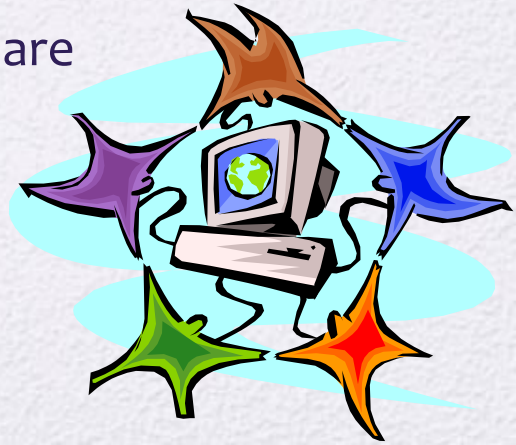
# Web Security Considerations

- The World Wide Web is fundamentally a client/server application running over the Internet and TCP/IP intranets
- The following characteristics of Web usage suggest the need for tailored security tools:
  - Web servers are relatively easy to configure and manage
  - Web content is increasingly easy to develop
  - The underlying software is extraordinarily complex
    - May hide many potential security flaws
  - A Web server can be exploited as a launching pad into the corporation's or agency's entire computer complex
  - 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<br>•Trojan horse browser<br>•Modification of memory<br>•Modification of message traffic in transit | •Loss of information<br>•Compromise of machine<br>•Vulnerabilty to all other threats | Cryptographic checksums |
| **Confidentiality** | •Eavesdropping on the net<br>•Theft of info from server<br>•Theft of data from client<br>•Info about network configuration<br>•Info about which client talks to server | •Loss of information<br>•Loss of privacy | Encryption, Web proxies |
| **Denial of Service** | •Killing of user threads<br>•Flooding machine with bogus requests<br>•Filling up disk or memory<br>•Isolating machine by DNS attacks | •Disruptive<br>•Annoying<br>•Prevent user from getting work done | Difficult to prevent |
| **Authentication** | •Impersonation of legitimate users<br>•Data forgery | •Misrepresentation of user<br>•Belief that false information is valid | Cryptographic techniques |

Table 17.1 A Comparison of Threats on the Web

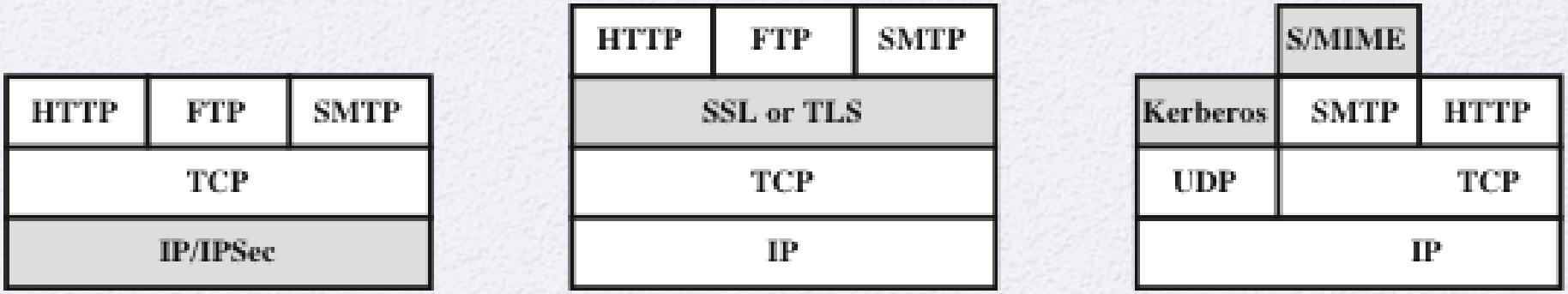



**Figure 17.1 Relative Location of Security Facilities in the TCP/IP Protocol Stack**

# Transport Layer Security (TLS)

One of the most widely used security services

Defined in RFC 5246

Is an Internet standard that evolved from a commercial protocol known as Secure Sockets Layer (SSL)

Is a general purpose service implemented as a set of protocols that rely on TCP

Could be provided as part of the underlying protocol suite and therefore be transparent to applications

Can be embedded in specific packages

Most browsers come equipped with TLS, and most Web servers have implemented the protocol

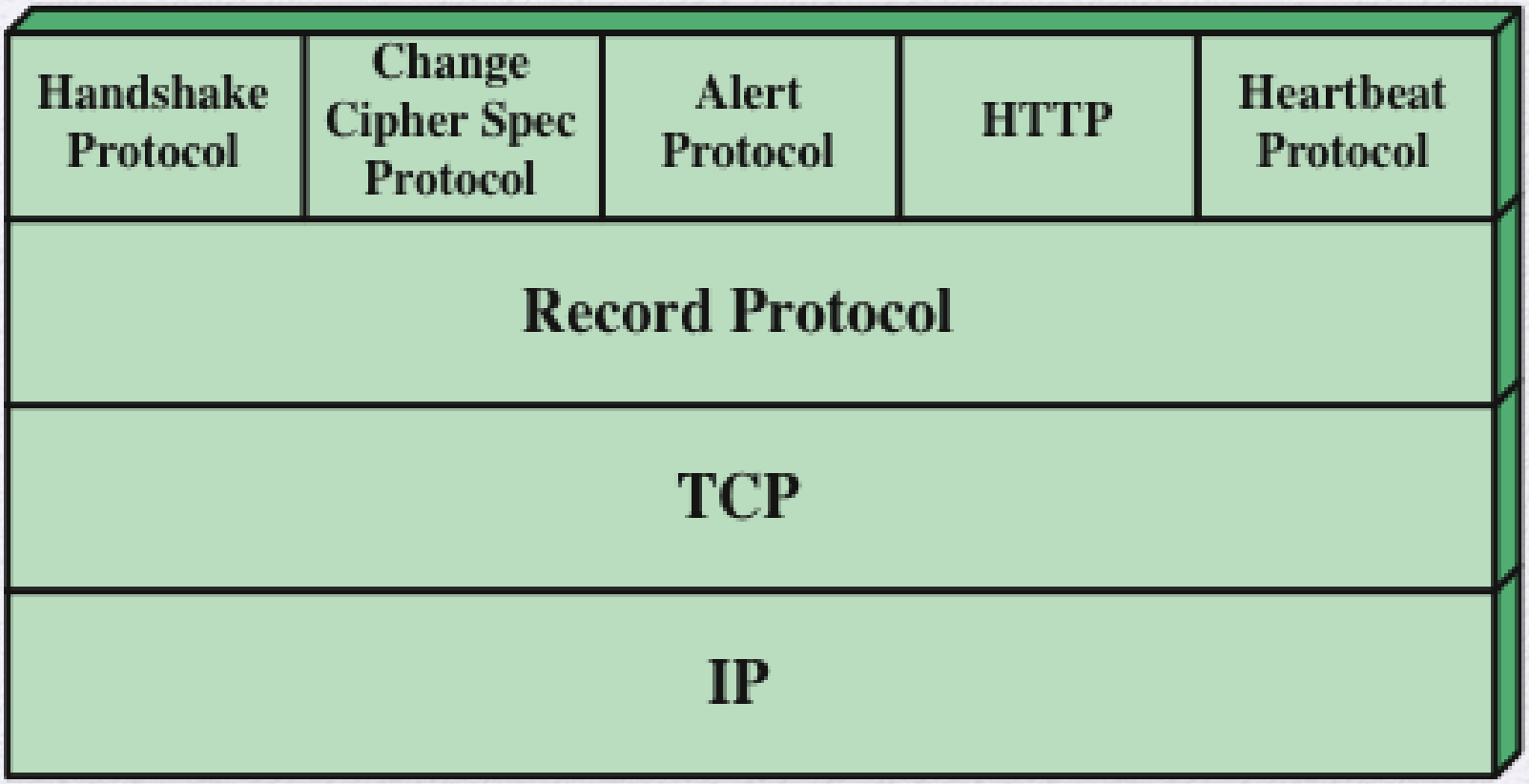



**Figure 17.2 SSL/TLS Protocol Stack**

# TLS Architecture

- Two important TLS concepts are:

**TLS connection**

- A transport that provides a suitable type of service
- For TLS such connections are peer-to-peer relationships
- Connections are transient
- Every connection is associated with one session

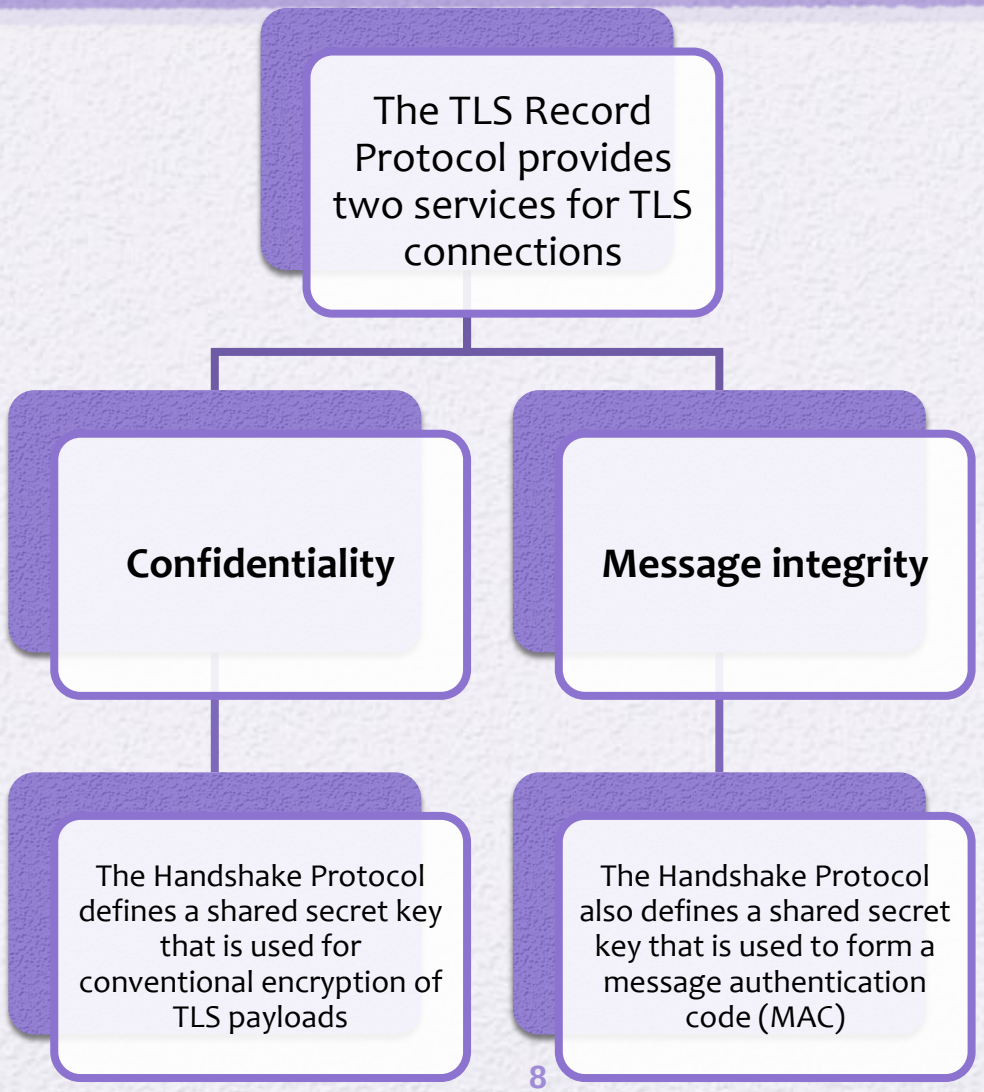
**TLS session**

- An association between a client and a server
- Created by the Handshake Protocol
- Define a set of cryptographic security parameters which can be shared among multiple connections
- Are used to avoid the expensive negotiation of new security parameters for each connection

# A session state is defined by the following parameters:

| Session identifier | Peer certificate | Compression method | Cipher spec | Master secret | Is resumable |
|---|---|---|---|---|---|
| An arbitrary byte sequence chosen by the server to identify an active or resumable session state | An X509.v3 certificate of the peer; this element of the state may be null | The algorithm used to compress data prior to encryption | Specifies the bulk data encryption algorithm and a hash algorithm used for MAC calculation; also defines cryptographic attributes such as the hash_size | 48-byte secret shared between the client and the server | A flag indicating whether the session can be used to initiate new connections |

# TLS Record Protocol



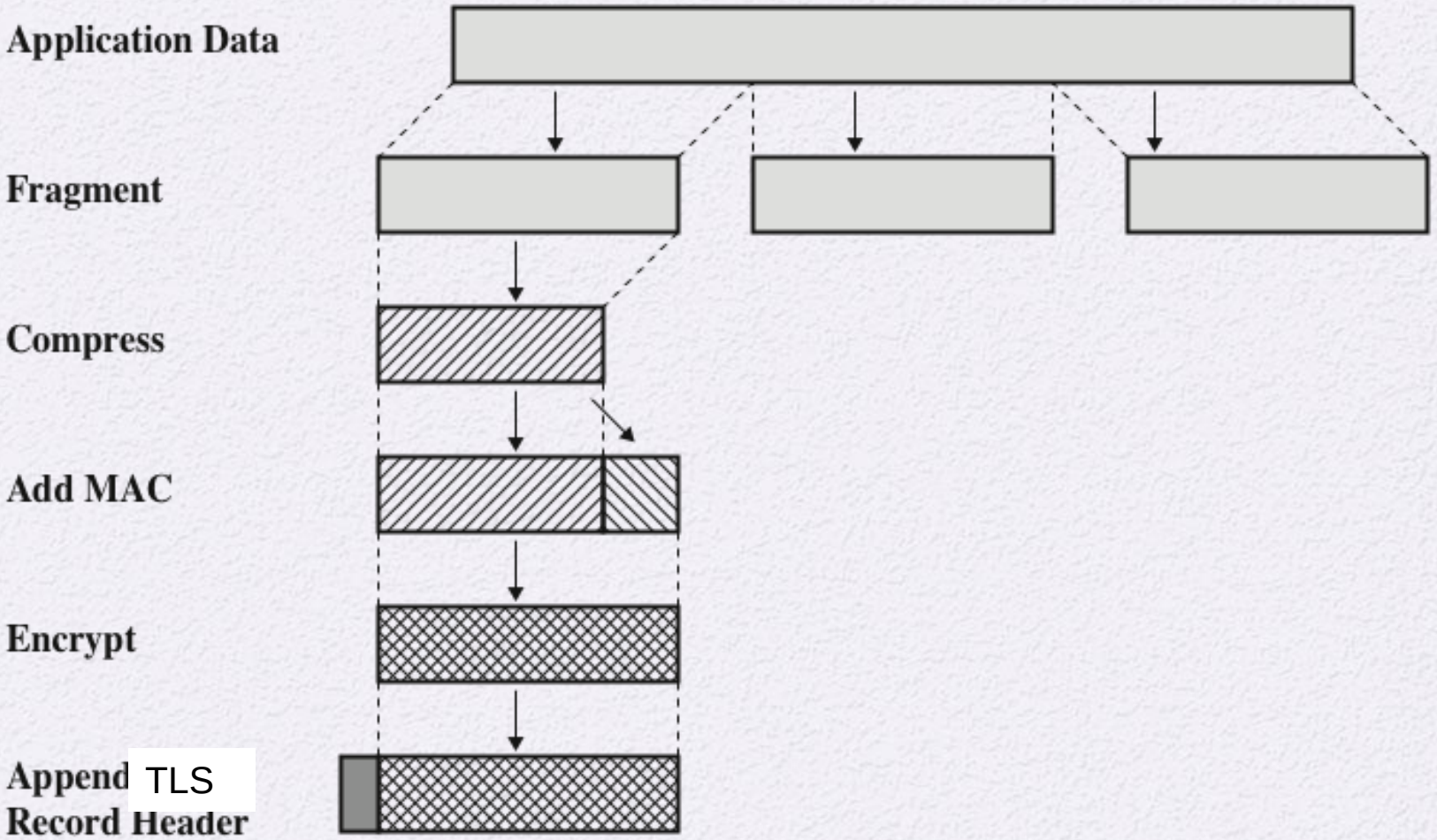



Figure 17.3 TLS Record Protocol Operation

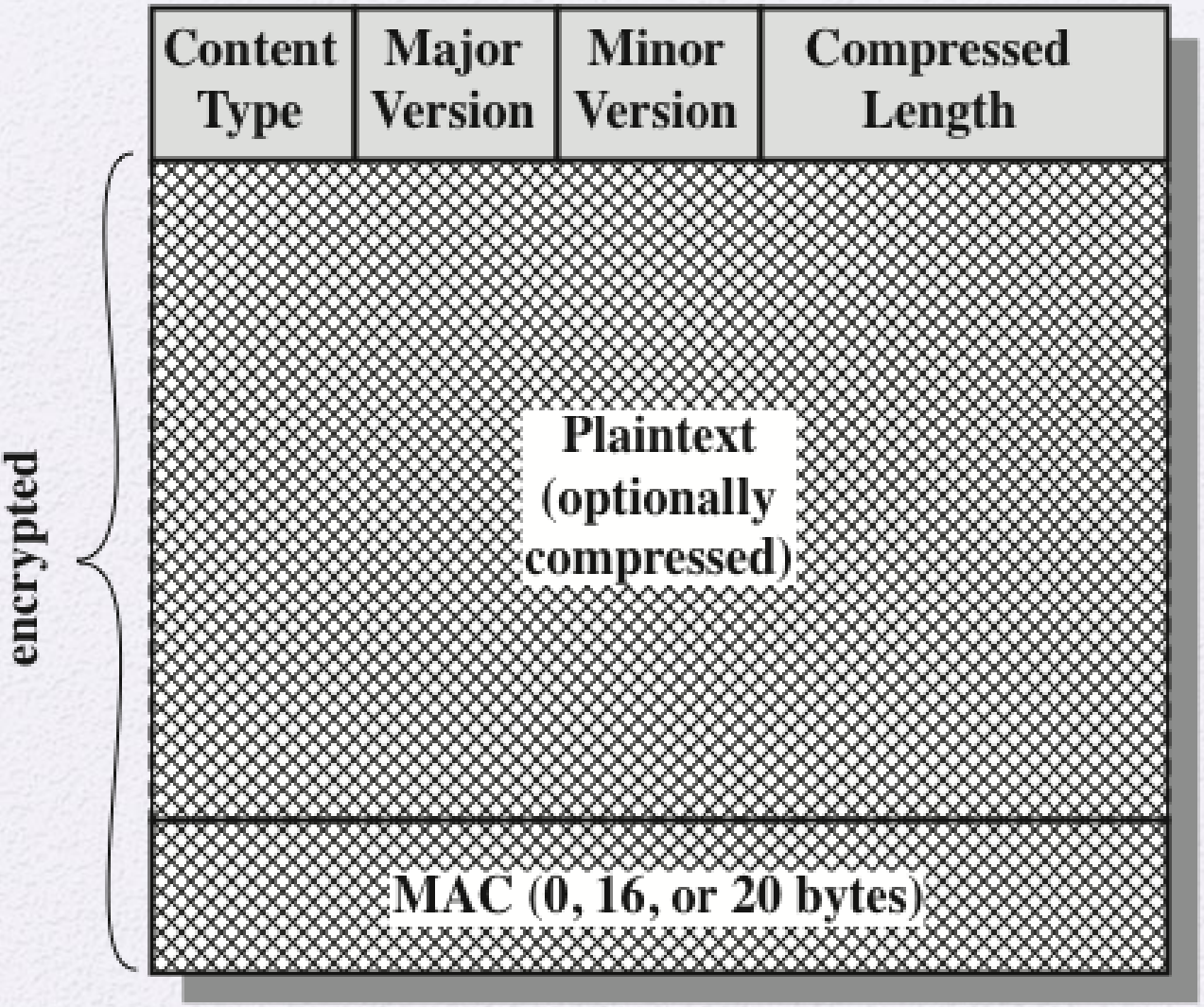
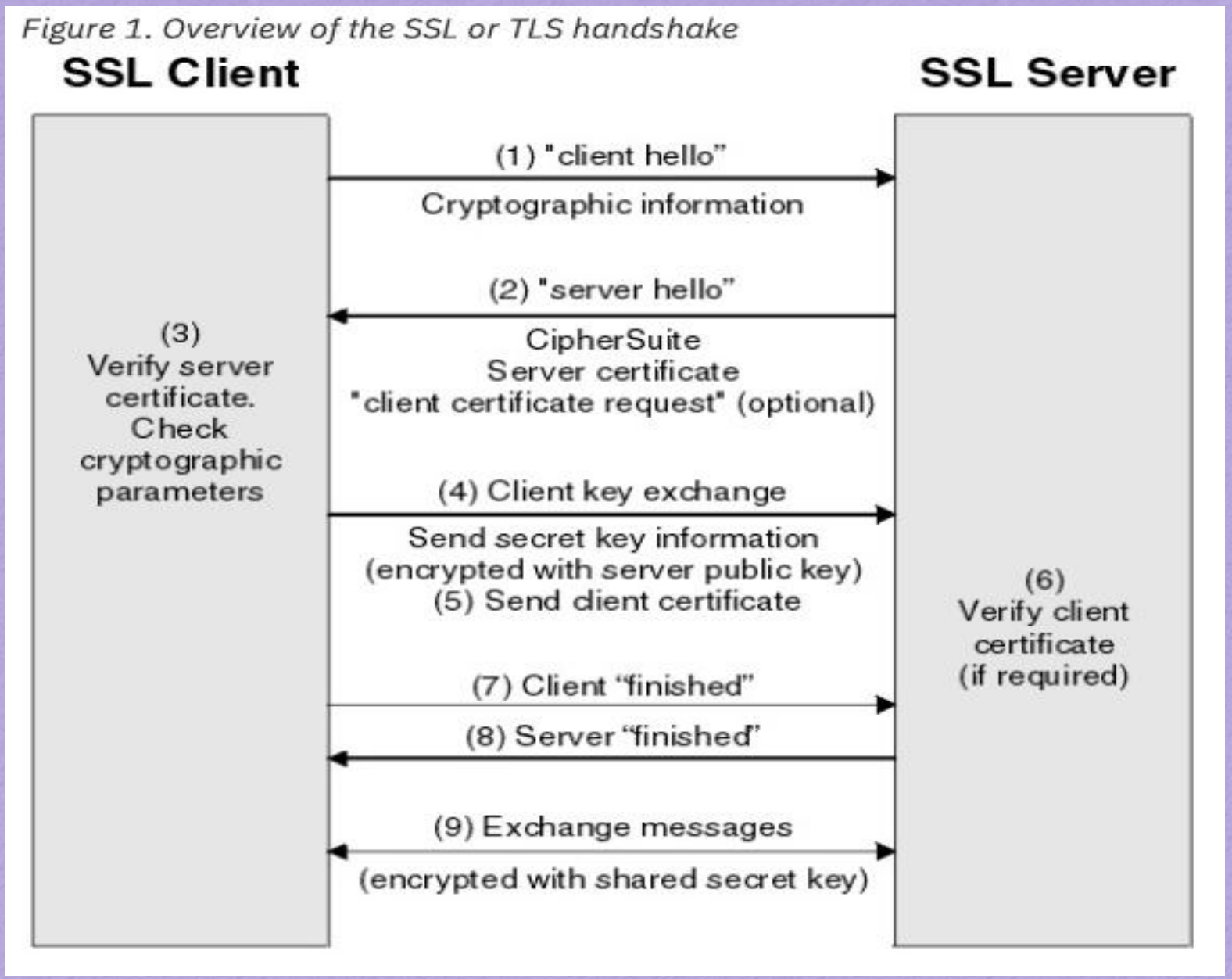



**Figure 17.4 TLS Record Format**

# SSL Handshake Protocol

- Allows server & client to:
  - authenticate each other
  - to negotiate encryption & MAC algorithms and keys
- Comprises a series of messages exchanged in phases:
  1. Establish Security Capabilities (to agree on encryption, MAC, and key-exchange algorithms)
  2. Server Authentication and Key Exchange
  3. Client Authentication and Key Exchange
  4. Finish

*Figure 1. Overview of the SSL or TLS handshake*



1. Client Hello

Information that the server needs to communicate with the client using SSL. This includes the SSL version number, cipher settings, session-specific data.

2. Server Hello

Information that the server needs to communicate with the client using SSL. This includes the SSL version number, cipher settings, session-specific data.

3. Authentication and Pre-Master Secret

Client authenticates the server certificate. (e.g. Common Name / Date / Issuer) Client (depending on the cipher) creates the pre-master secret for the session, Encrypts with the server's public key and sends the encrypted pre-master secret to the server.

4. Decryption and Master Secret

Server uses its private key to decrypt the pre-master secret. Both Server and Client perform steps to generate the master secret with the agreed cipher.

5. Encryption with Session Key

Both client and server exchange messages to inform that future messages will be encrypted.

# SSL/TLS Attacks

- The attacks can be grouped into four general categories:
  - Attacks on the handshake protocol
  - Attacks on the record and application data protocols
  - Attacks on the PKI
  - Other attacks
- The constant back-and-forth between threats and countermeasures determines the evolution of Internet-based protocols

# TLSv1.3

- Primary aim is to improve the security of TLS
- Significant changes from version 1.2 are:
  - TLSv1.3 removes support for a number of options and functions
    - Deleted items include:
      - Compression
      - Ciphers that do not offer authenticated encryption
      - Static RSA and DH key exchange
      - 32-bit timestamp as part of the Random parameter in the client_hello message
      - Renegotiation
      - Change Cipher Spec Protocol
      - RC4
      - Use of MD5 and SHA-224 hashes with signatures
  - TLSv1.3 uses Diffie-Hellman or Elleptic Curve Diffie-Hellman for key exchange and does not permit RSA
  - TLSv1.3 allows for a "1 round trip time" handshake by changing the order of message sent with establishing a secure connection

# 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
- A user of a Web browser will see URL addresses that begin with https:// rather than http://
- If HTTPS is specified, port 443 is used, which invokes SSL
- Documented in RFC 2818, *HTTP Over TLS*
  - There is no fundamental change in using HTTP over either SSL or TLS and both implementations are referred to as HTTPS
- When HTTPS is used, the following elements of the communication are encrypted:
  - URL of the requested document
  - Contents of the document
  - Contents of browser forms
  - Cookies sent from browser to server and from server to browser
  - Contents of HTTP header

# Connection Initiation

## For HTTPS, the agent acting as the HTTP client also acts as the TLS client

The client initiates a connection to the server on the appropriate port and then sends the TLS ClientHello to begin the TLS handshake

When the TLS handshake has finished, the client may then initiate the first HTTP request

All HTTP data is to be sent as TLS application data

## There are three levels of awareness of a connection in HTTPS:

At the HTTP level, an HTTP client requests a connection to an HTTP server by sending a connection request to the next lowest layer

- Typically the next lowest layer is TCP, but is may also be TLS/SSL

At the level of TLS, a session is established between a TLS client and a TLS server

- This session can support one or more connections at any time

A TLS request to establish a connection begins with the establishment of a TCP connection between the TCP entity on the client side and the TCP entity on the server side

# Connection Closure

- An HTTP client or server can indicate the closing of a connection by including the line `Connection: close` in an HTTP record
- The closure of an HTTPS connection requires that TLS close the connection with the peer TLS entity on the remote side, which will involve closing the underlying TCP connection
- TLS implementations must initiate an exchange of closure alerts before closing a connection
  - A TLS implementation may, after sending a closure alert, close the connection without waiting for the peer to send its closure alert, generating an "incomplete close"
- An unannounced TCP closure could be evidence of some sort of attack so the HTTPS client should issue some sort of security warning when this occurs