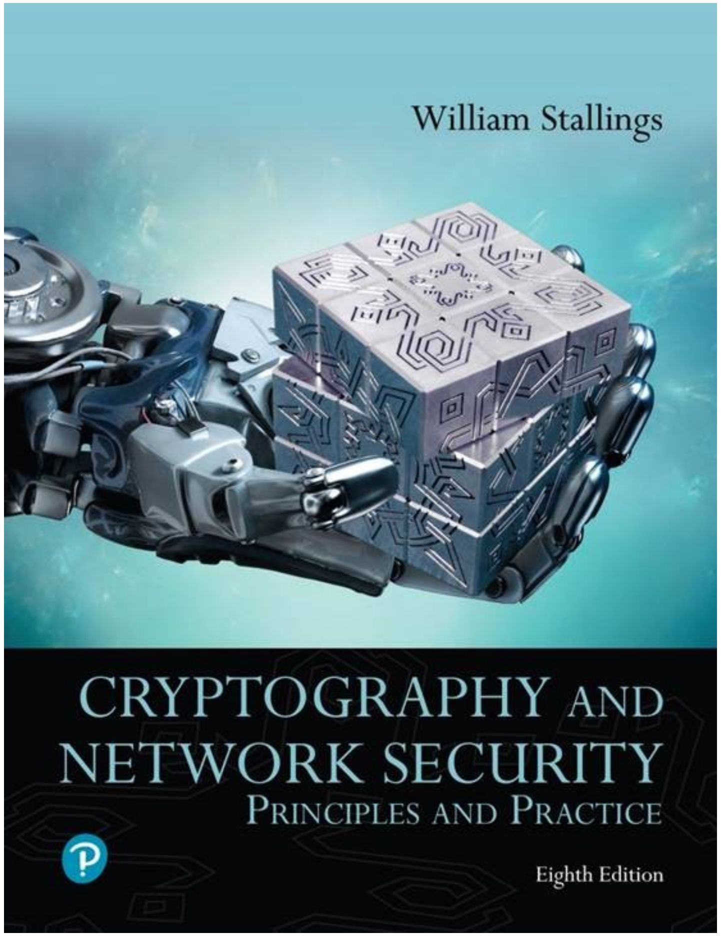
Cryptography and Network Security: Principles and Practice

Eighth Edition



Chapter 17

Transport-Level Security

Pearson

Web Security Considerations (1 of 2)

- 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

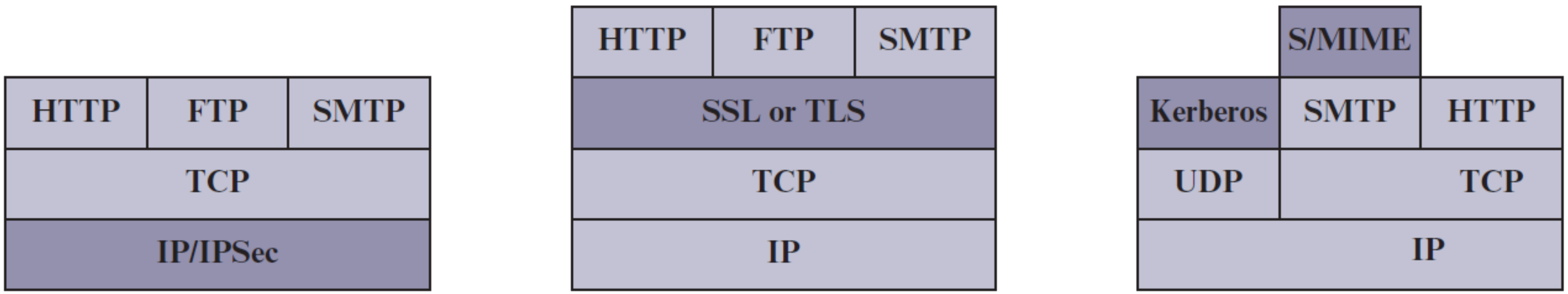
Web Security Considerations (2 of 2)

- 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

Table 17.1 A Comparison of Threats on the Web

	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 • Vulnerability 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

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



(a) Network level

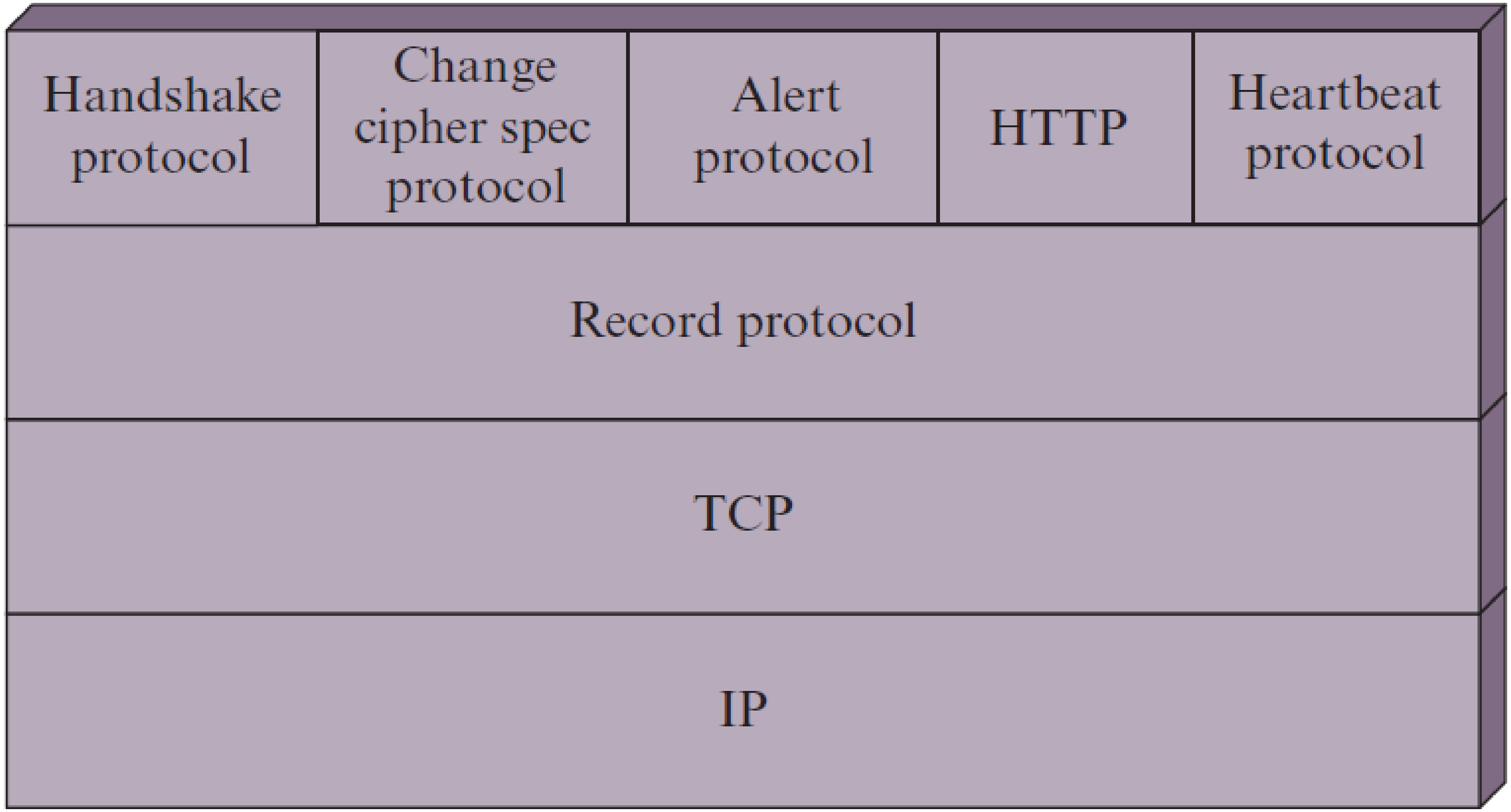
(b) Transport level

(c) Application level

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 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
  - 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: (1 of 2)

- 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

A connection state is defined by the following parameters: (2 of 2)

- 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 TLS 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 $2^{64} - 1$

TLS Record Protocol

- The TLS Record Protocol provides two services for TLS connections
  - **Confidentiality**
    - The Handshake Protocol defines a shared secret key that is used for conventional encryption of TLS payloads
  - **Message Integrity**
    - The Handshake Protocol also defines a shared secret key that is used to form a message authentication code (MAC)

Figure 17.3 TLS Record Protocol Operation

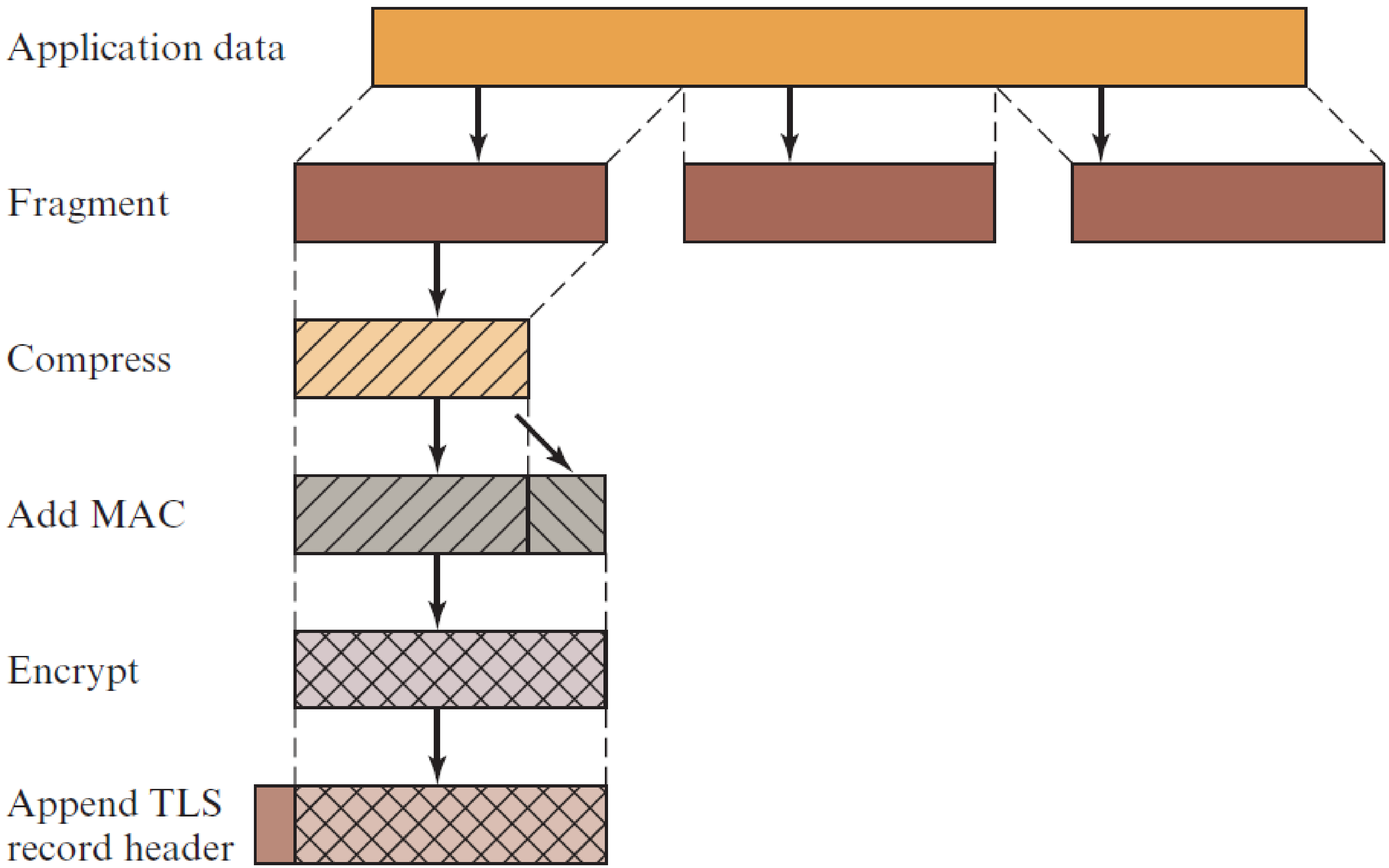
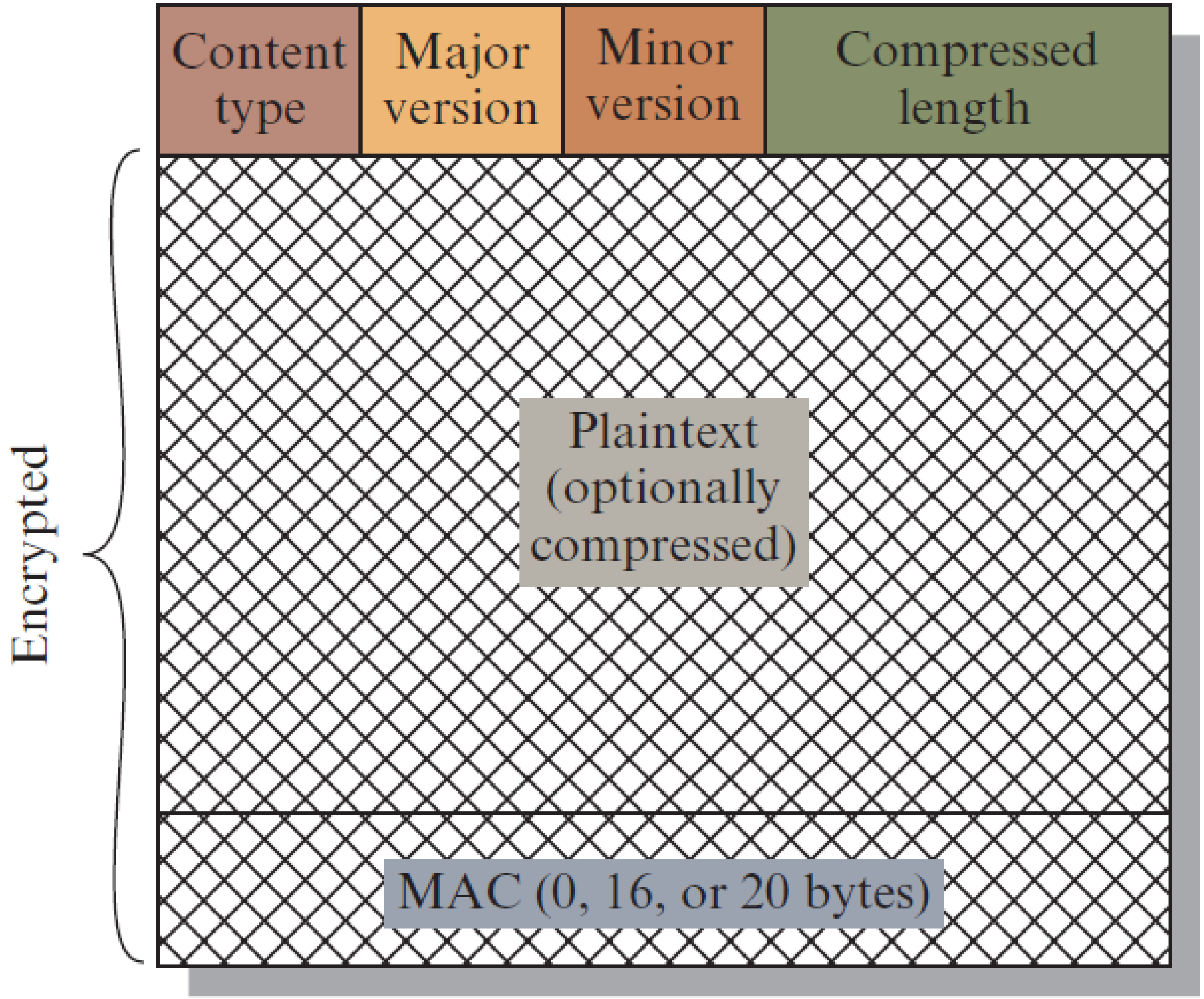


Figure 17.4 TLS Record Format



Pearson

Figure 17.5 TLS Record Protocol Payload

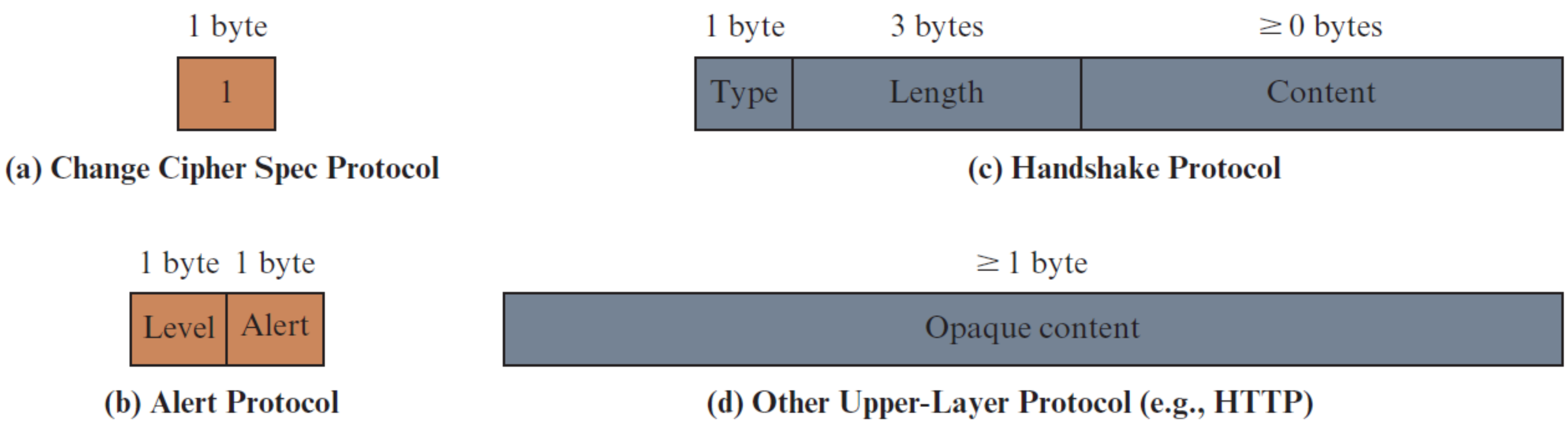
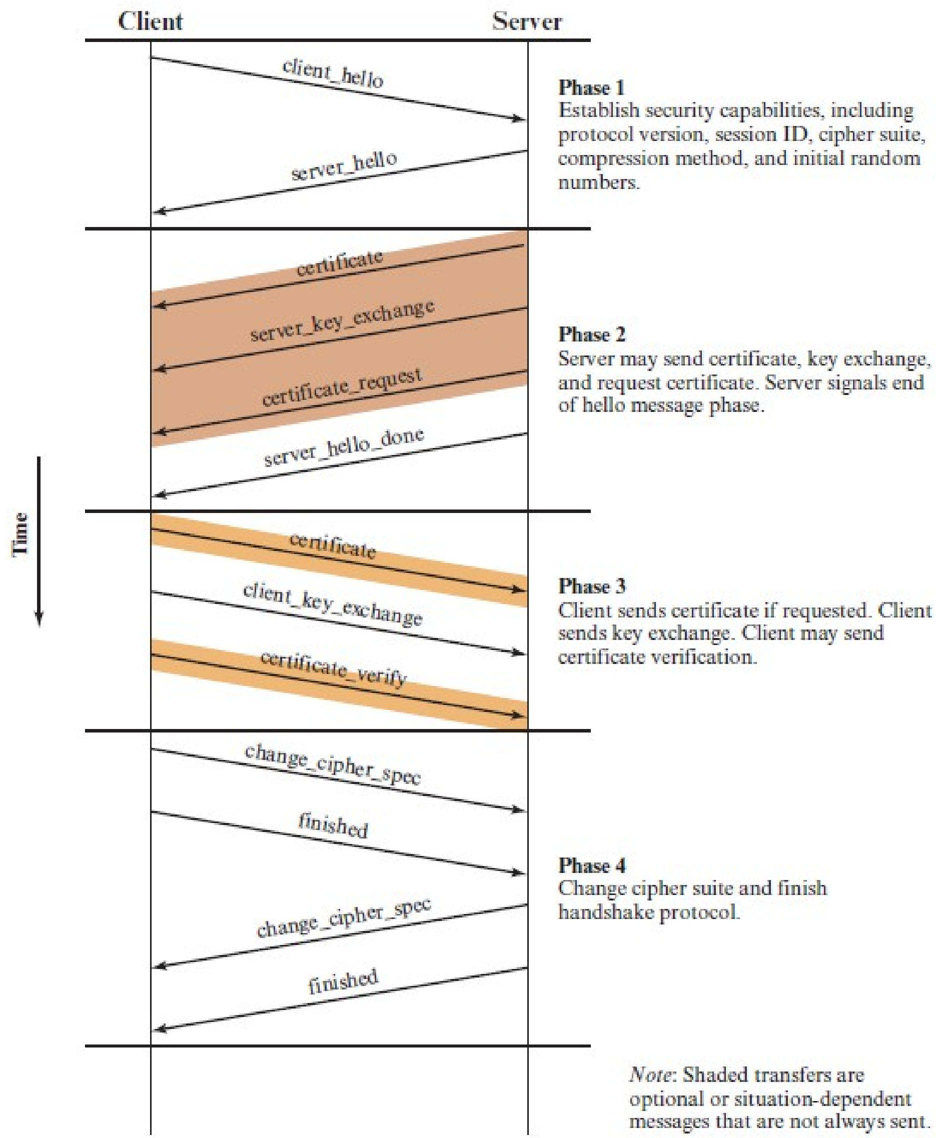


Table 17.2 TLS Handshake Protocol Message Types

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

Figure 17.6 Handshake Protocol Action



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 creation is in two stages
      - First, a pre_master_secret is exchanged
      - Second, the master_secret is calculated by both parties
  - The generation of cryptographic parameters from the master secret

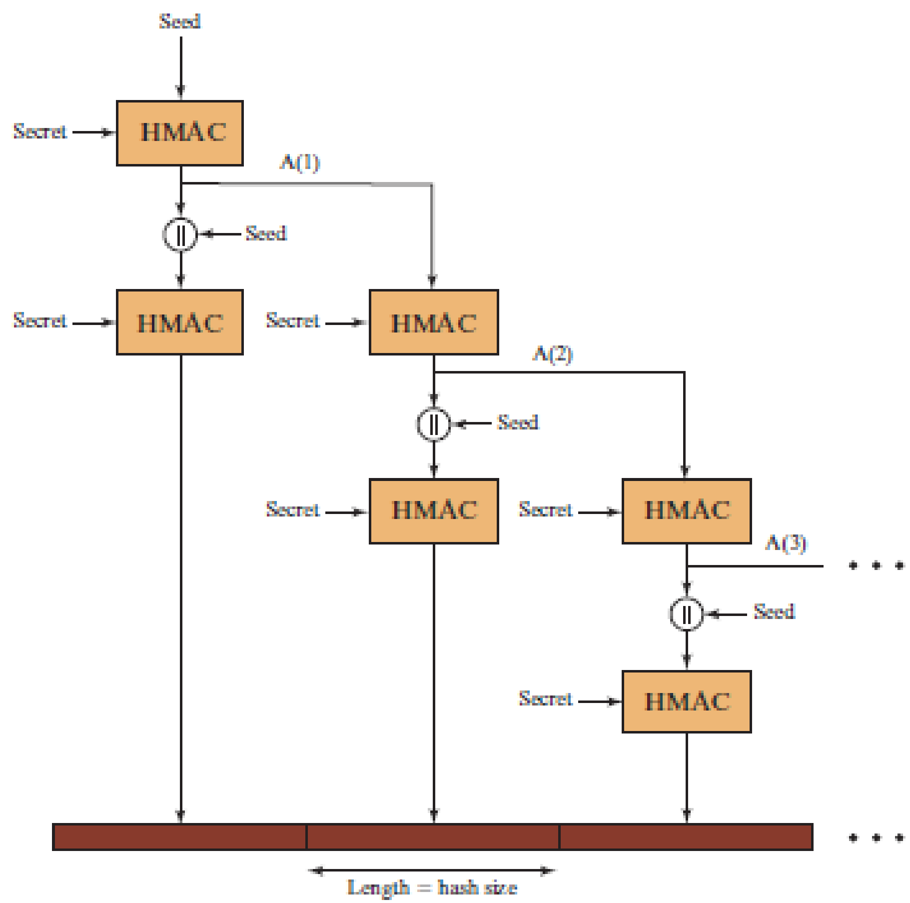
Generation of Cryptographic Parameters

- CipherSpecs require:
  - A client write MAC secret
  - A server write MAC secret
  - A client write key
  - A server write key
  - A client write IV
  - 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

Figure 17.7 TLS Function P_hash (secret, seed)



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

Hyper Text Transfer Protocol Secure (HTTPS)

- The secure version of HTTP
- HTTPS encrypts all communications between the browser and the website
- Data sent using HTTPS provides three important areas of protection:
  - Encryption
  - Data integrity
  - Authentication

Connection Initiation (1 of 2)

- 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

Connection Initiation (2 of 2)

- 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

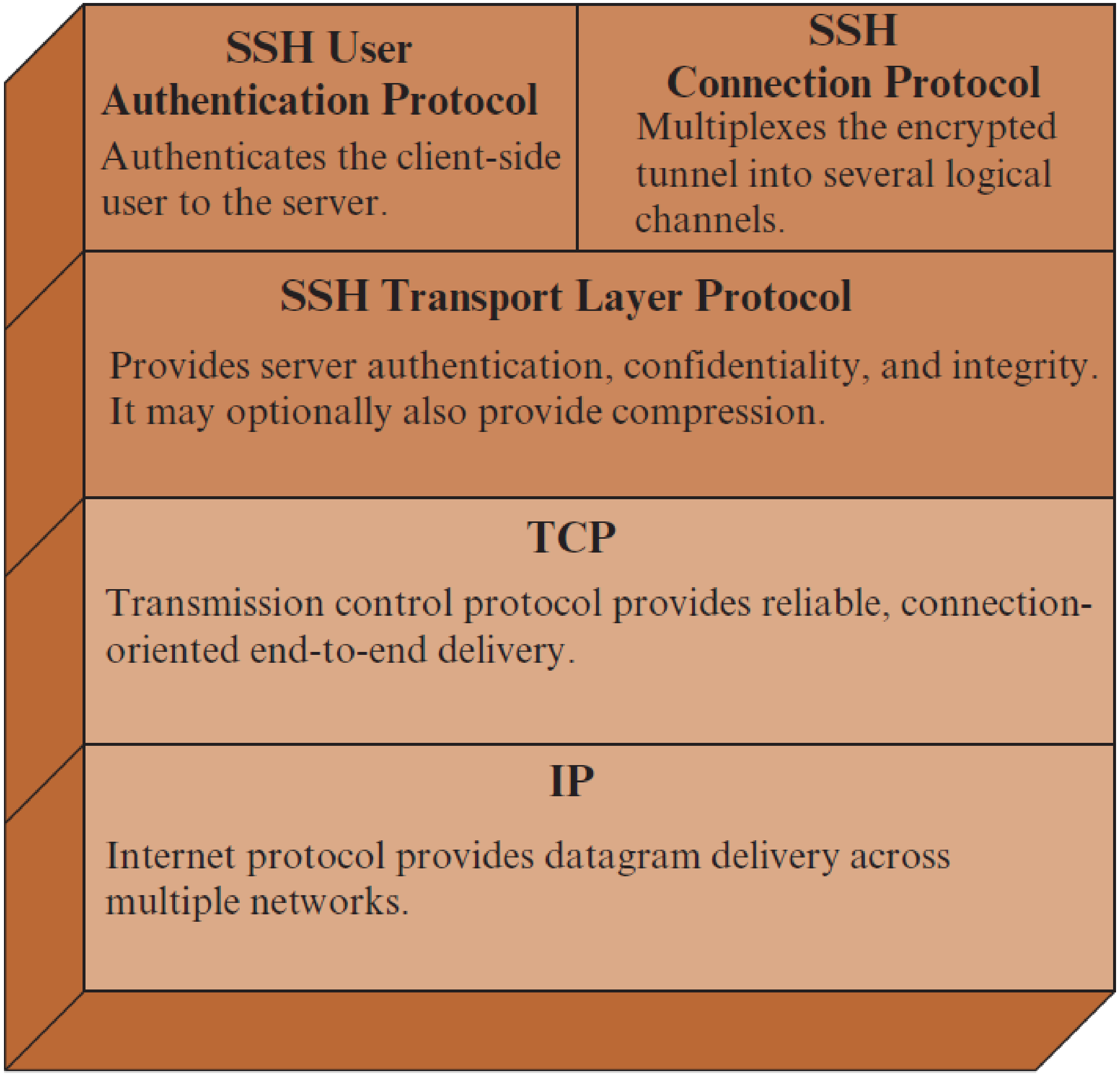
Connection Closure

- 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

Secure Shell (SSH)

- A protocol for secure network communications designed to be relatively simple and inexpensive to implement
- 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
- SSH also provides a more general client/server capability and can be used for such network functions as file transfer and e-mail
- SSH2 fixes a number of security flaws in the original scheme and is documented as a proposed standard in IETF RFCs 4250 through 4256
- 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

Figure 17.8 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

Pearson

Figure 17.9 SSH Transport Layer Protocol Packet Exchanges

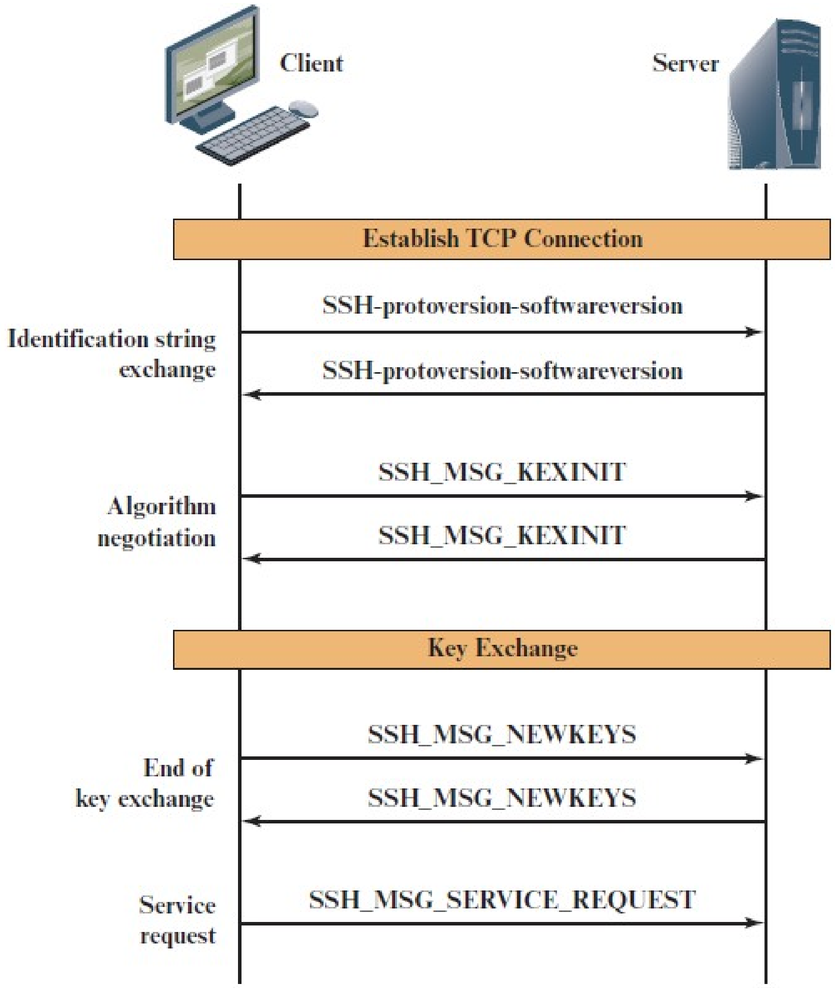


Figure 17.10 SSH Transport Layer Protocol Packet Formation

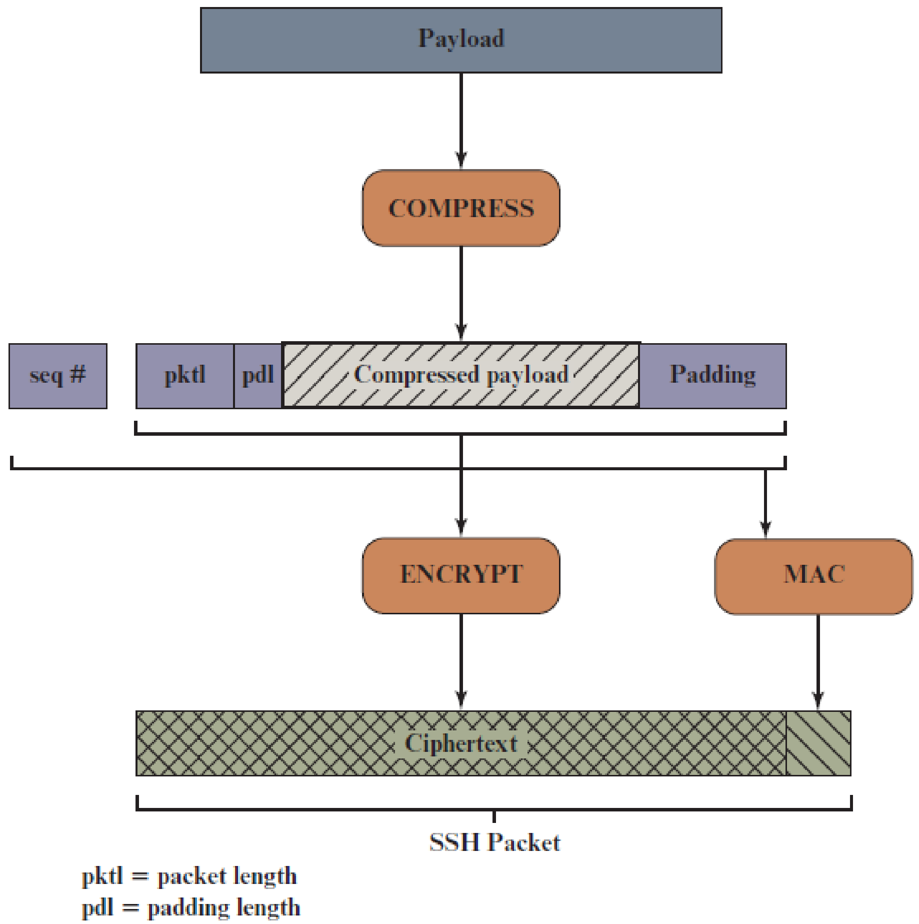


Table 17.3 SSH Transport Layer Cryptographic Algorithms

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

* = Required
** = Recommended

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

Key Generation

- The keys used for encryption and MAC (and any needed IVs) are generated from the shared secret key *K*, the hash value from the key exchange *H*, and the session identifier, which is equal to *H* unless there has been a subsequent key exchange after the initial key exchange

User Authentication Protocol

- The User Authentication Protocol provides the means by which the client is authenticated to the server
- Three types of messages are always used in the User Authentication Protocol
- User name is the authorization identity the client is claiming, service name is the facility to which the client is requesting access, and method name is the authentication method being used in this request

Message Exchange (1 of 2)

- The message exchange involves the following steps.
    - The client sends a SSH_MSG_USERAUTH_REQUEST with a requested method of none
    - The server checks to determine if the user name is valid. If not, the server returns SSH_MSG_USERAUTH_FAILURE with the partial success value of false. If the user name is valid, the server proceeds to step 3
    - The server returns SSH_MSG_USERAUTH_FAILURE with a list of one or more authentication methods to be used

Message Exchange (2 of 2)

- The client selects one of the acceptable authentication methods and sends a SSH_MSG_USERAUTH_REQUEST with that method name and the required method-specific fields. At this point, there may be a sequence of exchanges to perform the method
- If the authentication succeeds and more authentication methods are required, the server proceeds to step 3, using a partial success value of true. If the authentication fails, the server proceeds to step 3, using a partial success value of false
- When all required authentication methods succeed, the server sends a SSH_MSG_USERAUTH_SUCCESS message, and the Authentication Protocol is over

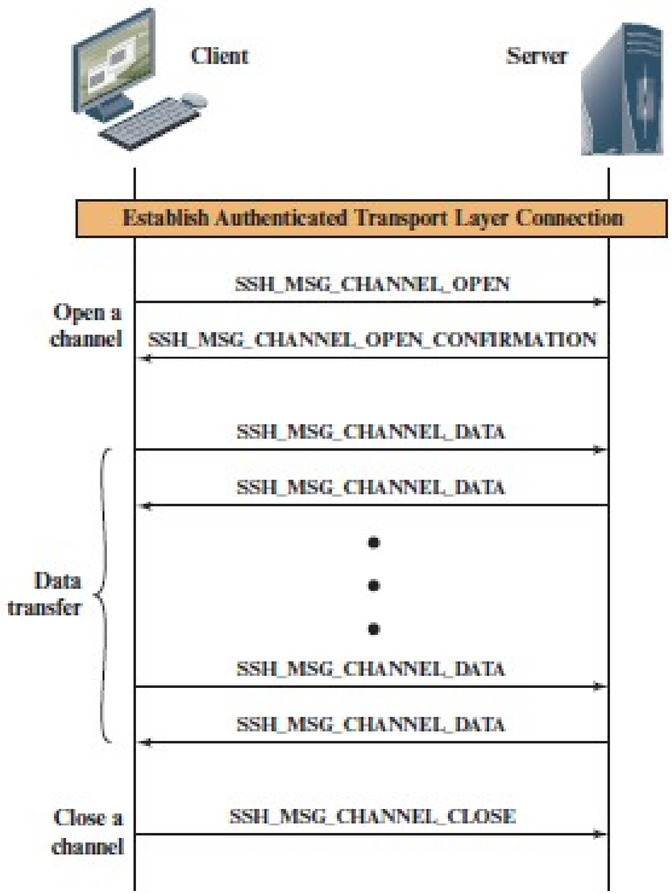
Authentication Methods

- 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 17.11 Example of SSH Connection Protocol Message Exchange



Channel Types

Four channel types are recognized in the SSH Connection Protocol specification

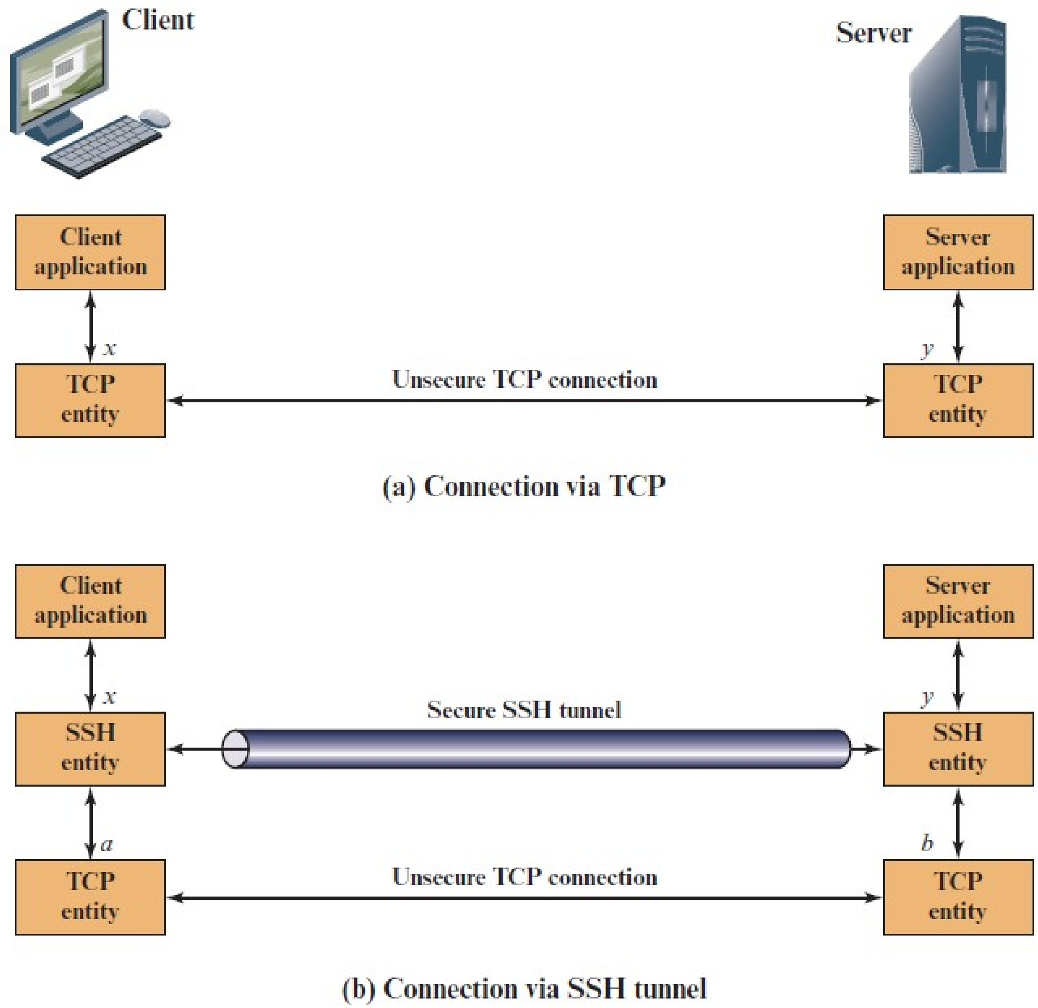
- Session
  - The remote execution of a program
  - The program may be a shell, an application such as file transfer or e-mail, a system command, or some built-in subsystem
  - Once a session channel is opened, subsequent requests are used to start the remote program
- X11
  - Refers to the X Window 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

Pearson

Figure 17.12 SSH Transport Layer Packet Exchanges



Summary

- Summarize Web security threats and Web traffic security approaches
- Present an overview of Transport Layer Security (TLS)
- Understand the differences between Secure Sockets Layer and Transport Layer Security
- Compare the pseudorandom function used in Transport Layer Security with those discussed earlier in the book
- Present an overview of HTTPS (HTTP over SSL)
- Present an overview of Secure Shell (SSH)

Pearson

Copyright