CHAPTER 5

Transport-Level Security

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 •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 6.1 A Comparison of Threats on the Web

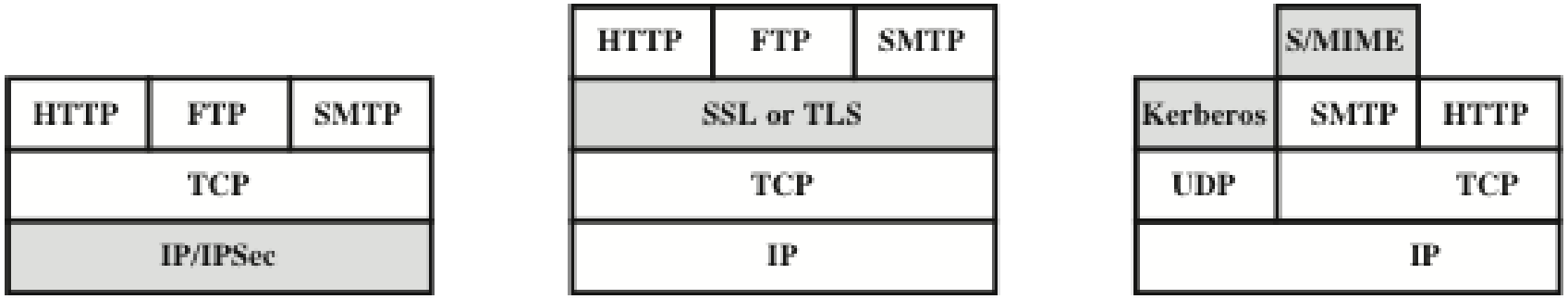


Figure 6.1 Relative Location of Security Facilities in the TCP/IP Protocol Stack

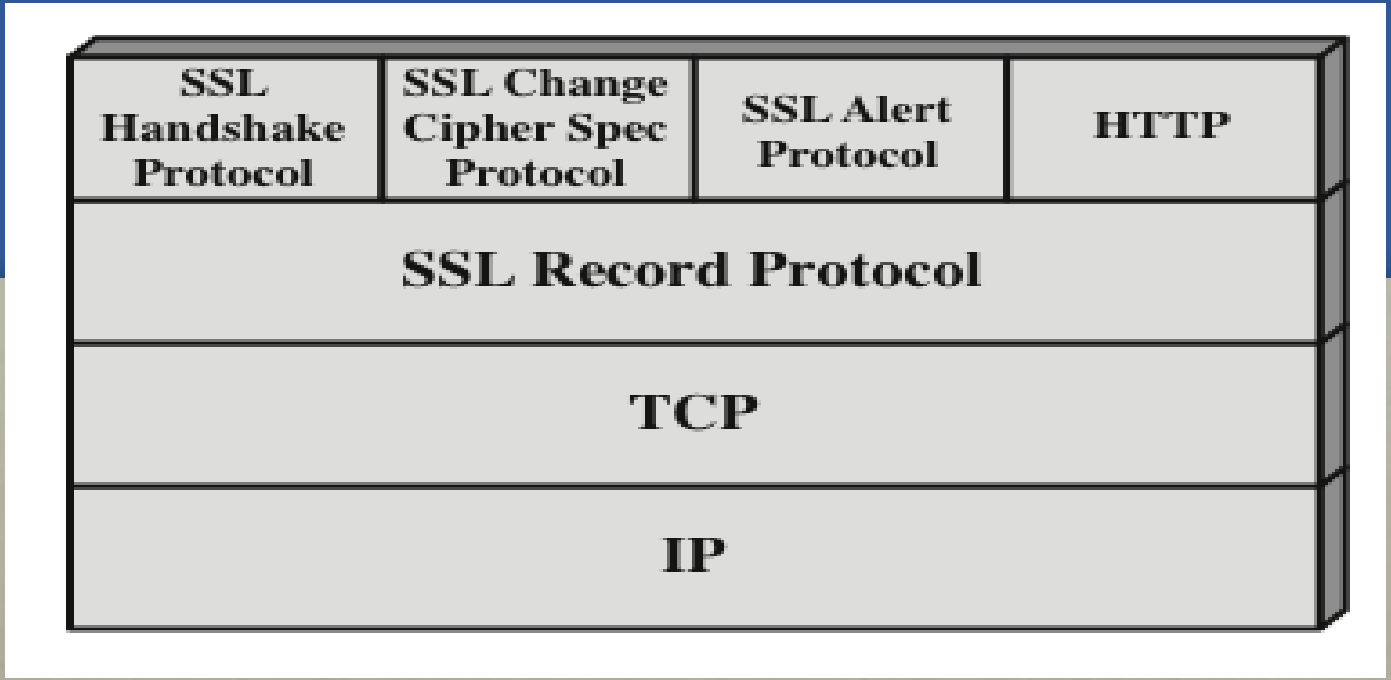
- One way to provide Web security is to use IP security (IPsec) (Figure 6.1a). The advantage of using IPsec is that it is transparent to end users and applications and provides a general-purpose solution.
- Furthermore, IPsec includes a filtering capability so that only selected traffic need incur the overhead of IPsec processing.

TRANSPORT LEVEL

- Another relatively general-purpose solution is to implement security just above TCP (Figure 6.1b).
- 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.

SECURE SOCKETS LAYER (SSL)

- One of the most widely used security services
- 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



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

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

SSL ARCHITECTURE

- Two important SSL concepts are:

SSL connection

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

SSL 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	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 and a hash algorithm used for MAC calculation; 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

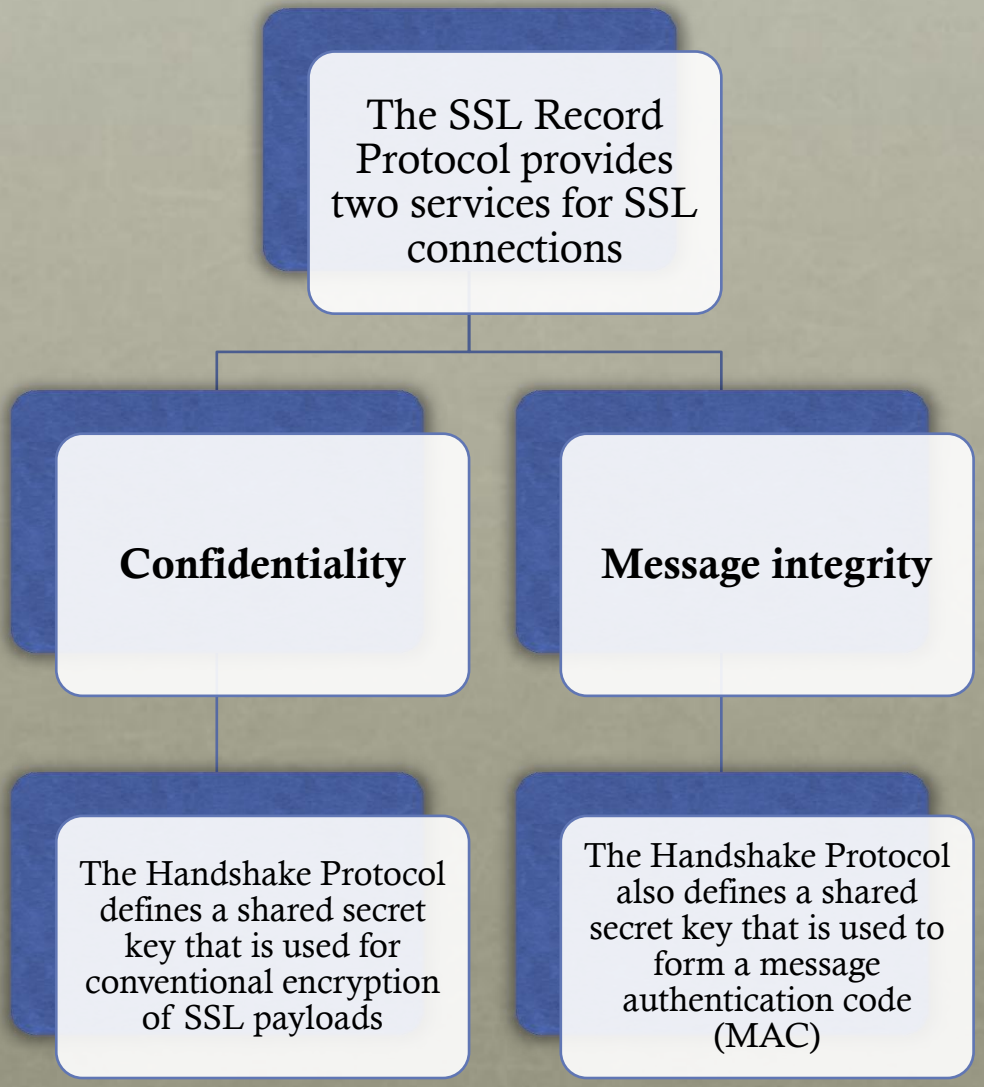
Initialization vectors

- 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
- 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



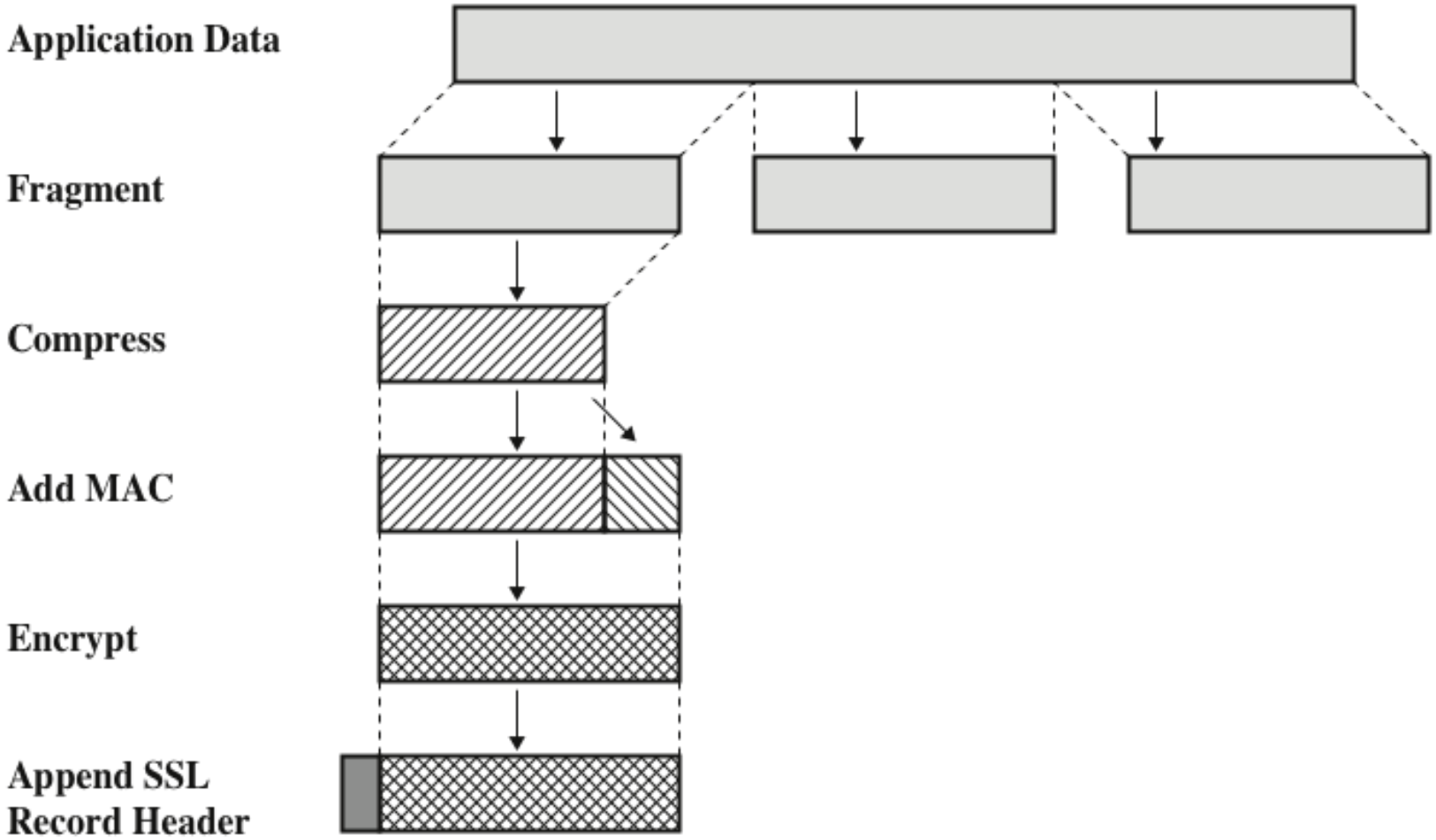


Figure 6.3 SSL Record Protocol Operation

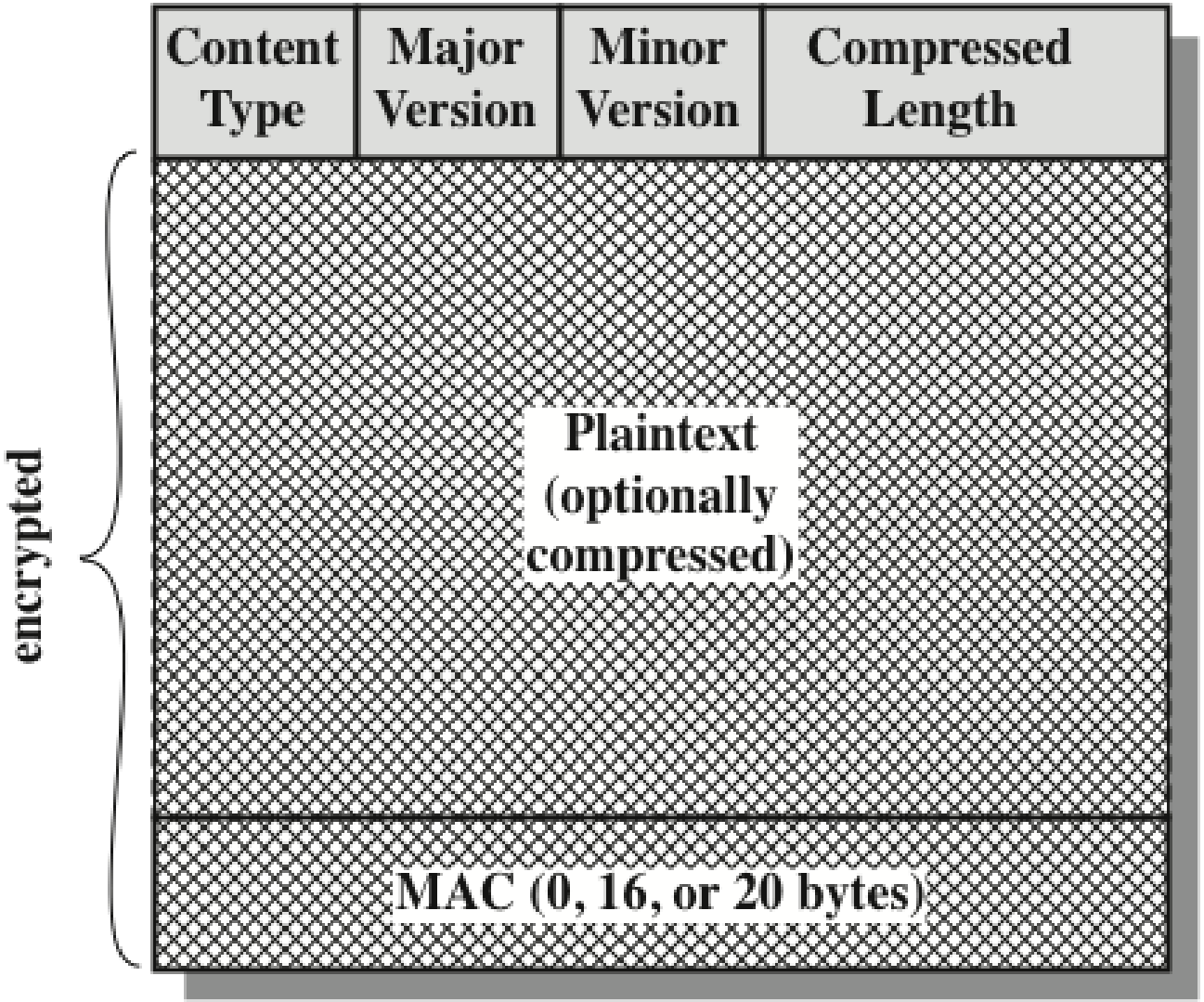


Figure 6.4 SSL Record Format

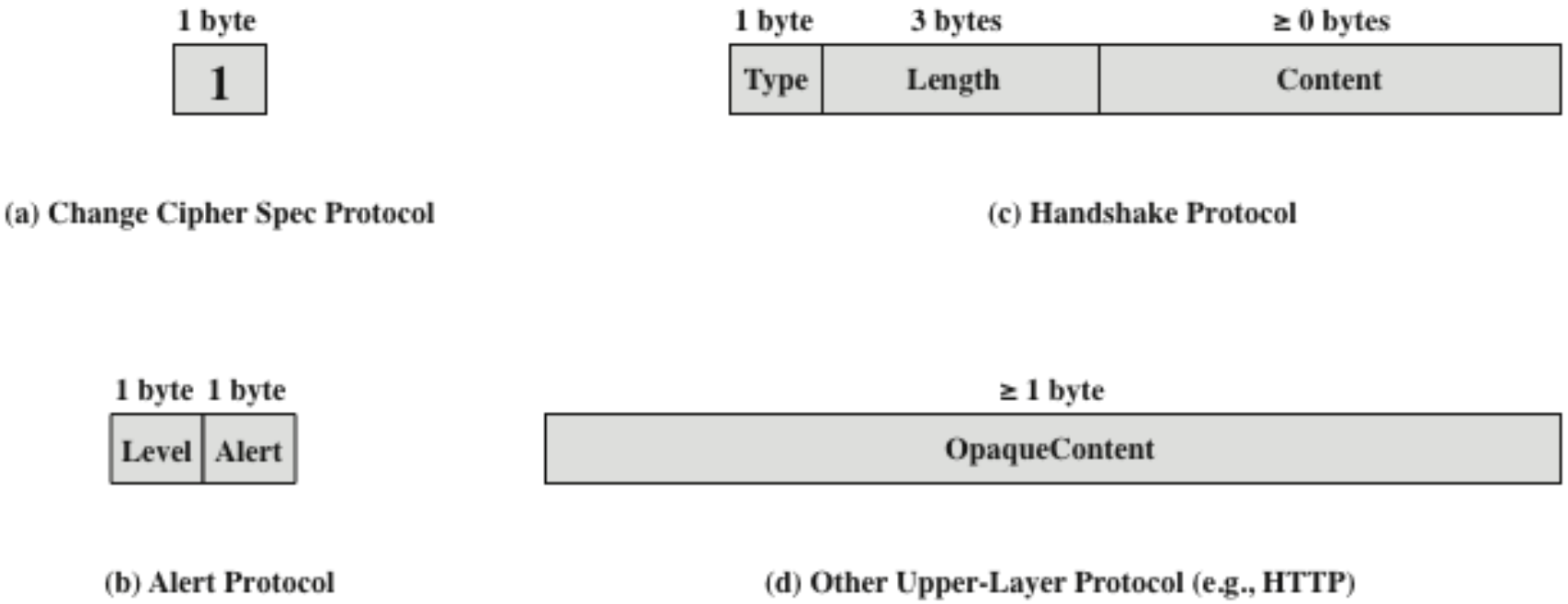


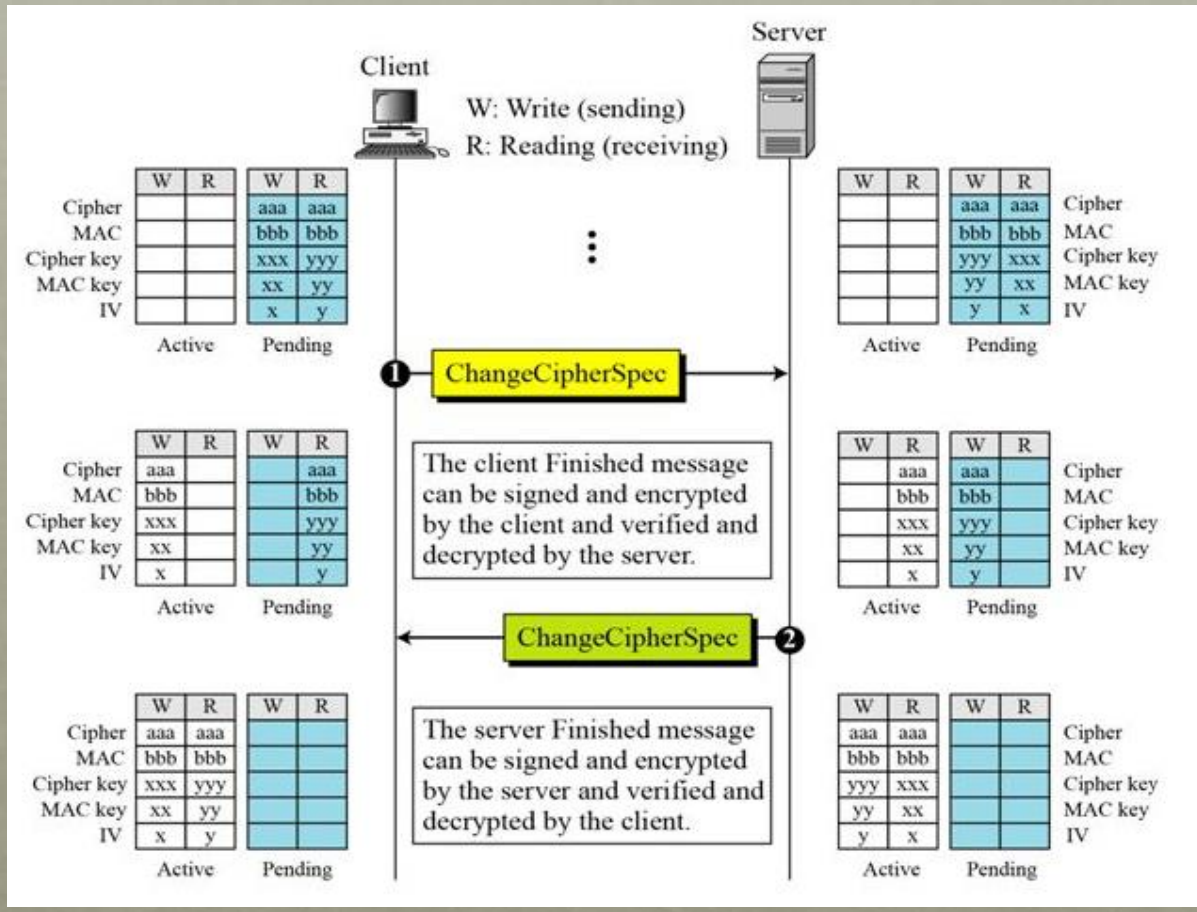
Figure 6.5 SSL Record Protocol Payload

CHANGE CIPHER PROTOCOL

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 (Figure6.5a), 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 CCS protocol is a single message that tells the peer that the sender wants to change to a new set of keys, which are then created from information exchanged by the handshake protocol.



ALERT PROTOCOL

- **Alert Protocol**
  - 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 6.5b).
    - 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.

HANDSHAKE PROTOCOL

- The most complex part of SSL is the Handshake Protocol.
- It 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.
- It is used before any application data is transmitted.
- Consists of a series of messages exchanged by client and server.
- All of these have the format shown in Figure 6.5c.
- .

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 6.2 SSL Handshake Protocol Message Types

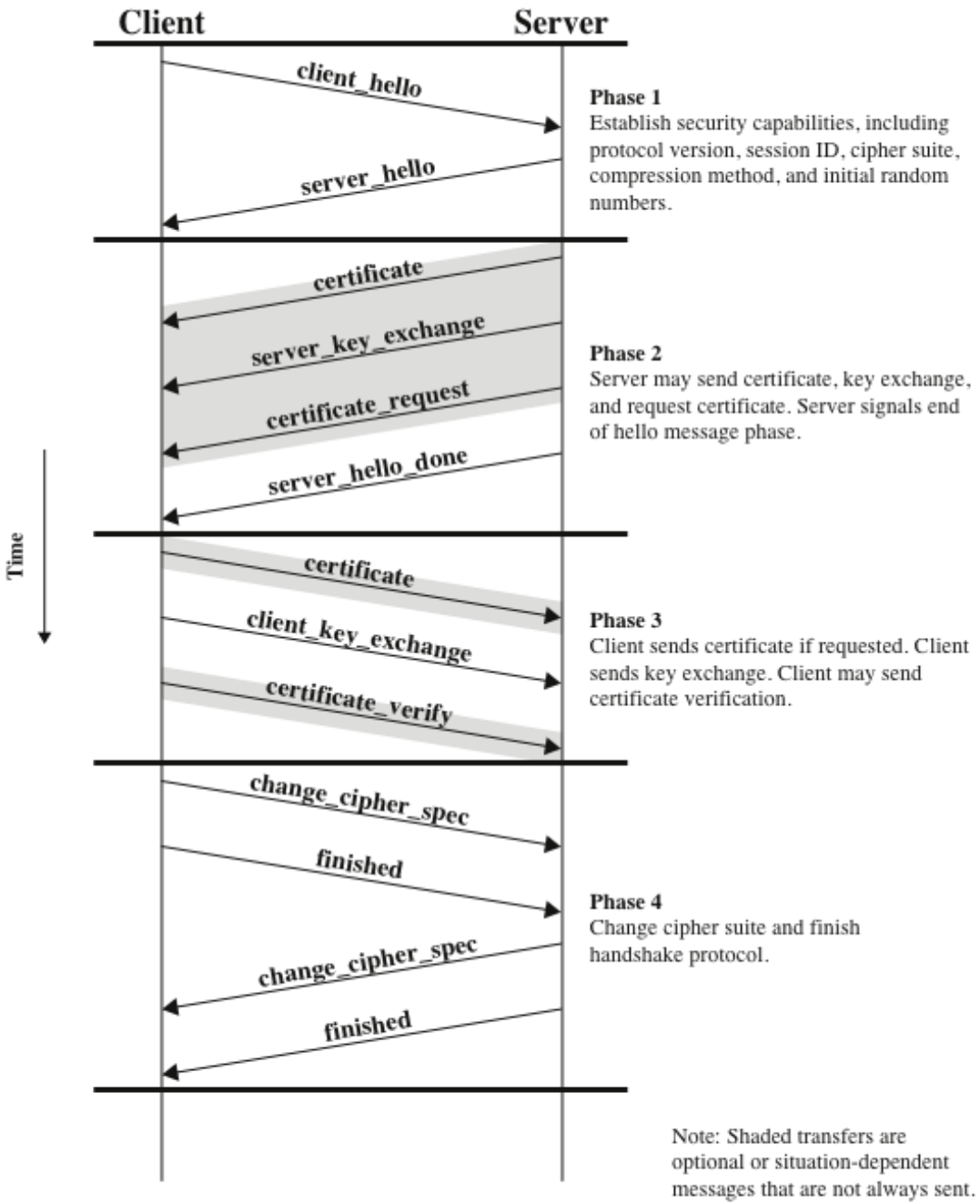
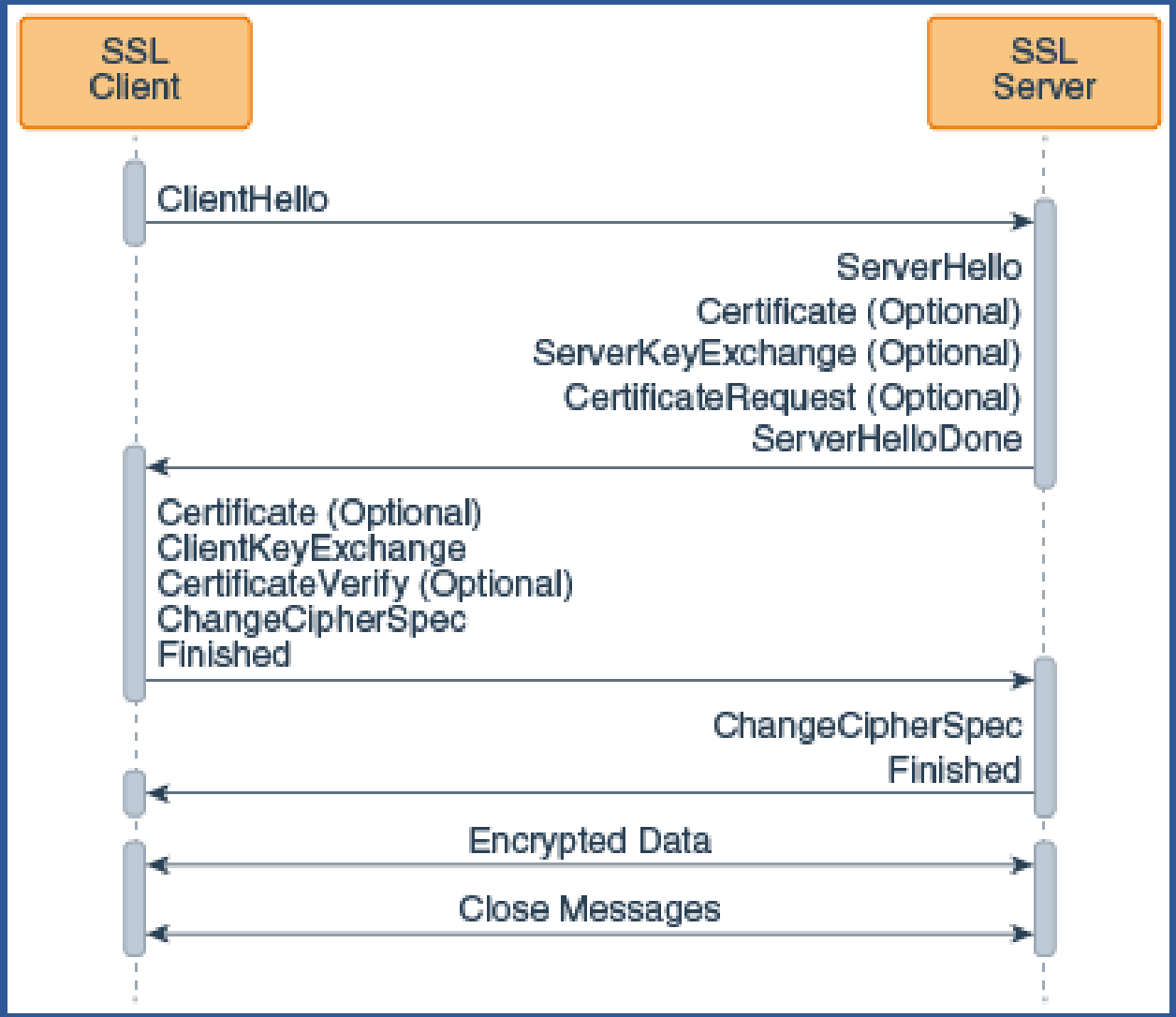


Figure 6.6 Handshake Protocol Action


SSL Client
SSL Server
ClientHello
ServerHello
Certificate (Optional)
ServerKeyExchange (Optional)
CertificateRequest (Optional)
ServerHelloDone
Certificate (Optional)
ClientKeyExchange
CertificateVerify (Optional)
ChangeCipherSpec
Finished
ChangeCipherSpec
Finished
Encrypted Data
Close Messages

CLIENT_HELLO 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 serve as nonces and 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 to 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; (RSA, Deffie-Hellman, Fourtezza)
- **Compression Method:** This is a list of the compression methods the client supports.

1. Computers agree on how to encrypt

Key	Cipher	Hash
RSA	RC4	HMAC-MD5
Diffie-Hellman	Triple DES	HMAC-SHA
DSA	AES	

Key	Cipher	Hash
RSA	RC4	HMAC-MD5
Diffie-Hellman	Triple DES	HMAC-SHA
DSA	AES	

Version	3.3
Random number	29873456234234...

2. Server sends a certificate

Serial: 123123
Issuer: Verisign
Valid: from-to
Public key
Subject:
- Site
- Company
- Address

3. Your computer says 'start encrypting'

Client Key Exchange

Both computers calculate a master secret code

Change Cipher Spec

Your computer is asking server to encrypt

Finished

Let's start now

4. The server says 'start encrypting'

I'm going to send encrypted messages now

Change Cipher Spec

Let's go

Finished

CRYPTOGRAPHIC COMPUTATIONS

- Two further items are of interest:
  - The creation of a shared master secret by means of the key exchange
    - The shared master secret is a one-time 48-byte value generated for this session by means of secure key exchange
  - The generation of cryptographic parameters from the master secret
    - CipherSpecs require a client write MAC secret, a server write MAC secret, a client write key, a server write key, a client write IV, and a server write IV which are generated from the master secret in that order
      - These parameters are generated from the master secret by hashing the master secret into a sequence of secure bytes of sufficient length for all needed parameters

TRANSPORT LAYER SECURITY (TLS)

- An IETF standardization initiative whose goal is to produce an Internet standard version of SSL
- Is defined as a Proposed Internet Standard in RFC 5246
  - RFC 5246 is very similar to SSLv3

Differences include:

- Version number
- Message Authentication Code
- Pseudorandom function
- Alert keys
- Cipher suites
- Client certificate types
- Certificate_verify and Finished Messages
- Cryptographic computations
- Padding

SSL TLS DIFFERENCE

- Version Number
  - The TLS Record Format is the same as that of the SSL Record Format and the fields in the header have the same meanings.
  - The one difference is in version values. For the current version of TLS, the major version is 3 and the minor version is 3.
- Message Authentication Code
  - There are two differences between the SSLv3 and TLS MAC schemes: the actual
  - algorithm and the scope of the MAC calculation.
  - TLS makes use of the HMAC algorithm defined in RFC 2104.
  - The MAC calculation covers all of the fields covered by the SSLv3 calculation, plus the field TLSCompressed.version , which is the version of the protocol being employed.

- Pseudorandom Function
  - TLS makes use of a pseudorandom function referred to as PRF to expand secrets into blocks of data for purposes of key generation or validation.
  - The objective is to make use of a relatively small shared secret value but to generate longer blocks of data in a way that is secure from the kinds of attacks made on hash functions and MACs.
- Alert Codes
  - TLS supports all of the alert codes defined in SSLv3 with the exception of no_certificate . A number of additional codes are defined in TLS.
- Cipher Suites
  - There are several small differences between the cipher suites available under SSLv3 and under TLS:
    - Key Exchange: TLS supports all of the key exchange techniques of SSLv3 with the exception of Fortezza.
    - Symmetric Encryption Algorithms: TLS includes all of the symmetric encryption algorithms found in SSLv3, with the exception of Fortezza.

- Client Certificate Types
  - TLS defines the following certificate types to be requested in a certificate_request message: rsa_sign, dss_sign, rsa_fixed_dh, and dss_fixed_dh.
  - These are all defined in SSLv3. In addition, SSLv3 includes rsa_ephemeral_dh, dss_ephemeral_dh, and fortezza_kea.

- certificate_verify and Finished Messages
  - In the TLS certificate_verify message, the MD5 and SHA-1 hashes are alculated only over handshake_messages. Recall that for SSLv3, the hash calculation also included the master secret and pads. These extra fields were felt to add no additional security. As with the finished message in SSLv3, the finished message in TLS is a hash based on the shared master_secret, the previous handshake messages, and a label that identifies client or server. The calculation is somewhat different.

- Cryptographic Computations
  - The pre_master_secret for TLS is calculated in the same way as in SSLv3. As in SSLv3, the master_secret in TLS is calculated as a hash function of the pre_master_secret and the two hello random numbers.
  - As with SSLv3, the key_block is a function of the master_secret and the client and server random numbers, but for TLS, the actual algorithm is different.

- Padding
  - In SSL, the padding added prior to encryption of user data is the minimum amount required so that the total size of the data to be encrypted is a multiple of the cipher's block length.
  - In TLS, the padding can be any amount that results in a total that is a multiple of the cipher's block length, up to a maximum of 255 bytes. For example, if the plaintext (or compressed text if compression is used) plus MAC plus padding.length byte is 79 bytes long, then the padding length (in bytes) can be 1, 9, 17, and so on, up to 249. A variable padding length may be used to frustrate attacks based on an analysis of the lengths of exchanged messages.

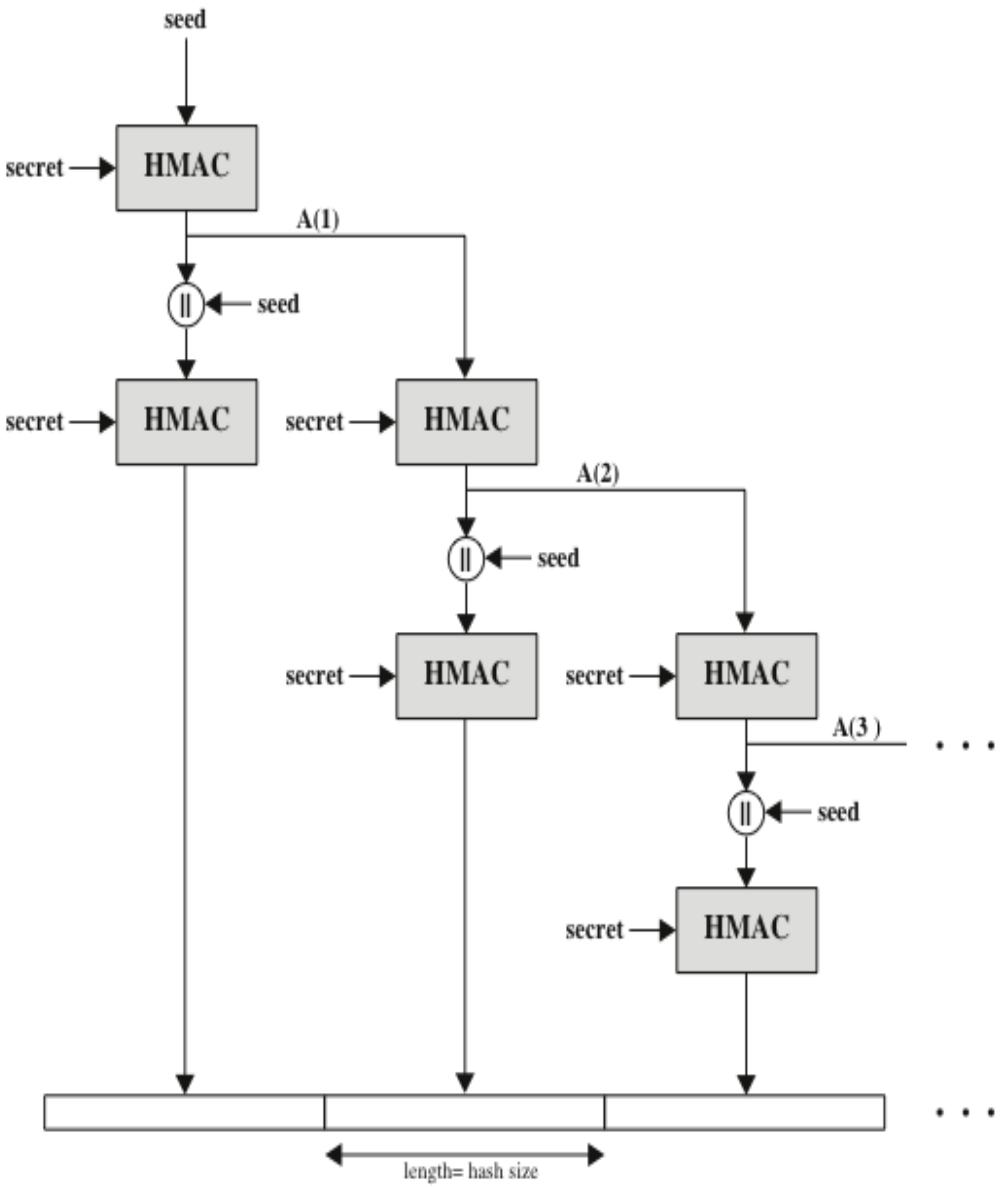


Figure 6.7 TLS Function P_hash (secret, seed)

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 it 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

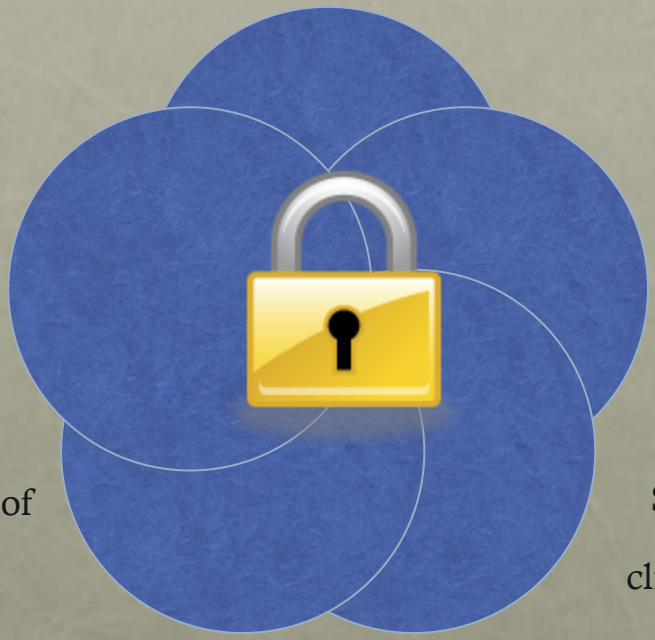
- An HTTP client or server can indicate the closing of a connection by including the line `Connection: close` in an HTTP record
- The closure of an HTTPS connection requires that TLS close the connection with the peer TLS entity on the remote side, which will involve closing the underlying TCP connection
- 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

SECURE SHELL (SSH)

A protocol for secure network communications designed to be relatively simple and inexpensive to implement

SSH client and server applications are widely available for most operating systems

- Has become the method of choice for remote login and X tunneling
- Is rapidly becoming one of the most pervasive applications for encryption technology outside of embedded systems



The initial version, SSH1 was focused on providing a secure remote logon facility to replace TELNET and other remote logon schemes that provided no security

SSH2 fixes a number of security flaws in the original scheme

- Is documented as a proposed standard in IETF RFCs 4250 through 4256

SSH also provides a more general client/server capability and can be used for such network functions as file transfer and e-mail

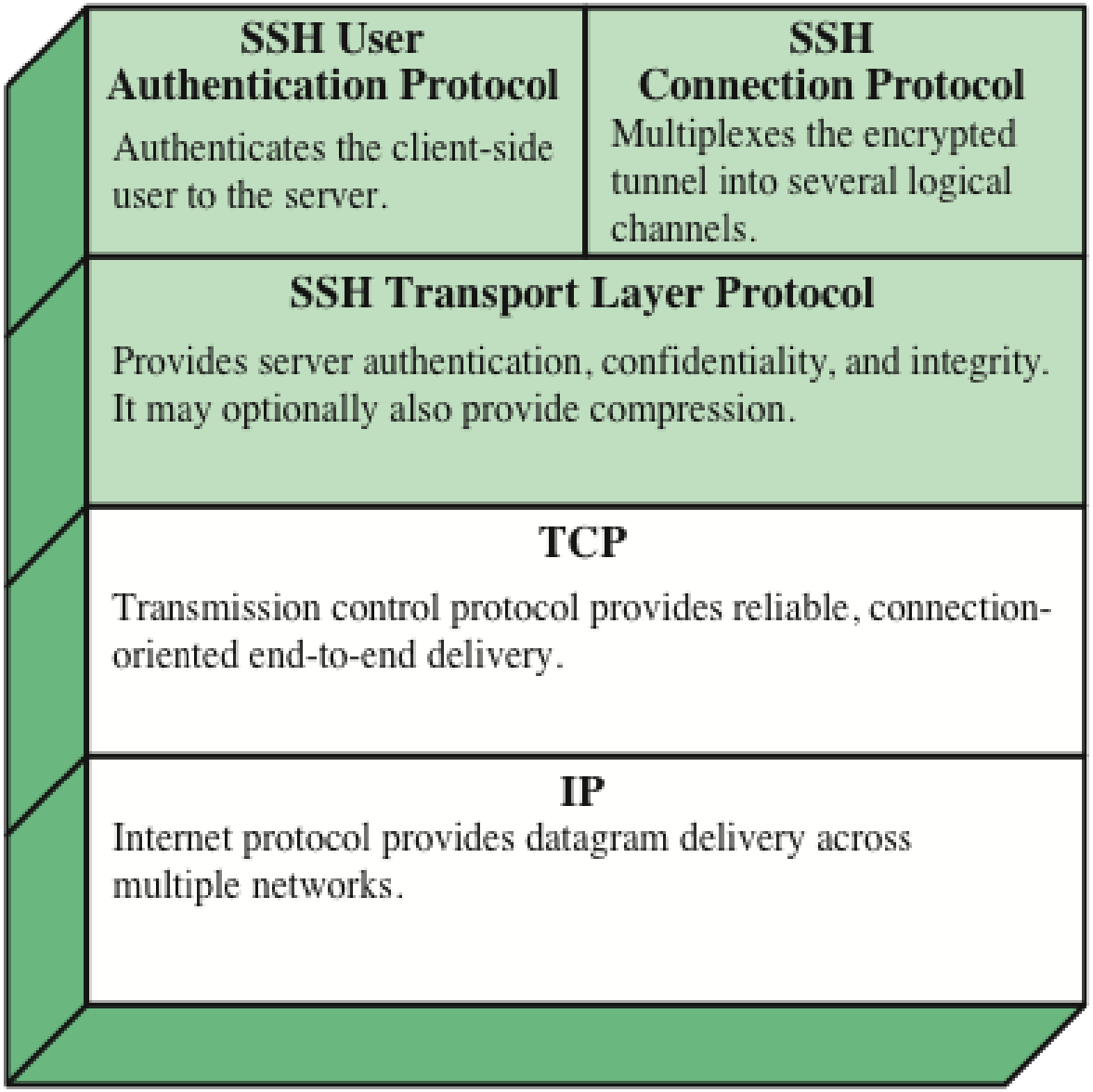


Figure 6.8 SSH Protocol Stack

TRANSPORT LAYER PROTOCOL

- Server authentication occurs at the transport layer, based on the server possessing a public/private key pair
- A server may have multiple host keys using multiple different asymmetric encryption algorithms
- Multiple hosts may share the same host key
- The server host key is used during key exchange to authenticate the identity of the host
- RFC 4251 dictates two alternative trust models:
  - The client has a local database that associates each host name with the corresponding public host key
  - The host name-to-key association is certified by a trusted certification authority (CA); the client only knows the CA root key and can verify the validity of all host keys certified by accepted CAs

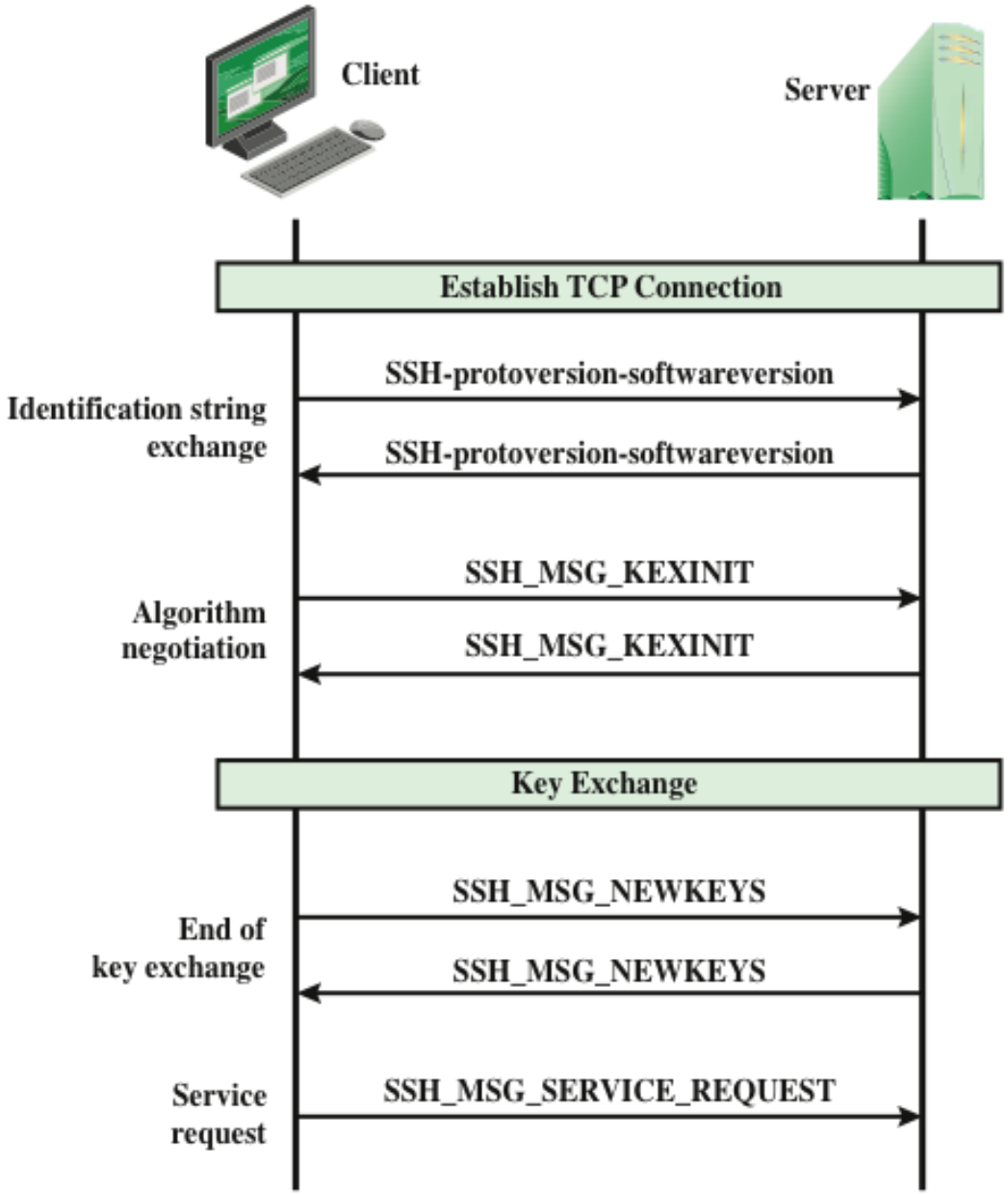
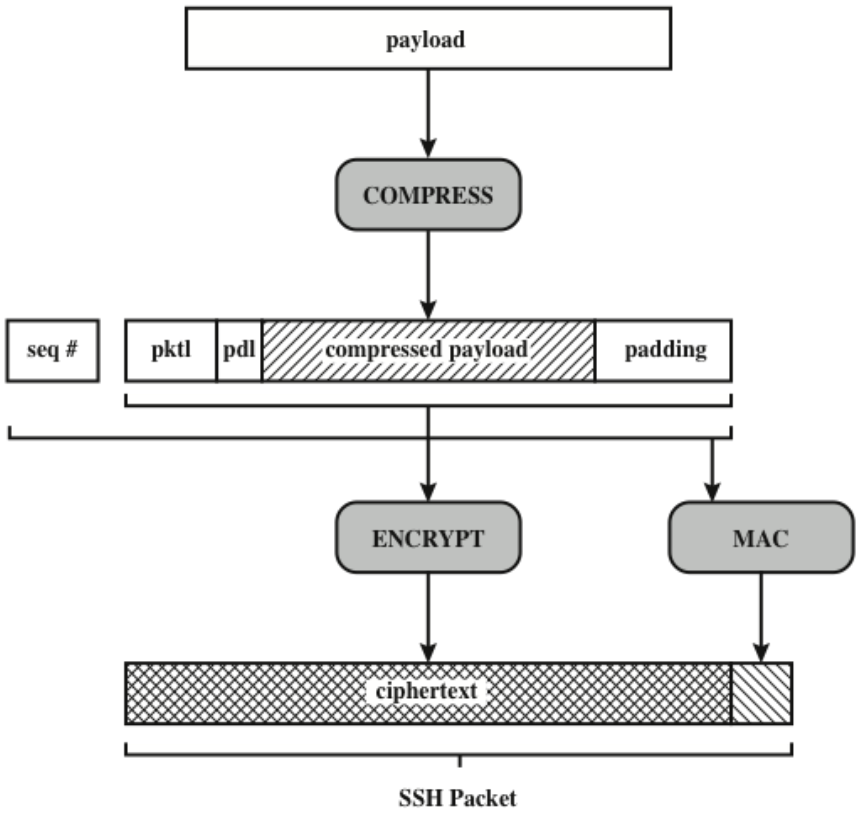


Figure 6.9 SSH Transport Layer Protocol Packet Exchanges



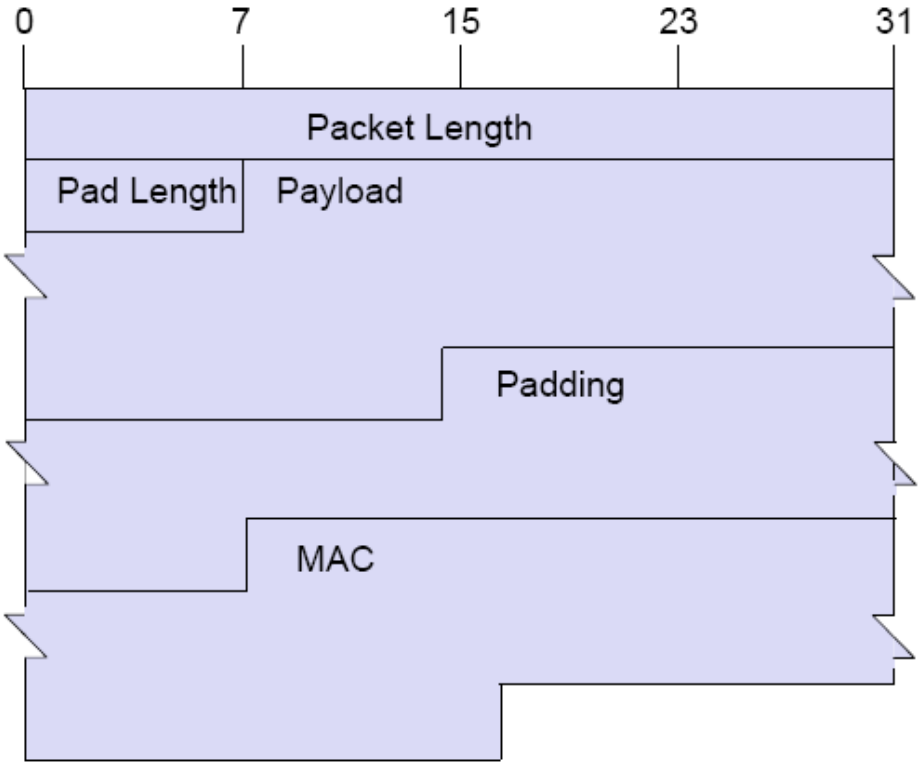


Figure 6.10 SSH Transport Layer Protocol Packet Formation

PACKET FIELDS

- *Packet length:* the length of the packet itself, not including this length field and the MAC

- *Padding length:* length of the padding field, must be between four and 255

- *Payload:* the actual payload of the packet, if compression is negotiated this field is compressed

- *Padding:* this field consists of randomly chosen octets to fill up the payload to an integer multiple of 8 or the block size of the encryption algorithm, whichever is larger

- *MAC:* if message authentication has been negotiated this contains theMAC over the entire packet without the MAC field itself, if the packet is to be encrypted the MAC is computed prior to encryption as follows:

- *MAC = HMAC(shared_secret, seq_number||unencrypted_packet)*

with *seq_number*d enoting a 32-bit sequence number for every packet

Cipher	
3des-cbc*	Three-key 3DES in CBC mode
blowfish-cbc	Blowfish in CBC mode
twofish256-cbc	Twofish in CBC mode with a 256-bit key
twofish192-cbc	Twofish with a 192-bit key
twofish128-cbc	Twofish with a 128-bit key
aes256-cbc	AES in CBC mode with a 256-bit key
aes192-cbc	AES with a 192-bit key
aes128-cbc**	AES with a 128-bit key
Serpent256-cbc	Serpent in CBC mode with a 256-bit key
Serpent192-cbc	Serpent with a 192-bit key
Serpent128-cbc	Serpent with a 128-bit key
arcfour	RC4 with a 128-bit key
cast128-cbc	CAST-128 in CBC mode

MAC algorithm	
hmac-sha1*	HMAC-SHA1; digest length = key length = 20
hmac-sha1-96**	First 96 bits of HMAC-SHA1; digest length = 12; key length = 20
hmac-md5	HMAC-MD5; digest length = key length = 16
hmac-md5-96	First 96 bits of HMAC-MD5; digest length = 12; key length = 16

Compression algorithm	
none*	No compression
zlib	Defined in RFC 1950 and RFC 1951

* = Required
** = Recommended

Table 6.3 SSH Transport Layer Cryptographic Algorithms

AUTHENTICATION METHODS

Publickey

- The client sends a message to the server that contains the client's public key, with the message signed by the client's private key
- When the server receives this message, it checks whether the supplied key is acceptable for authentication and, if so, it checks whether the signature is correct

Password

- The client sends a message containing a plaintext password, which is protected by encryption by the Transport Layer Protocol

Hostbased

- Authentication is performed on the client's host rather than the client itself
- This method works by having the client send a signature created with the private key of the client host
- Rather than directly verifying the user's identity, the SSH server verifies the identity of the client host

CONNECTION PROTOCOL

- The SSH Connection Protocol runs on top of the SSH Transport Layer Protocol and assumes that a secure authentication connection is in use
  - The secure authentication connection, referred to as a *tunnel,* is used by the Connection Protocol to multiplex a number of logical channels

- Channel mechanism
  - All types of communication using SSH are supported using separate channels
  - Either side may open a channel
  - For each channel, each side associates a unique channel number
  - Channels are flow controlled using a window mechanism
  - No data may be sent to a channel until a message is received to indicate that window space is available
  - The life of a channel progresses through three stages: opening a channel, data transfer, and closing a channel

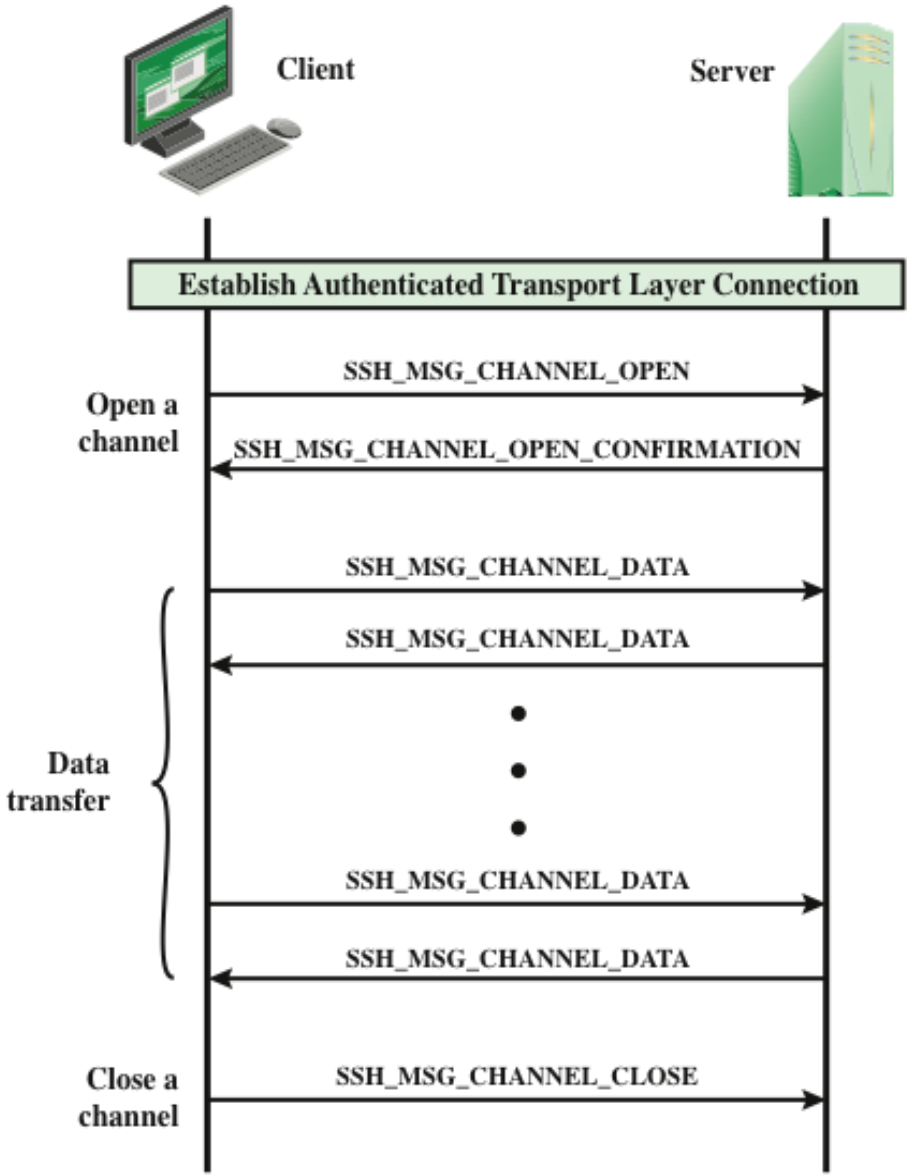


Figure 6.11 Example SSH Connection Protocol Message Exchange

CHANNEL TYPES

Four channel types are recognized in the SSH Connection Protocol specification

Session

- The remote execution of a program
- The program may be a shell, an application such as file transfer or e-mail, a system command, or some built-in subsystem
- Once a session channel is opened, subsequent requests are used to start the remote program

X11

- Refers to the X Window System, a computer software system and network protocol that provides a graphical user interface (GUI) for networked computers
- X allows applications to run on a network server but to be displayed on a desktop machine

Forwarded-tcpip

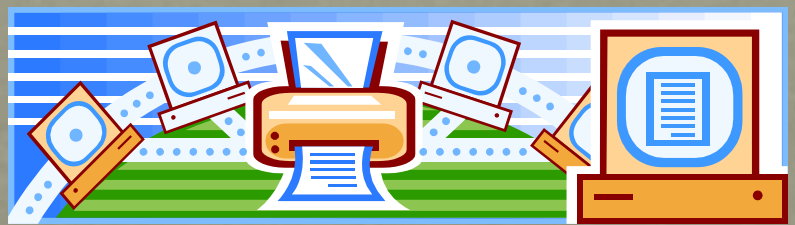
- Remote port forwarding

Direct-tcpip

- Local port forwarding

PORT FORWARDING

- One of the most useful features of SSH
- Provides the ability to convert any insecure TCP connection into a secure SSH connection (also referred to as SSH tunneling)
- Incoming TCP traffic is delivered to the appropriate application on the basis of the port number (a port is an identifier of a user of TCP)
- An application may employ multiple port numbers



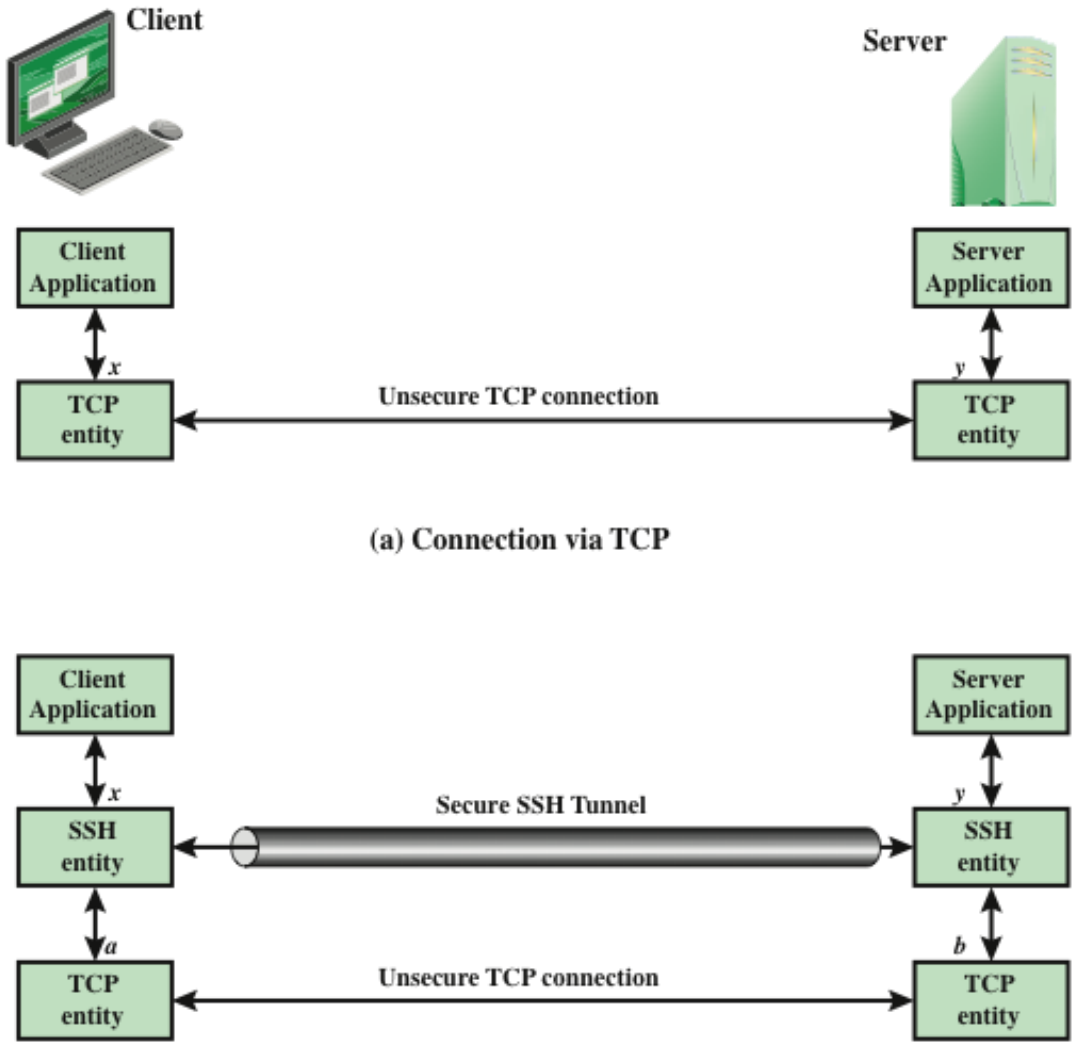


Figure 6.12 SSH Transport Layer Packet Exchanges