Fundamentos de Segurança Informática

LEI

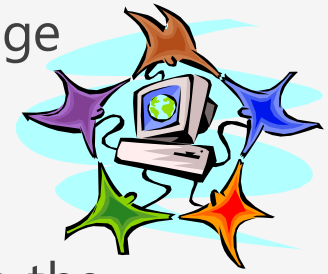
2025/2026

T7 – Web and 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 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 •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

Security approaches in the context of the TCP/IP stack

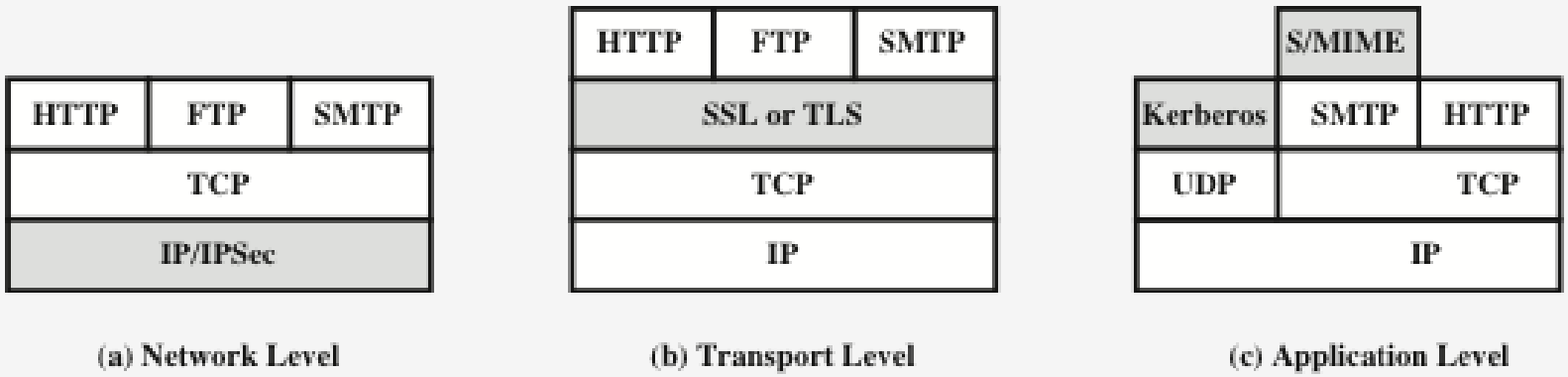


Figure 17.1 Relative Location of Security Facilities in the TCP/IP Protocol Stack

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

SSL/TLS Protocol Stack

- TLS is not a single protocol, but rather two layers of protocols
- The **TLS Record Protocol** provides basic security services to various higher layer protocols, in particular, the Hypertext Transfer Protocol (HTTP)
- Three higher-layer protocols are defined as part of TLS: the **Handshake Protocol**, the **Change Cipher Spec Protocol**, and the **Alert Protocol**. These TLS specific protocols are used in the management of TLS exchanges

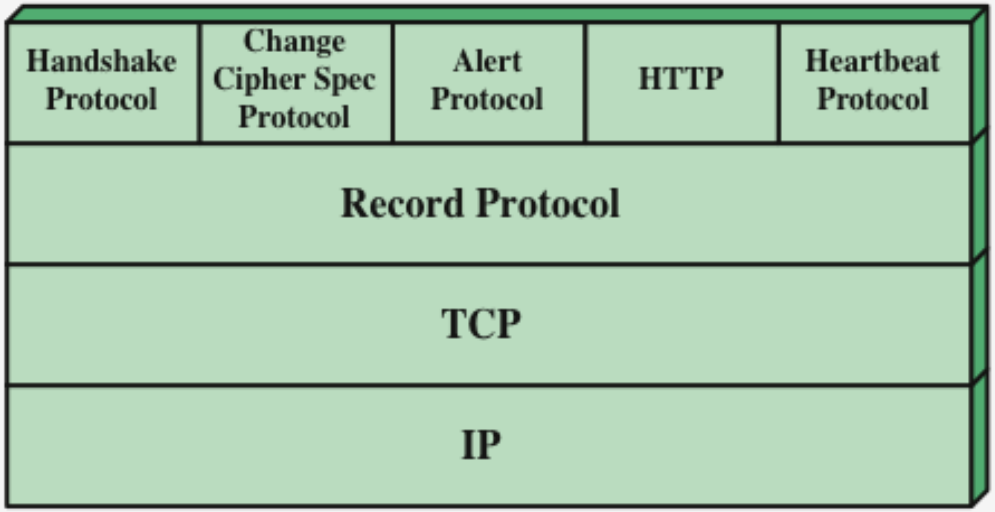
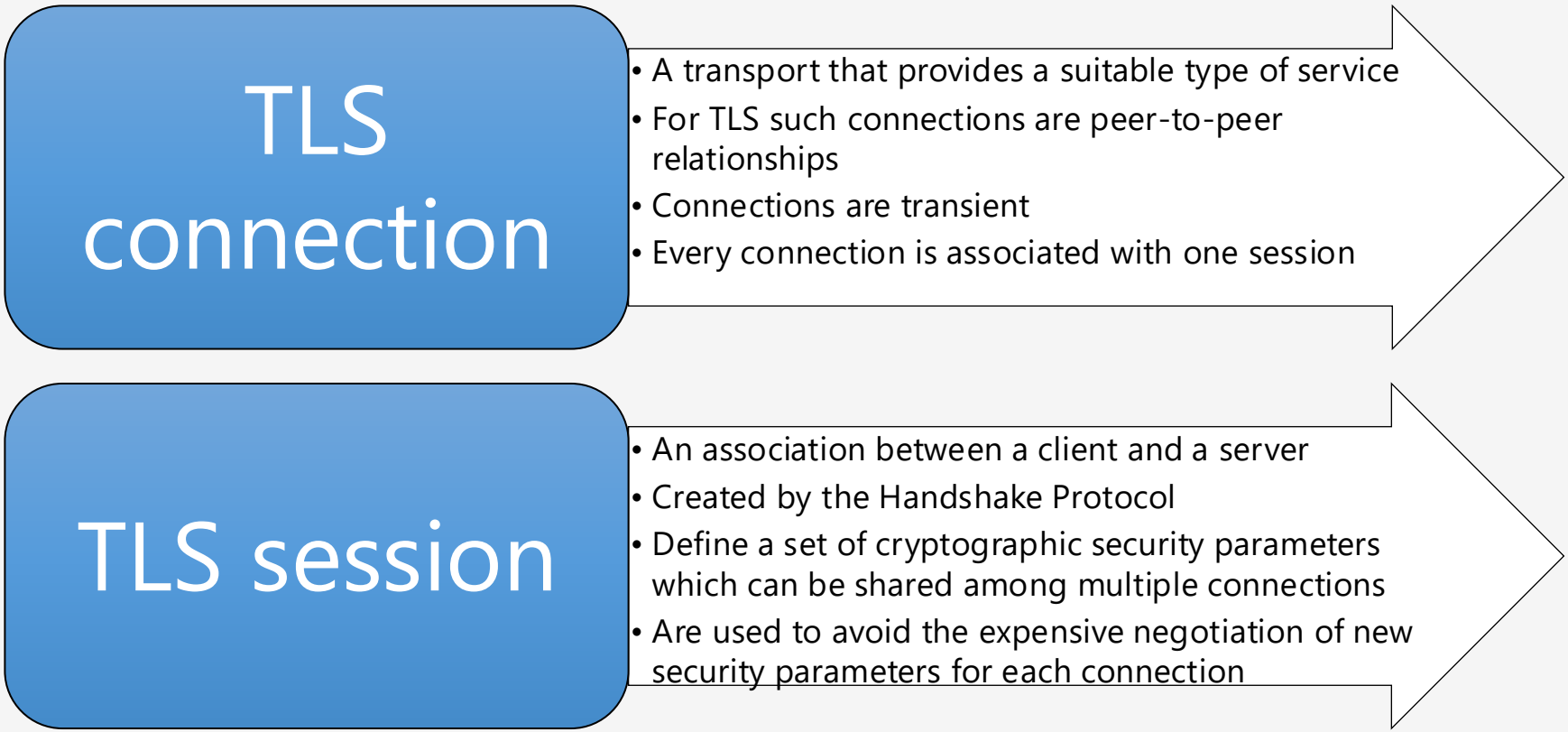


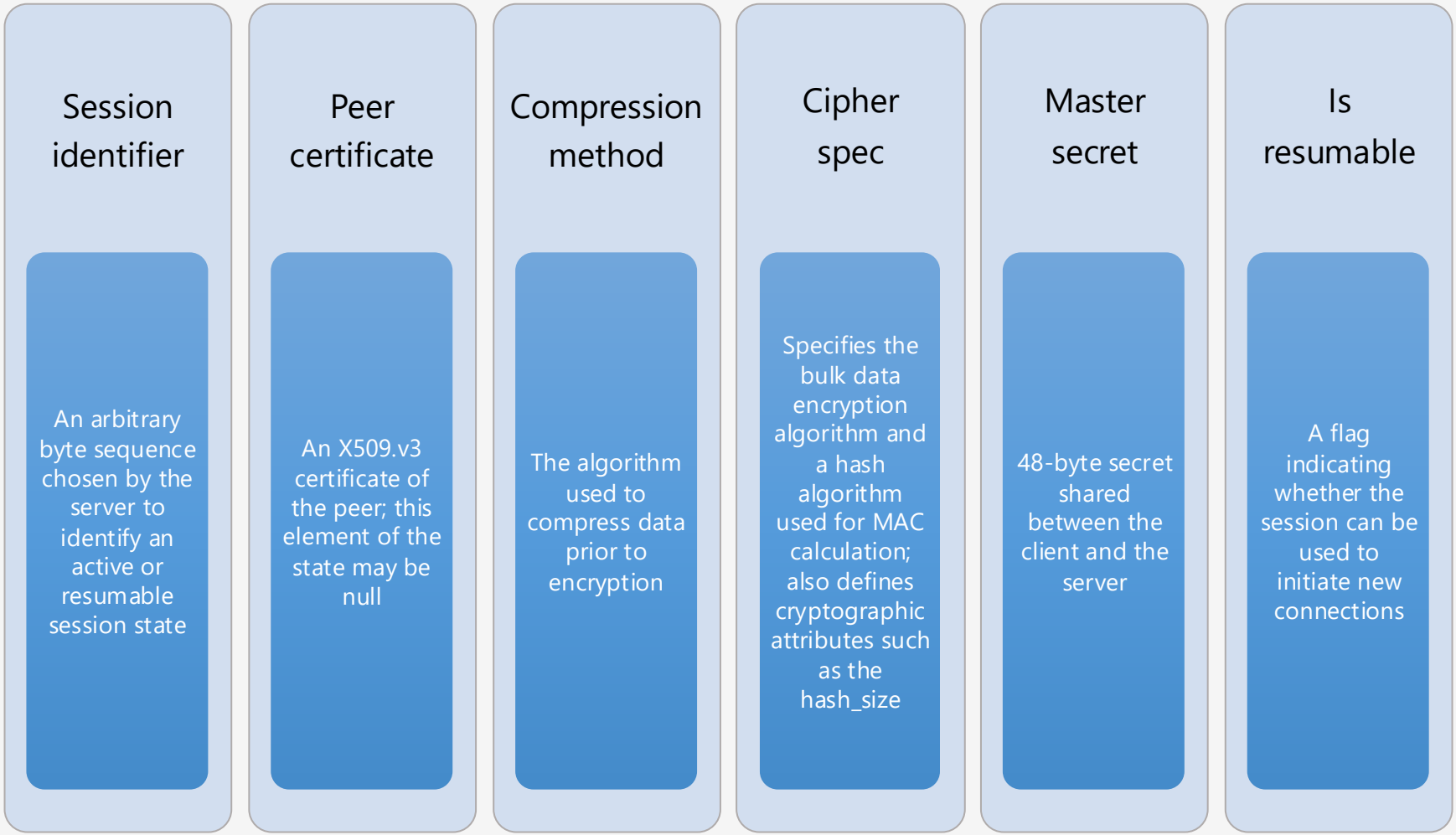
Figure 17.2 SSL/TLS Protocol Stack

TLS Architecture

- Two important TLS concepts are:



A **Session State** is defined by the following parameters:

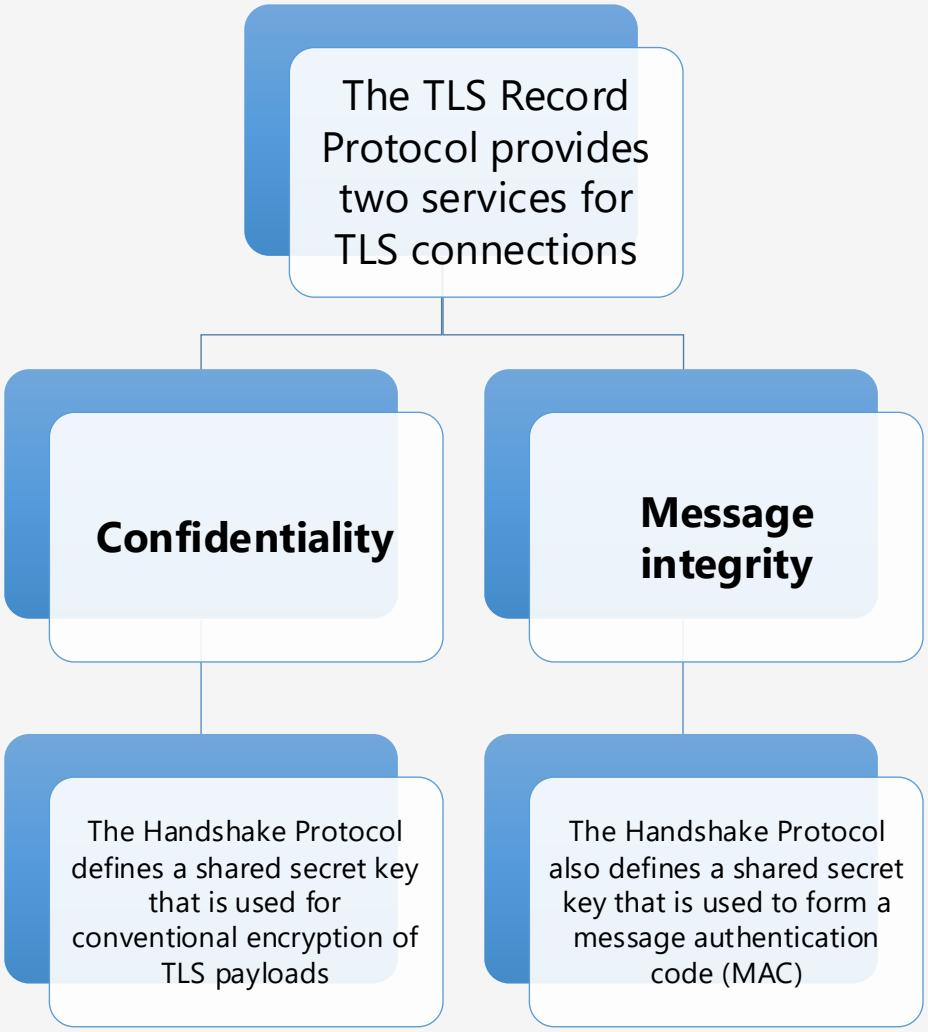


A **Connection State** is defined by the following parameters:

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 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



TLS Record Protocol Operation

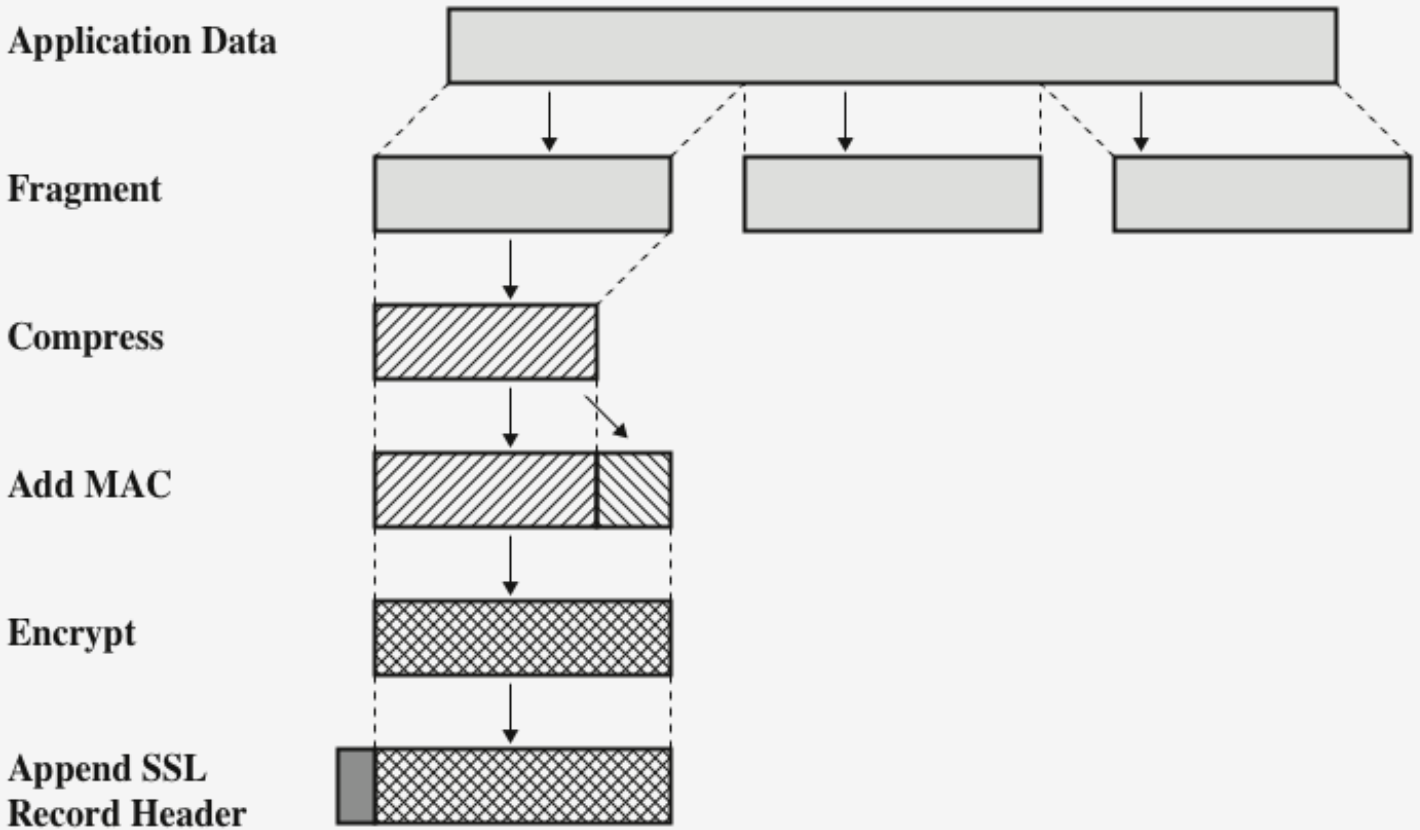


Figure 17.3 SSL Record Protocol Operation

TLS Record Format

- Content Type (8 bits): The higher-layer protocol used to process the fragment
- Major and Minor Version: The version of TLS used
- Compressed Length (16 bits): The length in bytes of the plaintext fragment (possibly compressed)

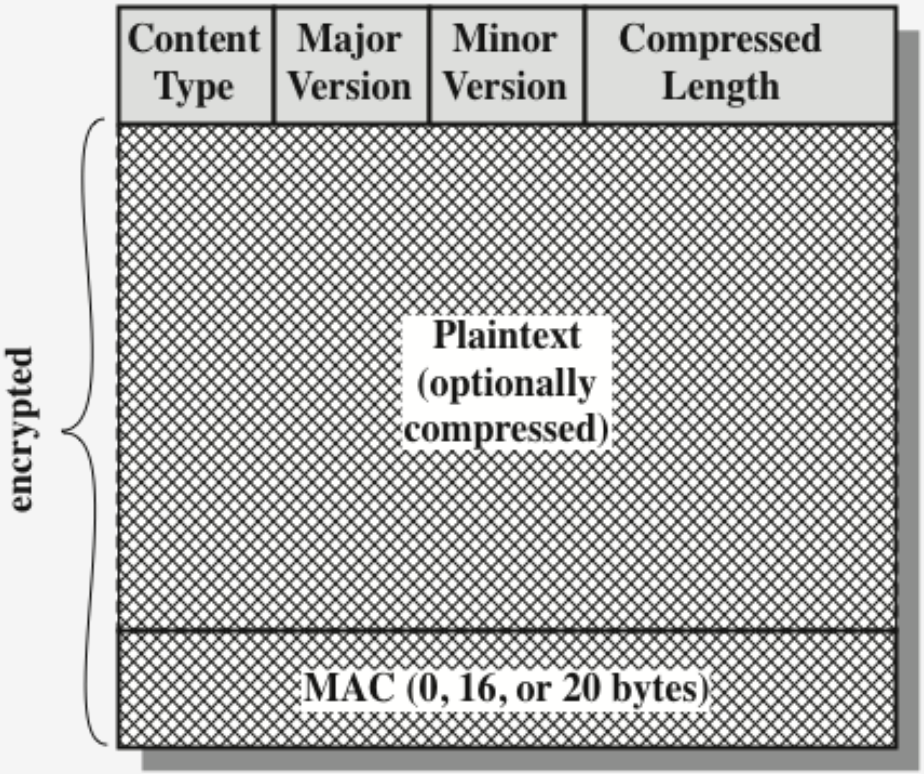


Figure 17.4 SSL Record Format

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

TLS Handshake Protocol in action

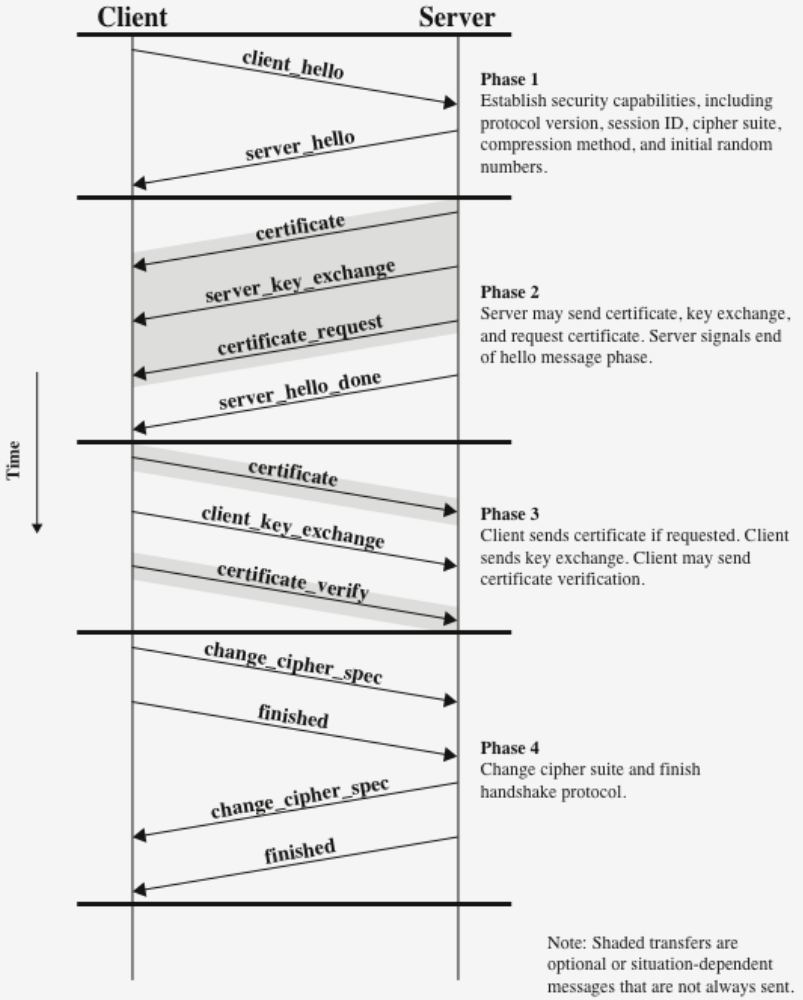


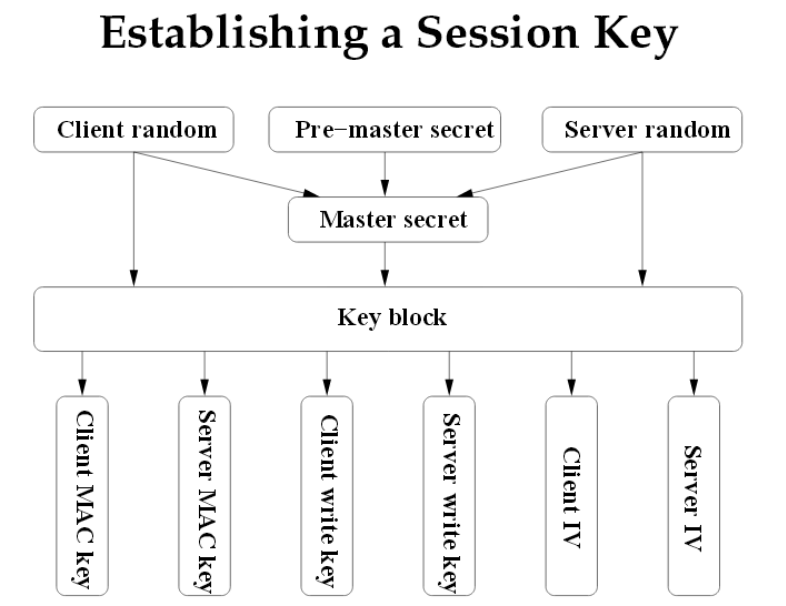
Figure 17.6 Handshake Protocol Action

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…

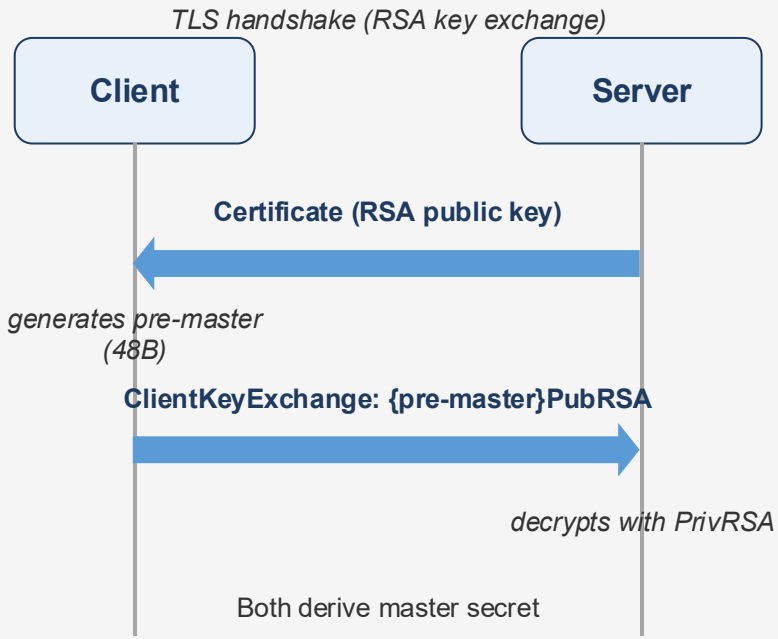
✓ These parameters are generated from the master secret by hashing the master secret into a sequence of secure bytes of sufficient length for all needed parameters

Pre-master secret via RSA

Pre-master generation & exchange

1. Server sends X.509 certificate with its RSA public key.
2. Client validates the certificate (CA chain, validity, hostname).
3. Client randomly generates the pre-master secret (48 bytes).
4. Client encrypts the pre-master with the server's RSA public key.
5. Client sends it in the ClientKeyExchange message.
6. Server decrypts with its RSA private key and recovers the pre-master.
7. Both derive the master secret from pre-master + ClientHello.random + ServerHello.random.

Only the holder of the server's private key can recover the pre-master — implicitly authenticates the server.



How Diffie-Hellman works

Two parties agree on a shared secret over a public channel — without ever transmitting the secret itself.

The idea

- Public parameters: a large prime p and a generator g (everyone knows them).
- Each side picks a private number and shares only the result of mixing it with g.
- Mixing privates the same way on both ends gives the same shared secret.

Steps

- Alice picks private a → sends A = g^a mod p
- Bob picks private b → sends B = g^b mod p
- Alice computes B^a mod p; Bob computes A^b mod p
- **Both get the same value: g^(ab) mod p ← shared secret**

Tiny example (p=23, g=5)

```
Alice's private:  a = 6
Bob's private:    b = 15

A = 5^6 mod 23  = 8
B = 5^15 mod 23 = 19
```

Alice sends 8, Bob sends 19 (over public channel)

```
Alice: 19^6  mod 23 = 2
Bob:    8^15 mod 23 = 2
```

Shared secret = 2

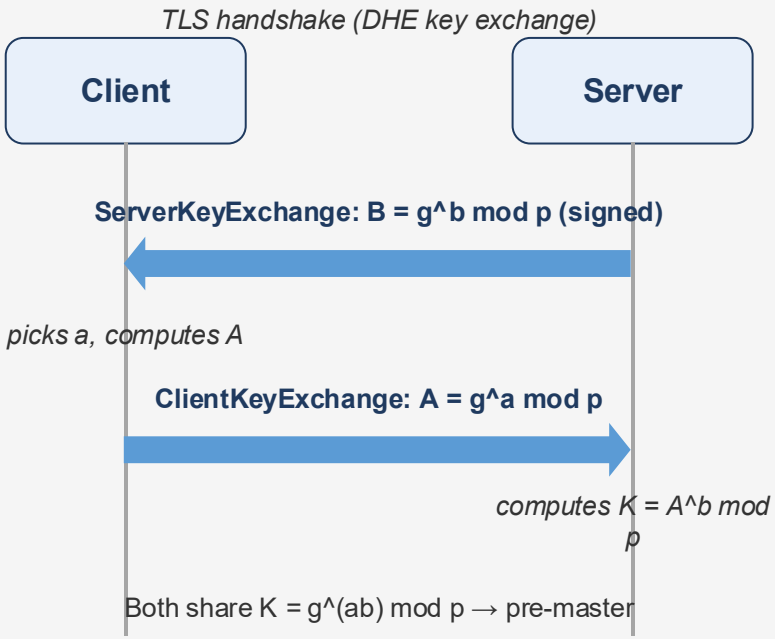
Eavesdropper sees p, g, A, B — but recovering a from A = g^a mod p or b B = g^b mod p from (the discrete log problem) is computationally infeasible for large p.

Pre-master secret via Diffie-Hellman

Pre-master generation & exchange

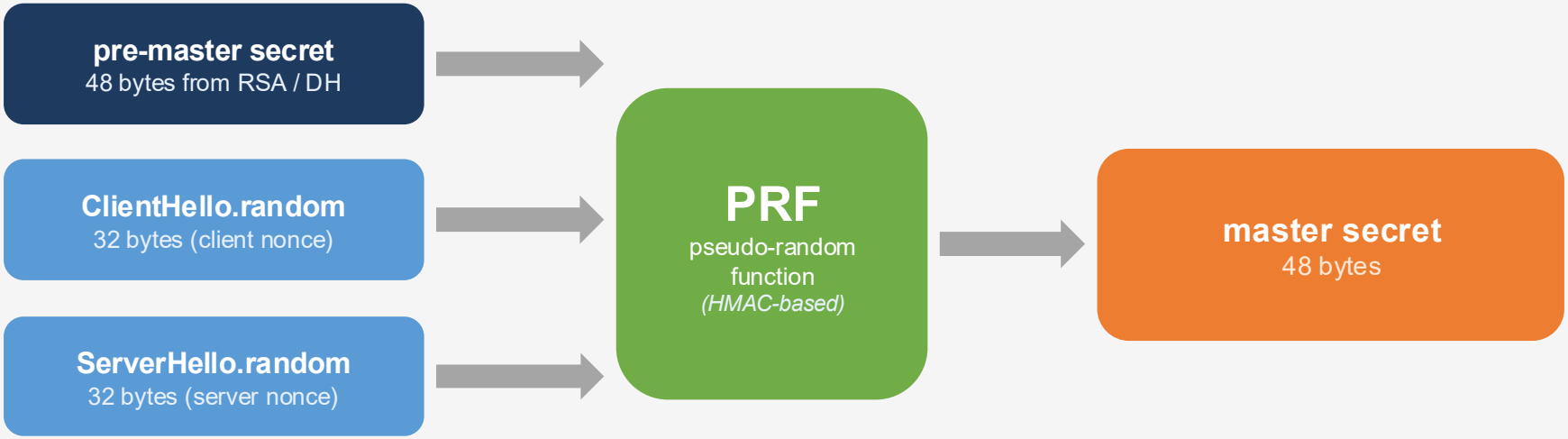
1. Server chooses DH parameters: large prime p and generator g.
2. Server picks private b, computes B = g^b mod p; sends B together with p and g (all signed with its RSA/DSA key for authentication)
3. Client validates the server's signature using the certificate's public key.
4. Client picks private a, computes A = g^a mod p; sends A in ClientKeyExchange.
5. Each side computes the shared secret: client = B^a mod p, server = A^b mod p — both equal g^(ab) mod p.
6. That shared value is the pre-master secret.
7. Both derive the master secret from pre-master + ClientHello.random + ServerHello.random.

Ephemeral DH (DHE) gives forward secrecy: a/b are discarded after the handshake — even if the server's long-term key leaks later, past sessions stay safe.



From pre-master to master secret

After the key exchange, both sides combine three values to derive the master secret — a 48-byte symmetric key used to generate all session keys.




```
master_secret = PRF(pre_master,  "master secret",  ClientHello.random || ServerHello.random)
```

Why mix in the randoms?

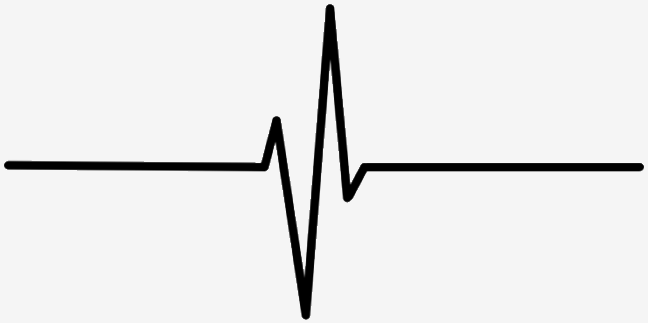
- Each side contributes fresh entropy → unique master per session.
- Replaying an old handshake gives a different master.
- Even if pre-master repeated, master would still differ.

What the master is used for

- Run PRF again with master + randoms to derive:
- client/server MAC keys, encryption keys, IVs.
- Master itself is never used directly to encrypt.

Heartbeat Protocol

- Is a periodic signal generated by hardware or software to indicate normal operation or to synchronize other parts of a system
- Typically used to monitor the availability of a protocol entity
- In the specific case of TLS, a Heartbeat protocol was defined in 2012 in RFC 6250 (Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS) Heartbeat Extension)



HTTPS (HTTP over SSL)

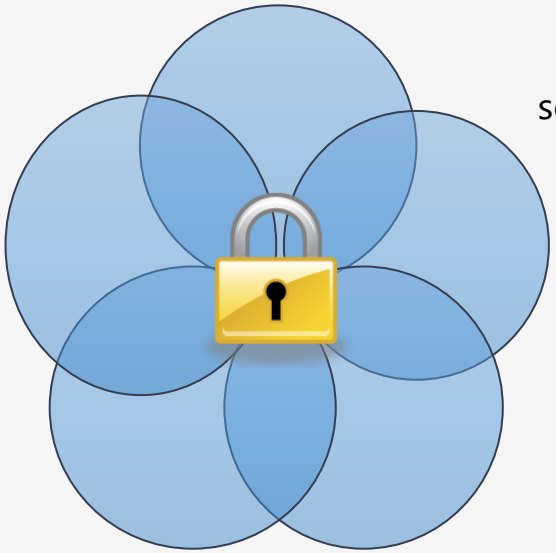
- Refers to the combination of HTTP and SSL to implement secure communication between a Web browser and a Web server
- The HTTPS capability is built into all modern Web browsers
- A user of a Web browser will see URL addresses that begin with https:// rather than http://
- If HTTPS is specified, port 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

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 and 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

SSH Transport Layer Packet Exchanges

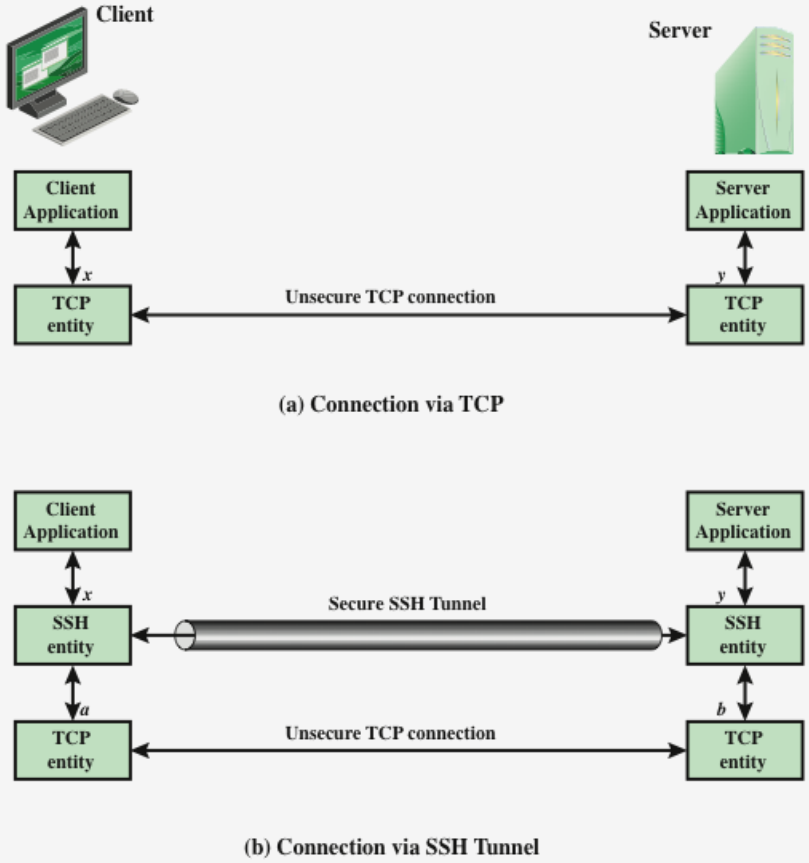


Figure 17.12 SSH Transport Layer Packet Exchanges

SSH Protocol Stack

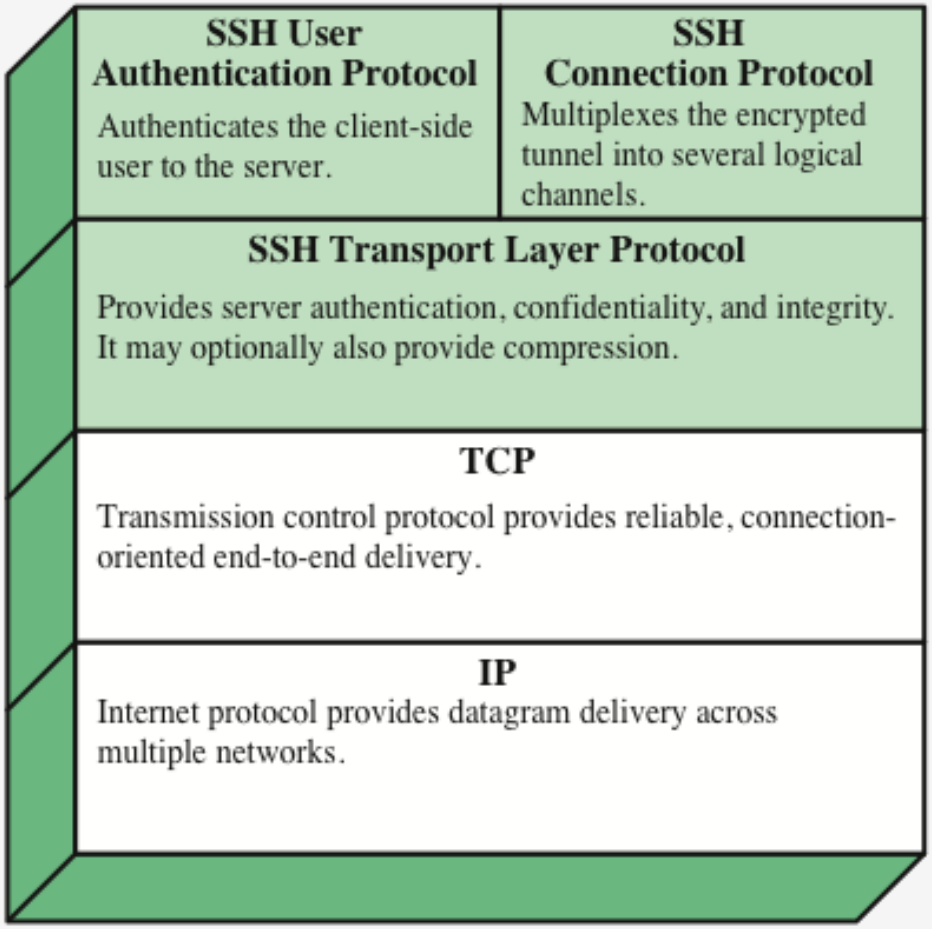


Figure 17.8 SSH Protocol Stack

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

Transport Layer Protocol Packet Formation

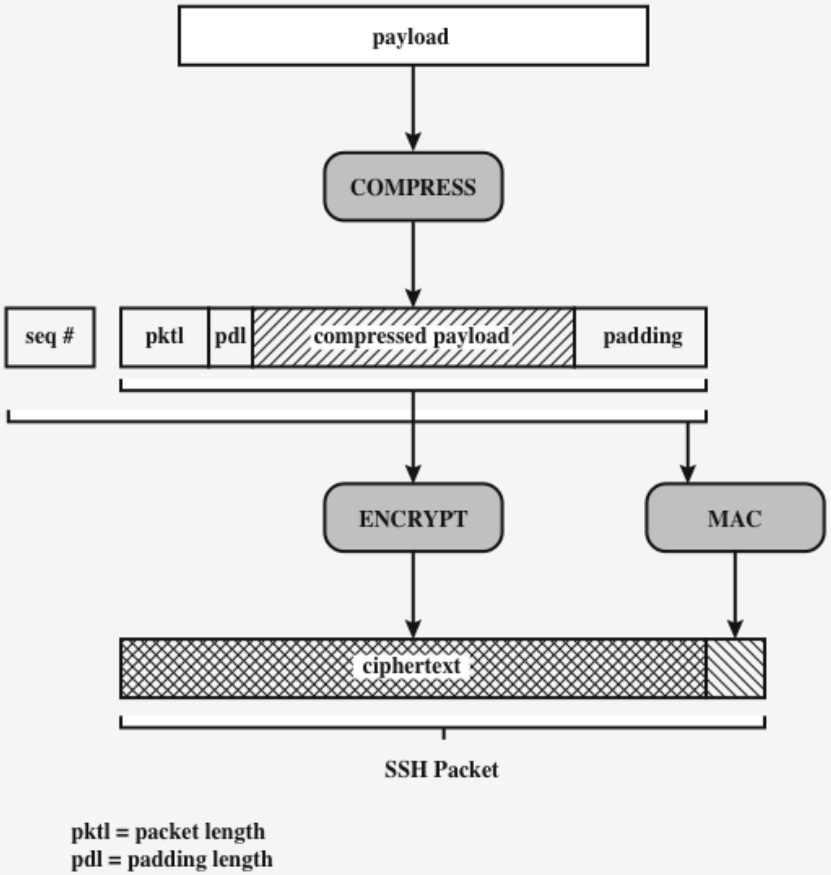


Figure 17.10 SSH Transport Layer Protocol Packet Formation

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

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

Authentication Methods

- Public-key
  - ✓ 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
- Host-based
  - ✓ Authentication is performed on the client's host rather than the client (user) 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

Connection Protocol Message Exchange

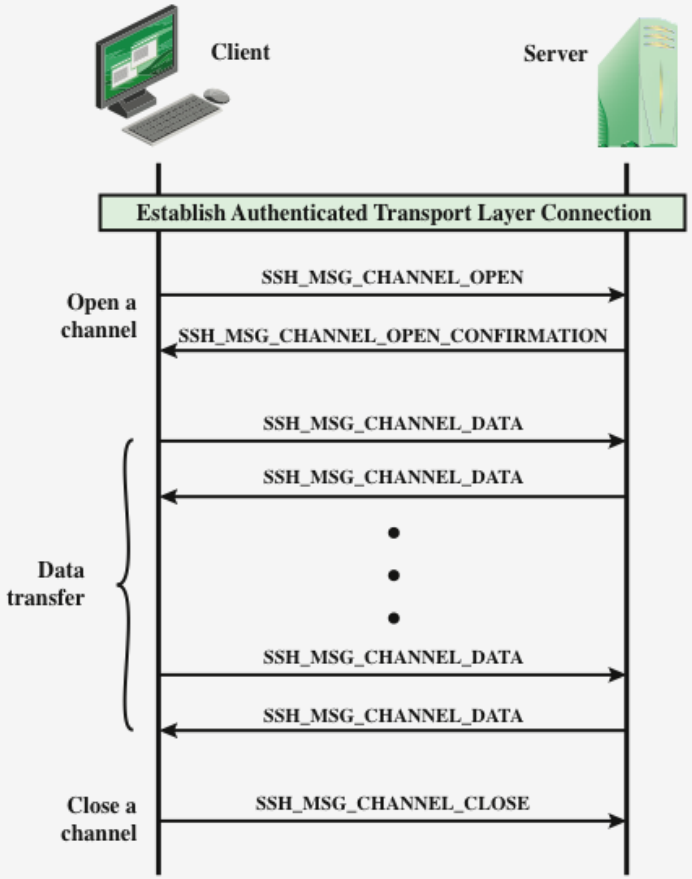


Figure 17.11 Example SSH Connection Protocol Message Exchange

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

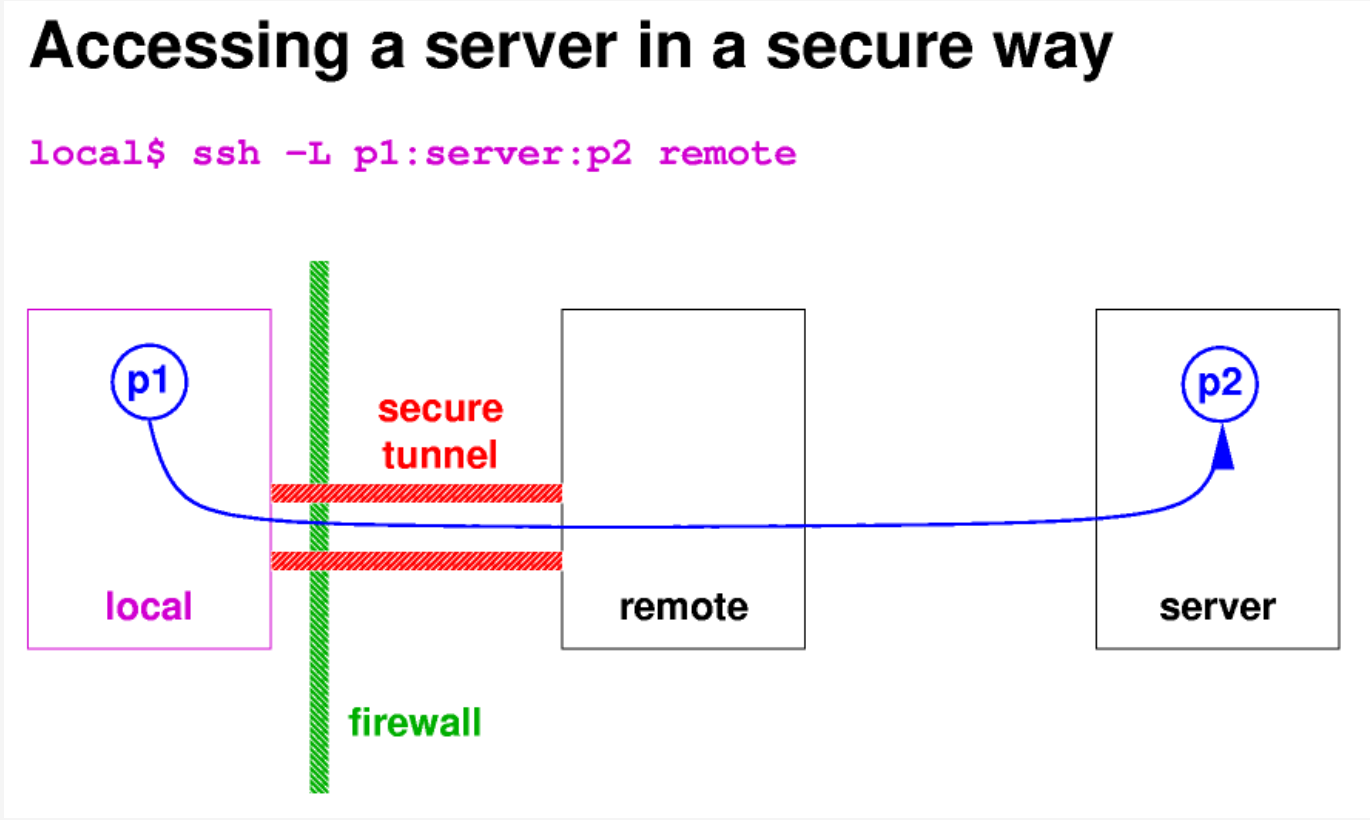
Direct-tcpip

- Local port forwarding

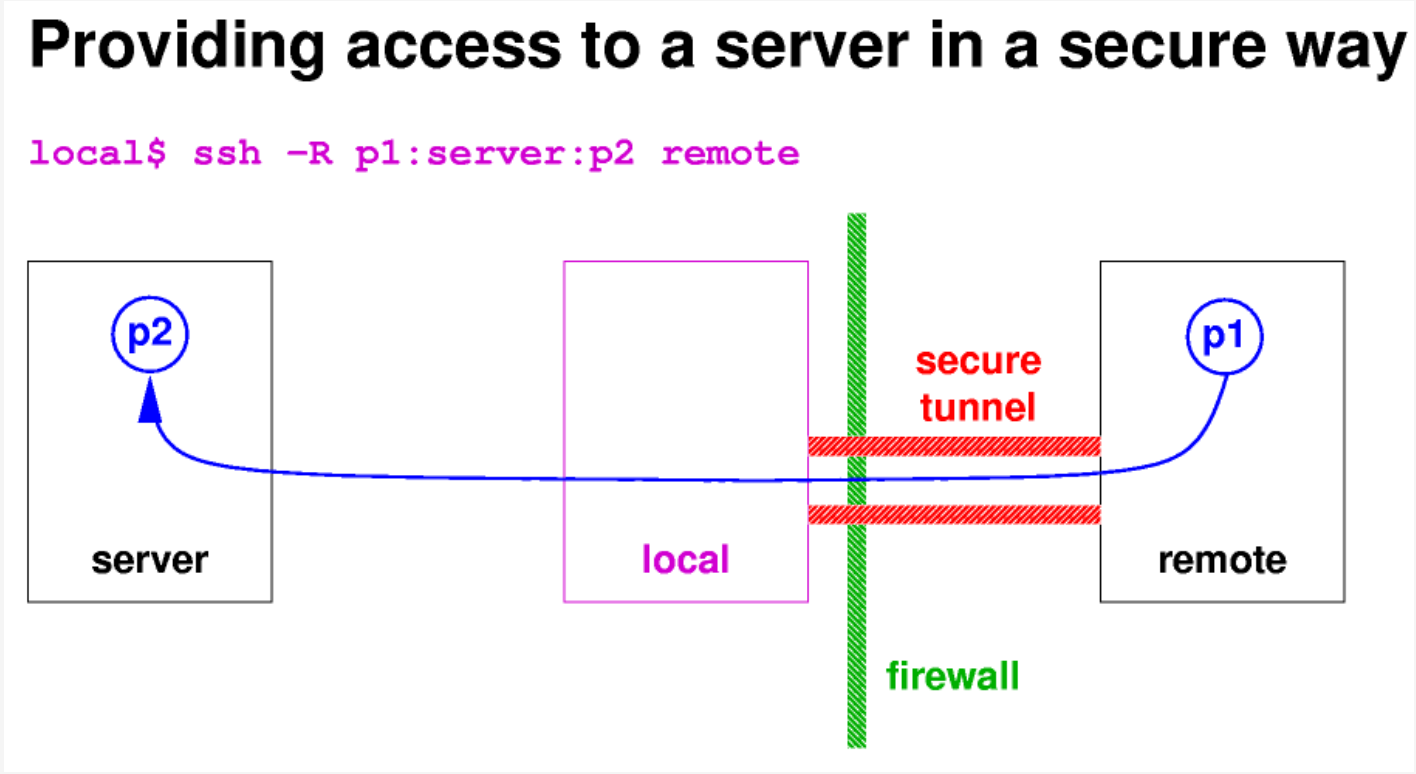
Port Forwarding

- ✓ 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
- ✓ Local and remote Port Forwarding over SSH tunnel

Port Forwarding (local)



Port Forwarding (remote)

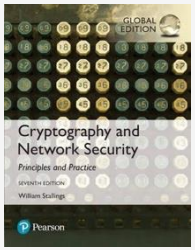


Summary

- Web security considerations
  - ✓ Web security threats
  - ✓ Web traffic security approaches
- Secure sockets layer
  - ✓ SSL architecture
  - ✓ SSL record protocol
  - ✓ Change cipher spec protocol
  - ✓ Alert protocol
  - ✓ Handshake protocol
  - ✓ Cryptographic computations
  - ✓ Heartbeat protocol
  - ✓ TLSv1.3
- Secure shell (SSH)
  - ✓ Transport layer protocol
  - ✓ User authentication protocol
  - ✓ Communication protocol
- HTTPS
  - ✓ Connection initiation
  - ✓ Connection closure

Bibliography

Cryptography and network security, Stallings, Pearson, 2017, Chapter 17: Web and Transport-Layer Security



Segurança Prática em Sistemas e Redes com Linux, Capítulo 3: Autoridades de certificação digital e Capítulo 4: Ligações seguras com SSH