryption key for data encrypted by the client and decrypted by the server.

- **Initialization vectors:** An initialization vector (IV) for each key used by a block cipher operating in the CBC mode is maintained. The vectors are initialized by the **SSL Handshake Protocol**. Subsequently, the final ciphertext block from each record is preserved for use as the IV with the following record. (*This will become clearer after we have discussed the SSL Record Protocol.*)

- **Sequence Numbers:** Each party maintains separate sequence numbers for the transmitted and received messages through 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

- As shown by the figure on Slide 38, the **SSL Record Protocol** sits directly above the TCP protocol.

- This protocol provides two services: **Confidentiality** and **Message Integrity**.

- In a nutshell, this protocol is in charge of taking the actual data that the server wants to send to a client or that the client wants to send to a server, fragmenting the data into blocks, applying authentication and encryption primitives to each block, and handing the block to TCP for transmission over the network. On the receive side, the blocks are decrypted, verified for message integrity, reassembled, and delivered to the higher-level protocol.

- The operation of the **SSL Record Protocol** is shown in the figure on Slide 45. It consists of the following **five** steps:

  - **Fragmentation:** The message (either from server to client, or from client to server) is fragmented into blocks whose length does not exceed 2^{14} (16384) bytes.

- **Compression:** This optional step requires lossless compression and carries the stipulation that the size of the input block will not increase by more than 1024 bytes. [As you'd expect, compression will, in most cases, reduce the length of a block produced by the fragmentation step. But for very short blocks, the length may increase.] SSLv3, the current version of SSL, does not specify compression.

- **Adding MAC:** This step computes the MAC for the block. The MAC is appended to the compressed message block.

- **Encryption:** The compressed message and the MAC are encrypted using symmetric-key encryption. The encryption may be carried out with a block cipher such as 3DES or with a stream cipher such as RC4-128. A number of choices are available for the encryption step depending on the level of security needed.

- **Append SSL Record Header:** Finally, an SSL header is is **prepended** to the encrypted block. The header consists of 8 bits for declaring the content type, 8 bits for declaring the major version used for SSL, 8 bits for declaring the minor version used, and 16 bits for declaring the length of the compressed plaintext (or the plaintext if no compression was used).

- Each output block produced by the **SSL Record Protocol** is referred to as an **SSL record**. The length of a record is not to exceed $32,767$ bytes.

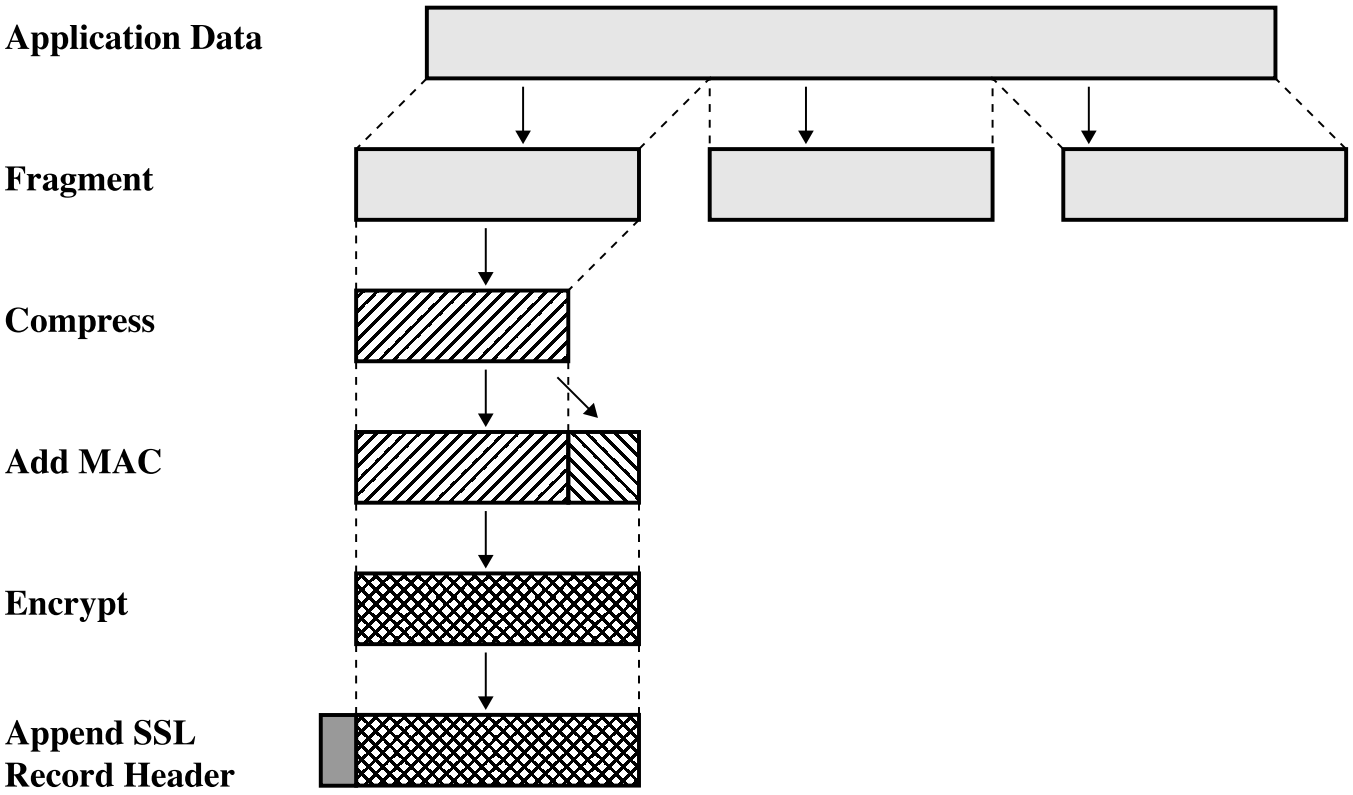

Figure 17.3 SSL Record Protocol Operation

This figure is from Chapter 17 of Stallings: "Cryptography and Network Security", Fourth Edition

SSL Handshake Protocol

- Before the **SSL Record Protocol** can do its thing, it must become aware of what algorithms to use for compression, authentication, and encryption. All of that information is generated by the **SSL Handshake Protocol**.

- The **SSL Handshake Protocol** is also responsible for the server and the client to authenticate each other.

- This protocol must also come up with the cryptographic keys to be used for the encryption and the authentication of each SSL record.

- As shown by the figure on Slide 50, the **SSL Handshake** protocol works in **four phases**.

- **Phase 1** handshaking, **initiated by the client**, is used to establish the **security capabilities** present at the two ends of a connection. The client sends to the server a **client_hello message** with the following parameters:
  - **Version** (the highest SSL version understood by the client)
  - **Random** (a 32-bit timestamp and a 28-byte random field

that together serve as **nonces** during key exchange to prevent replay attacks)

- **Session ID** (a variable length session identifier);
- **Cipher Suite** (a list of cryptographic algorithms supported by the client, in decreasing order of preference); and
- **Compression Method** (a list of compression methods the client supports).

The server responds with its **server_hello message** that has a similar set of parameters. Server's response, as you'd expect, includes the specific algorithms selected by the server from the client's lists for compression, authentication, and encryption.

- The **Cipher Suite** parameter in the **server hello message** consists of two elements. The first element declares the **key exchange method** selected. (The choice is between **RSA**, three different types of **Diffie-Hellman**, etc.) The second element of the **Cipher Suite** parameter is called **CipherSpec**; it has a number of fields that indicate the authentication algorithm selected, the length of MAC, the encryption algorithm, etc.

- **Phase 2** handshaking is initiated by the server if server authentication is needed by the client. The server sends to the client a **certificate message** containing its one or more certificates validating its public key. This could be followed by a **server_key_exchange message** and a **certificate_request message**, both from the server to the client. The **server_key**

exchange message could, for example, consist of the global Diffie-Hellman values (a prime number and a primitive root of that number) and the server's Diffie-Hellman public key. **Phase 2** handshaking ends when the server sends the client a **server hello done message**.

- **Phase 3** handshaking is initiating by the client. If client authentication is required, the client sends to the server the **certificate message** containing one or more certificates validating its public key. Next, the client sends to the server a mandatory **client_key_exchange message** that could, for example, consist of a secret session key encrypted with the server's public key. This phase ends when the client sends to the server a **certificate verify message** to provide a verification of its certificates if they are signed by a certificate authority.

- **Phase 4** handshaking completes the setting up of a secure connection between the client and the server. The client sends to the server a **change_cipher_spec message** indicating that it is copying the pending CipherSpec into the current CipherSpec. (See Phase 1 handshaking for CipherSpec.) Next, the client sends to the server the **finished** message. As shown in the figure, the server does the same vis-a-vis the client.

- The **change_cipher_spec message** format must correspond to the **Change Cipher Spec Protocol**. This protocol says

that the message must consist of a single byte with a value of 1 indicating the change.

- The last of the SSL protocols, **Alert Protocol**, is used to convey SSL-related alerts to the peer entity.

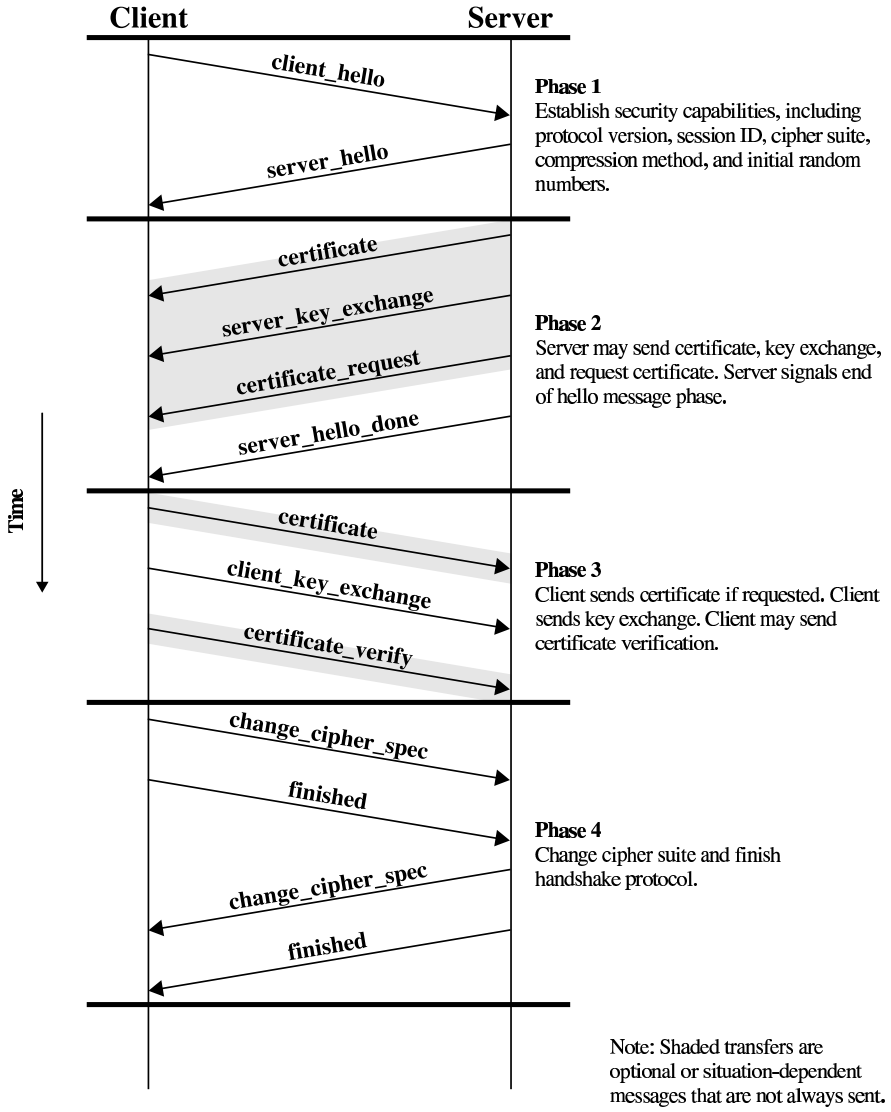

Figure 17.6 Handshake Protocol Action

This figure is from Chapter 17 of Stallings: "Cryptography and Network Security", Fourth Edition