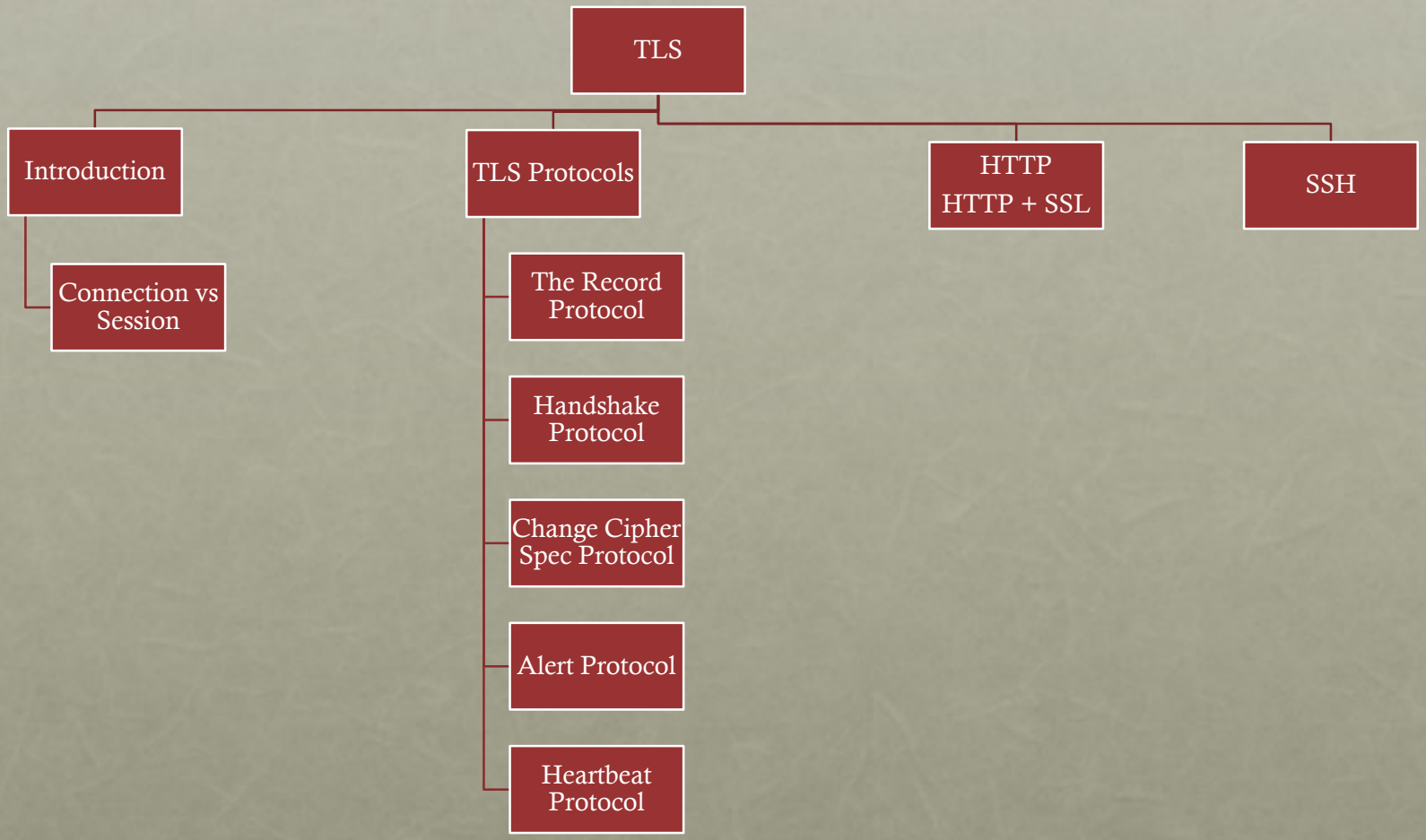
CHAPTER 6

Transport-Level Security

TLS


TLS
Introduction
Connection vs Session
TLS Protocols
The Record Protocol
Handshake Protocol
Change Cipher Spec Protocol
Alert Protocol
Heartbeat Protocol
HTTP
HTTP + SSL
SSH

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

TRANSPORT LAYER SECURITY (TLS)

- One of the most widely used security services
- TLS is an Internet standard that evolved from a commercial protocol known as *Secure Sockets Layer* (SSL)
- TLS is a general purpose service implemented as a set of protocols that rely on TCP
  - TLS either provided as part of the underlying protocol suite and therefore be transparent to applications
  - Or, TLS can be embedded in specific packages

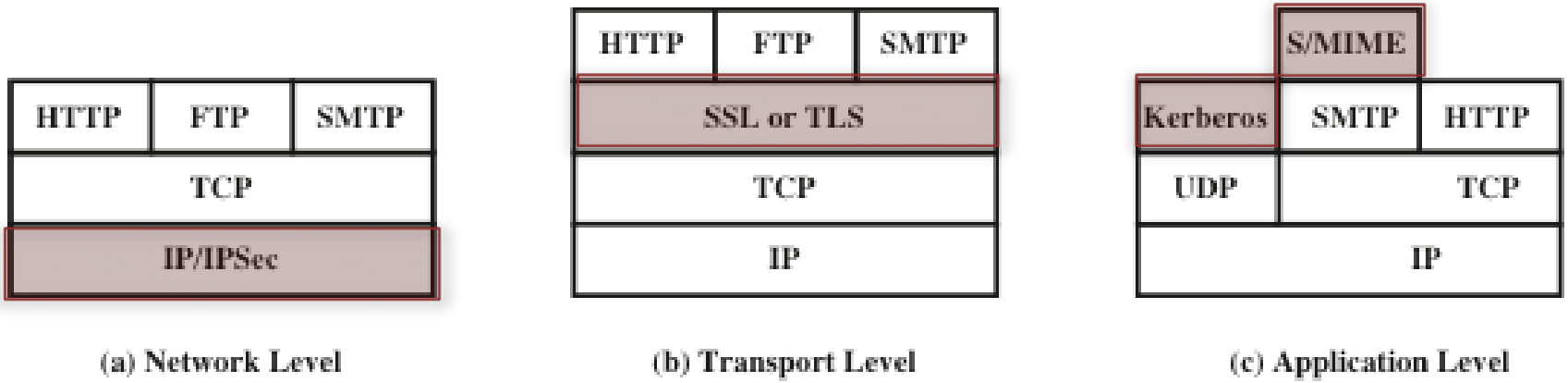


Figure 6.1 Relative Location of Security Facilities in the TCP/IP Protocol Stack

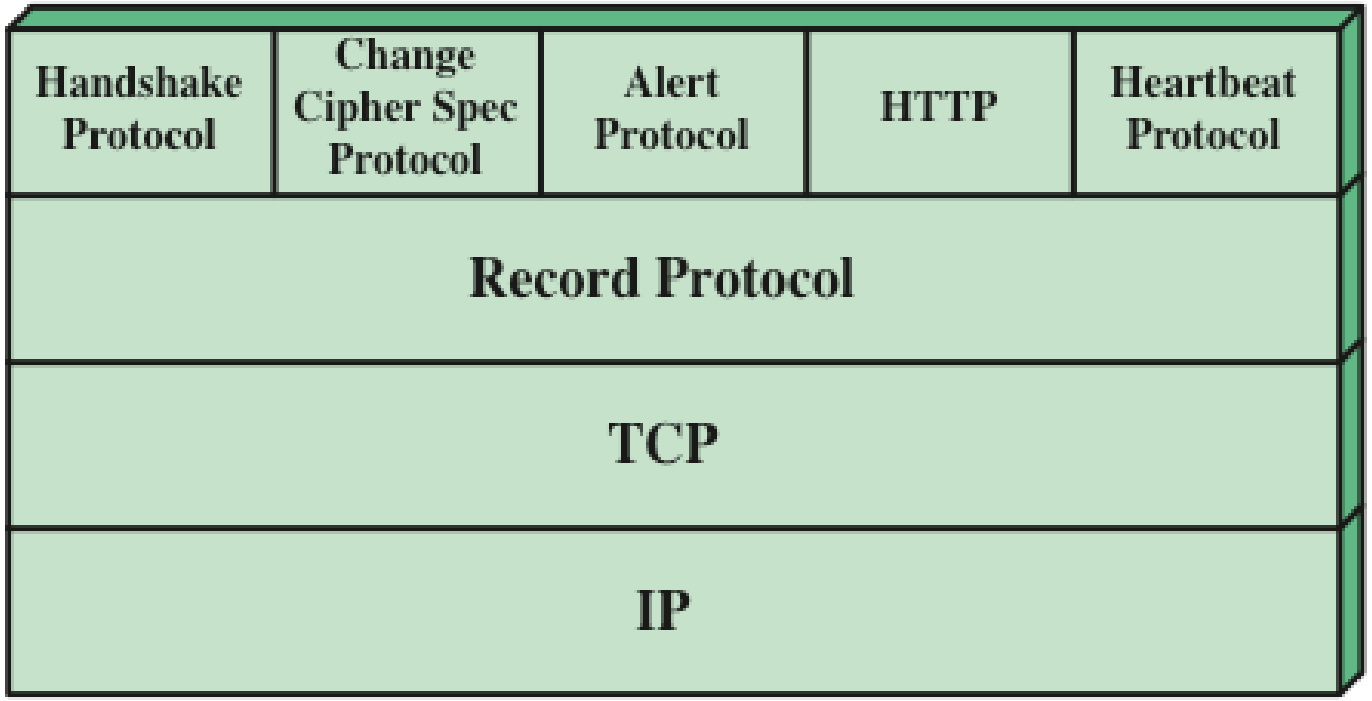


Figure 6.2 SSL/TLS Protocol Stack

TLS ARCHITECTURE

- Two important TLS concepts are:

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**

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**

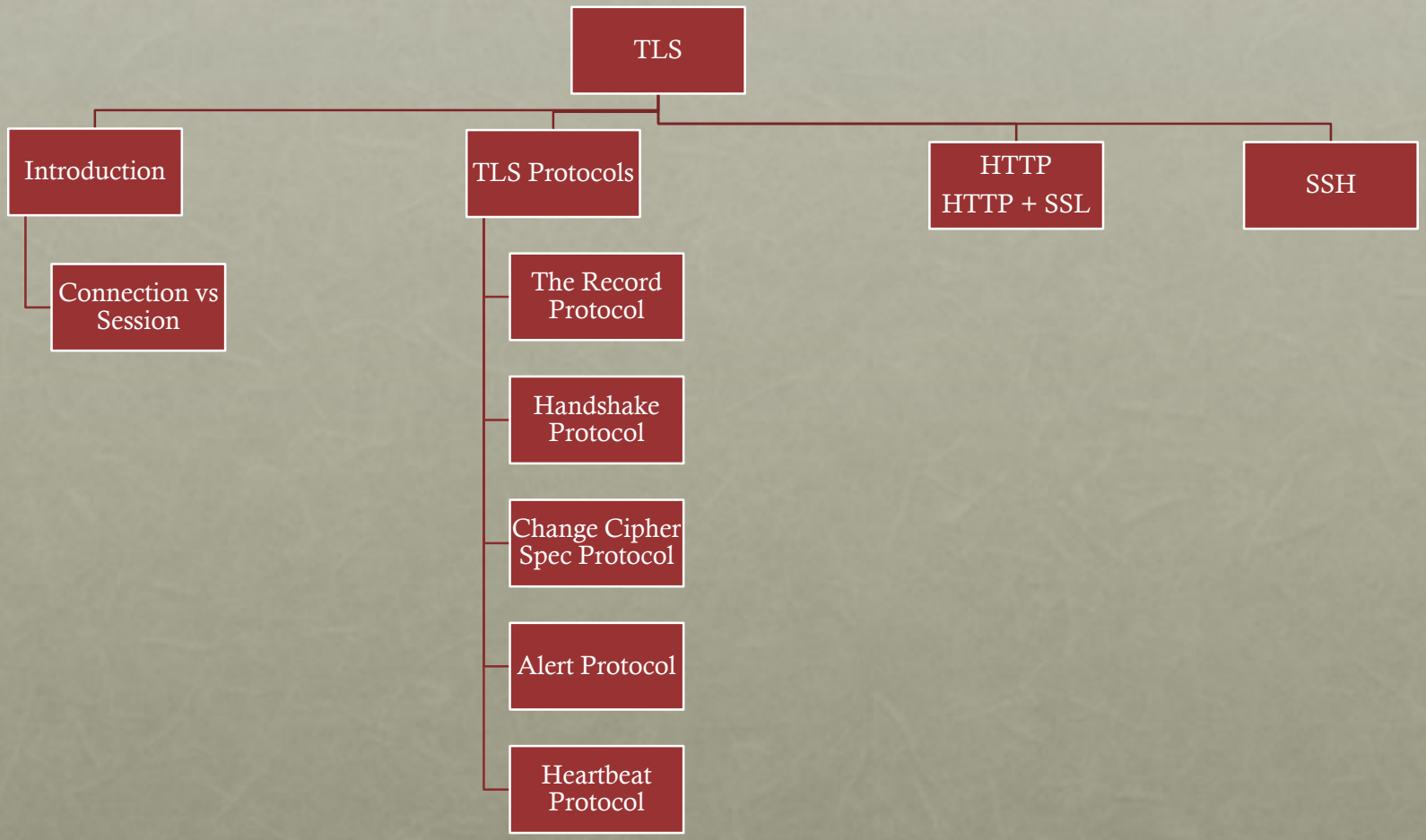
A SESSION STATE IS DEFINED BY THE FOLLOWING PARAMETERS:

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

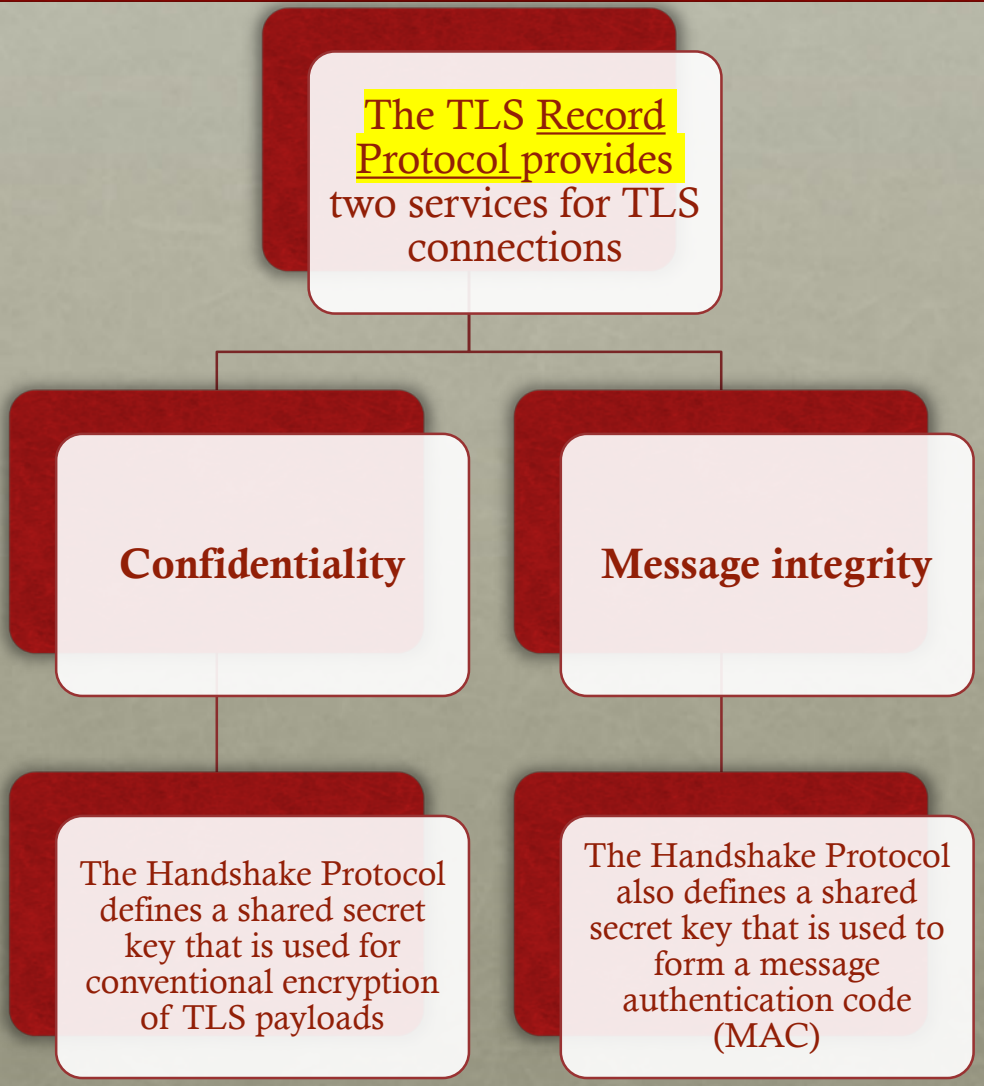
A CONNECTION STATE IS DEFINED BY THE FOLLOWING PARAMETERS:

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 $2^{64} - 1$

TLS


TLS
Introduction
Connection vs Session
TLS Protocols
The Record Protocol
Handshake Protocol
Change Cipher Spec Protocol
Alert Protocol
Heartbeat Protocol
HTTP
HTTP + SSL
SSH

TLS RECORD PROTOCOL



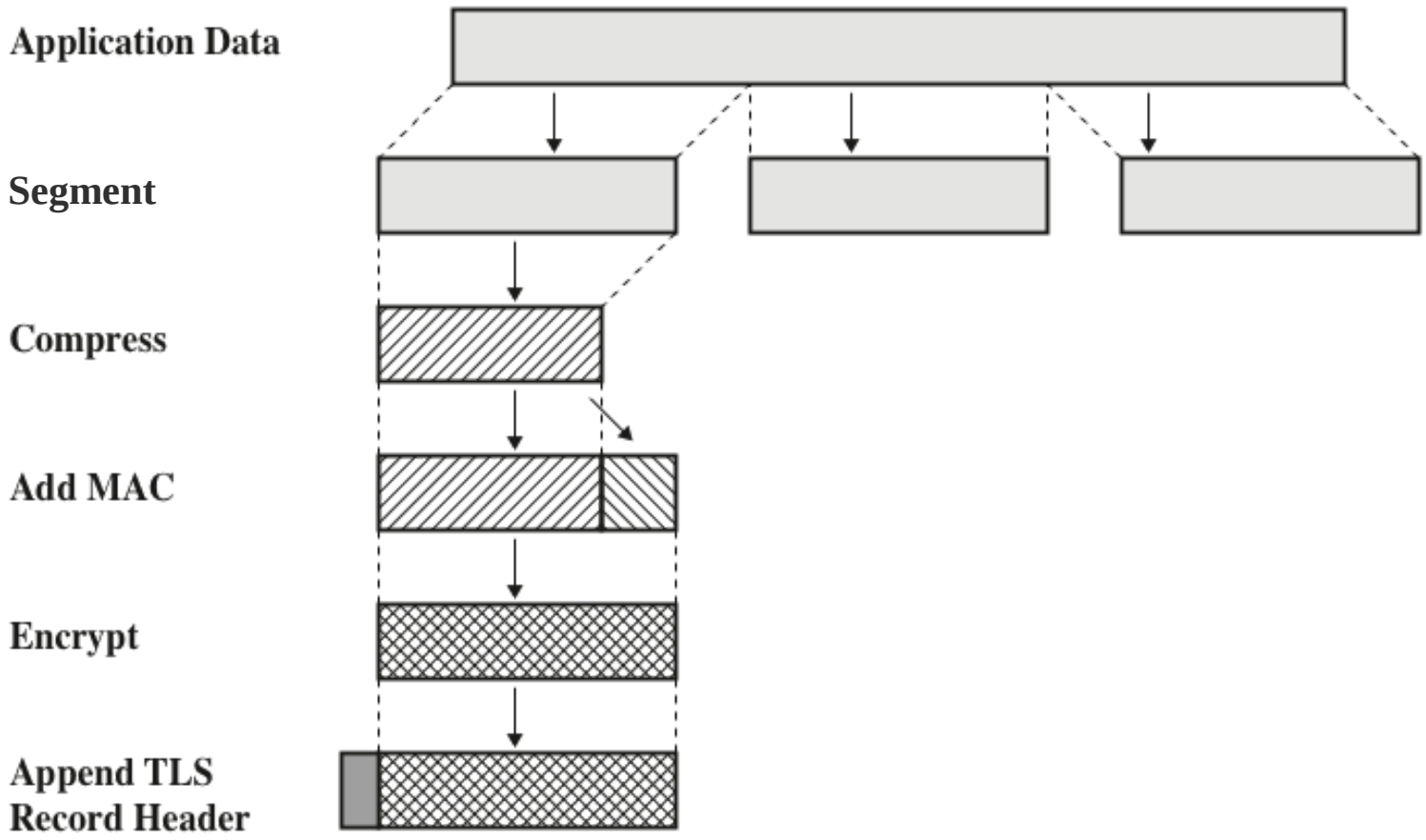


Figure 6.3 TLS Record Protocol Operation

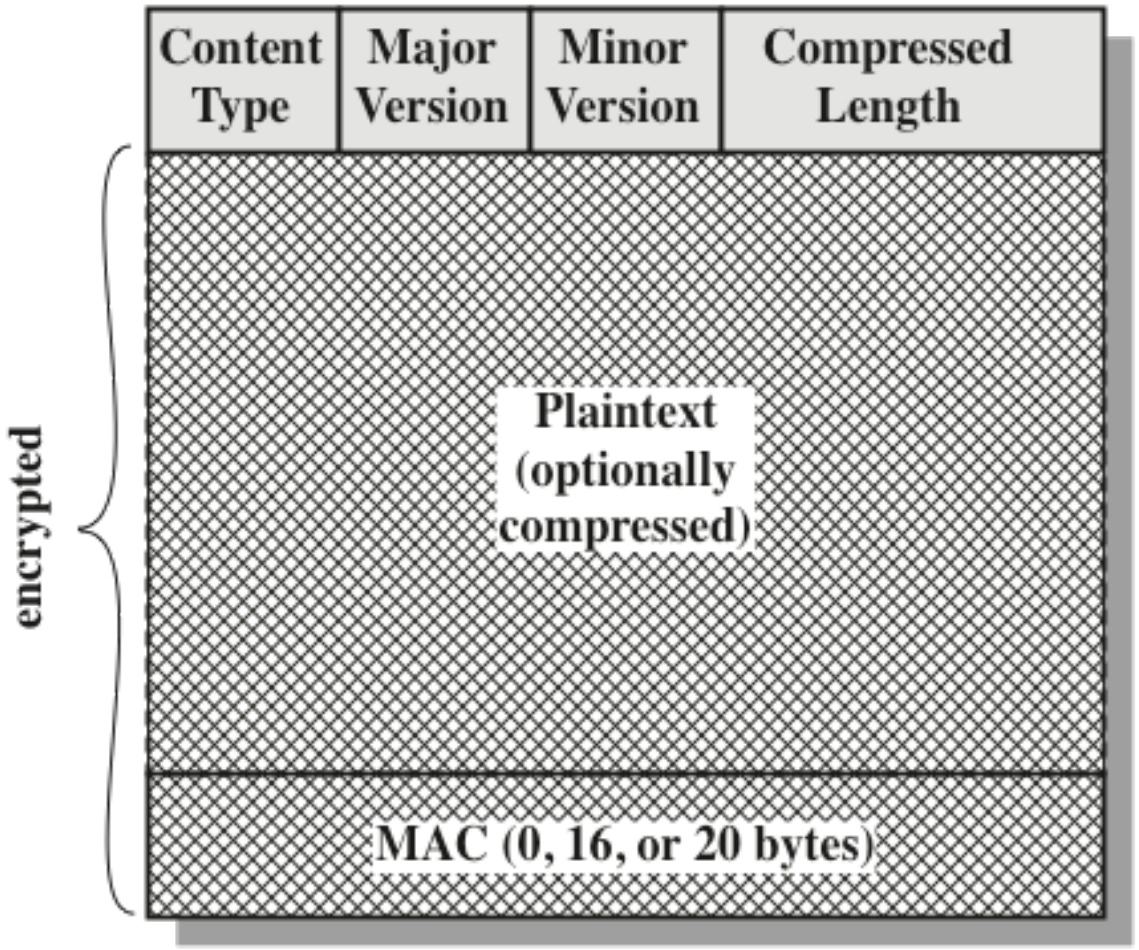


Figure 6.4 SSL Record Format

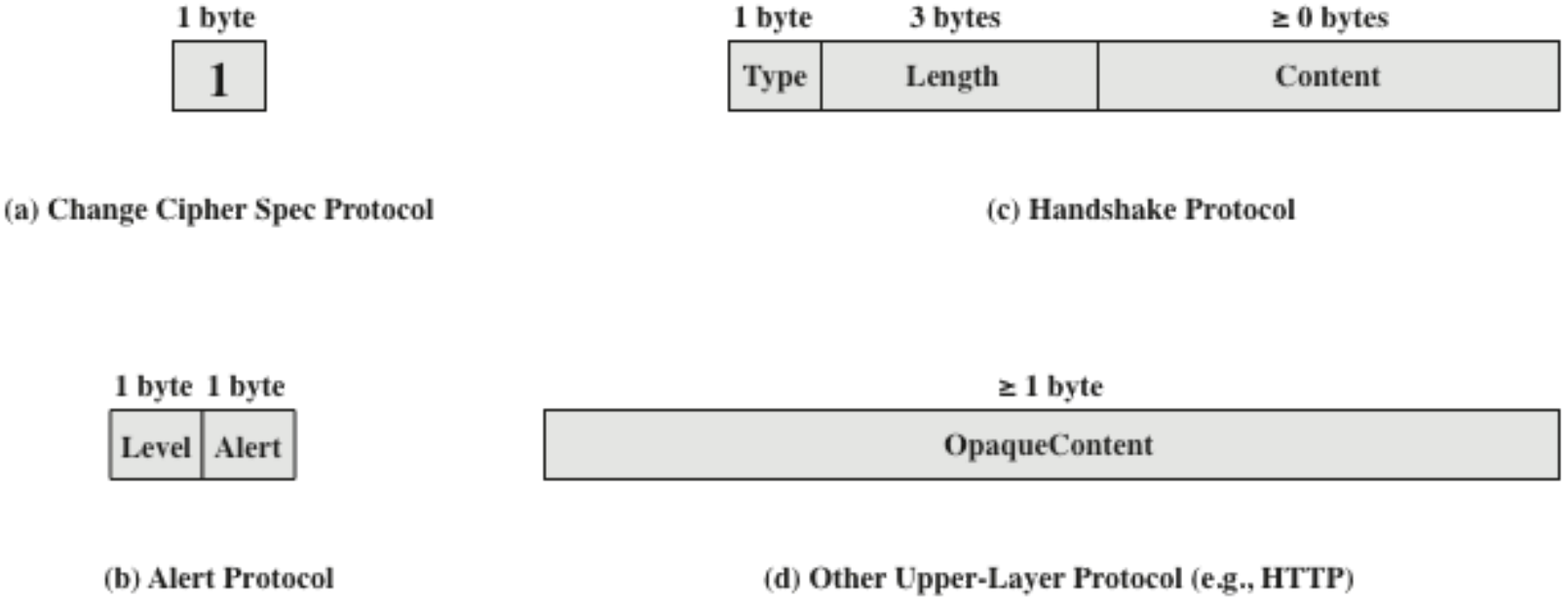


Figure 6.5 TLS Record Protocol Payload

TLS PROTOCOLS

- **The Change Cipher Spec Protocol:** it is the simplest, consists of a single message, 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 spec to be used on this connection.

- **The Alert Protocol:** used to convey TLS-related alerts to the peer entity. Each message consists of two bytes. The first byte takes the value warning (1) or fatal (2) to convey the severity of the message. If the level is fatal, TLS 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.

- **The Handshake Protocol:** 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 a TLS record. The Handshake Protocol is used before any application data is transmitted. The Handshake Protocol messages have the format shown in the following slide

HEARTBEAT PROTOCOL

- A heartbeat is a periodic signal generated by hardware or software to indicate normal operation or to synchronize other parts of a system
- A heartbeat protocol is typically used to monitor the availability of a protocol entity
- Was first designed to work with DTLS (connectionless)
- The heartbeat serves Four purposes:

  (1) It assures the sender that the recipient is still alive, even though there may not have been any activity over the underlying TCP connection

  (2) It generates activity across the connection during idle periods, which avoids closure by a firewall that does not tolerate idle connections

  (3) Synchronization

HEARTBEAT PROTOCOL

- The heartbeat protocol runs on top of the TLS Record Protocol
  - Consists of two message types: heartbeat request and heartbeat response.
  - The heartbeat request message includes payload length, payload, and padding fields.
  - The payload is a random content between 16 bytes and 64 Kbytes in length.
  - The corresponding heartbeat response message must include an exact copy of the received payload.

(4) The padding enables the sender to perform a path MTU (maximum transfer unit) discovery operation, by sending requests with increasing padding until there is no answer anymore because one of the hosts on the path cannot handle the message.

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 TLS Handshake Protocol Message Types

The Handshake Protocol

C→ S: client_hello
- Max TLS **supported** version
- Nonce, Session ID
- List of **supported** cipher suites and compression methods

S→ C: server_hello
- **Chosen** TLS version
- **Chosen** Cipher **suite** and compression method
- Nonce

Phase 1

S→C: Certificate (user authentication)
- Includes a Public Key

S→C: server_key_exchange
- Exchange of cipher **specs**
- This msg is hashed and signed with the sender's private key

S→C: certificate_request

S→C: certificate_hello_done

Phase 2

C→S: Certificate (user authentication)
- Includes a Public Key

C→S: client_key_exchange
- Diffie-Hellman parameters
- This msg is hashed and signed with the sender's private key

C→S: certificate verify

Phase 3

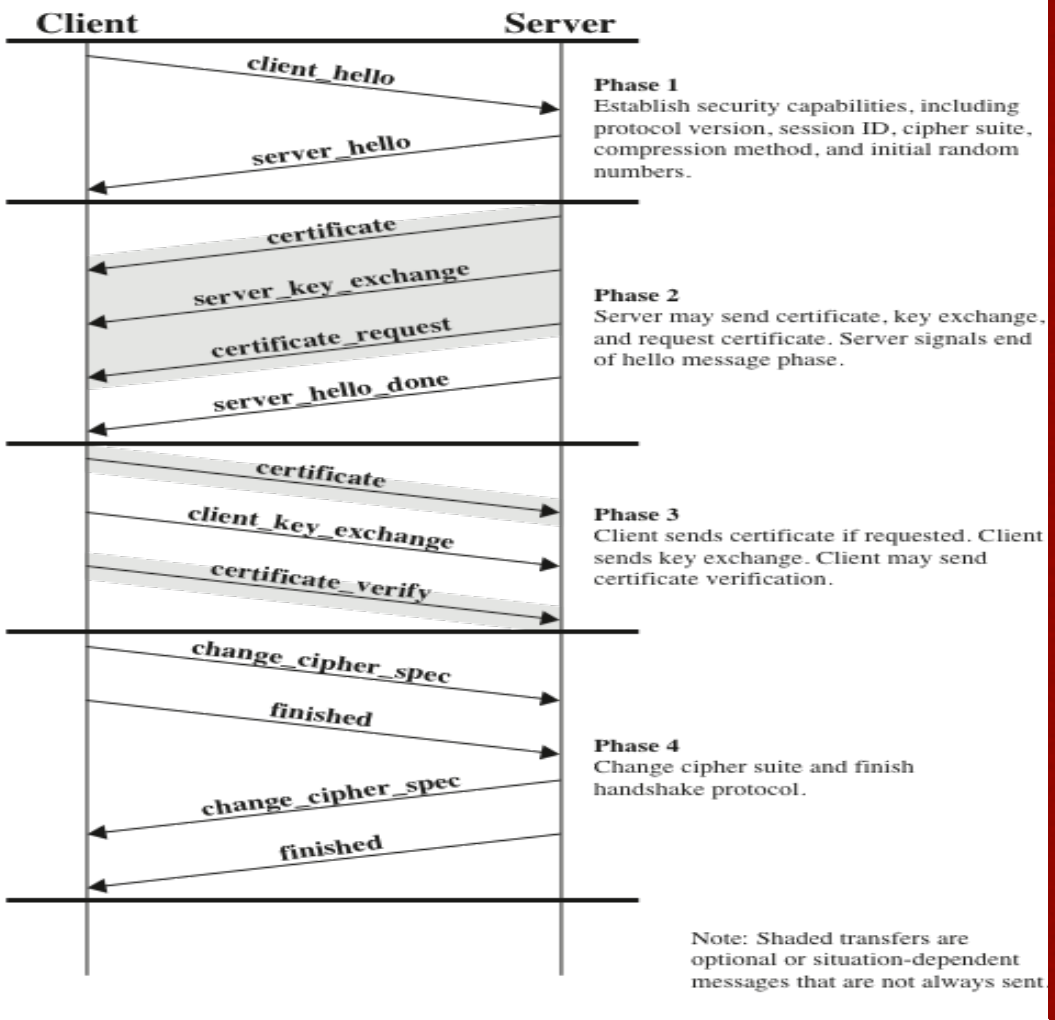
C→S: Change_Cipher_Spec
- Just a flag → now is the time to make the transition to encrypted communication

C:S →Finished
- First encrypted message

S→C: Change_Cipher_Spec
- Just a flag → now is the time to make the transition to encrypted communication

S:C →Finished
- First encrypted message

Phase 4



CRYPTOGRAPHIC COMPUTATIONS

- Two further items are of interest:
  - The creation of a 1) pre-master key by means of the key exchange 2) shared master secret using the pre-master key. The shared master secret is a one-time 48-byte value generated for this session
    - Master Key = PRF(pre-master key, S nonce, C nonce)
  - 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
      - Key block = PRF(Master Key, S nonce, C nonce)

IKEv2 Exchange

Initial Exchange:

- **First Message:** I sends to R (IKE_SA_INIT), which contains:
  - offered encryption and message integrity algorithms,
  - Diffie-Hellman public Keys,
  - and nonces.
- **Second Message:** R responds to I with (IKE_SA_INIT),
  - offered encryption and message integrity algorithms,
  - Diffie-Hellman Keys,
  - nonces, and
  - **Certificate request**
- **NOTE,**
  - by the end of this step, both I and R have created a secret key and all following messages are encrypted, and
  - **ONE SA (IKE SA) FOR CONTROL MESSAGES HAS BEEN ESTABLISHED**
- **Third and Fourth Messages (IKE_AUTH):**
  - Initiators and responder's identities, certificates (optional), SA parameters are exchanged
  - Validate the identity
- **NOTE,**
  - by the end of these two steps, users have been authenticated, and
  - **ONE SA (IPSEC SA) FOR communication between the peers**

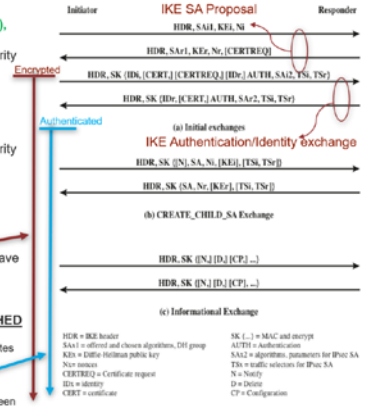


Figure 9.11 IKEv2 Exchanges

IKEv2 Exchange

Create Child SA :

- Used to create new additional child SA to using a new tunnel.
- New Diffie-Hellman values and message authentication parameters are exchanged
- Why do we need to create child SA?
  - Create SAs for ESP and AH
    - That is why you see traffic selector negotiations **AGAIN** in this step
  - We can also use them for:
    - Rekey both IKE SAs → create a new SA and delete the old one
  - And
    - Compatible with IKEv1
- **Informational Exchange:**
  - Maintenance Exchange control information related to deleting the tunnel or change in the configuration or a notification as in the case of detecting a past packet outside the anti-replay window.

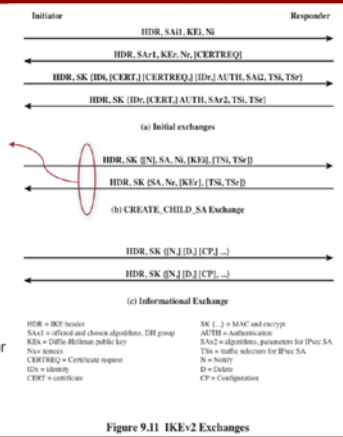


Figure 9.11 IKEv2 Exchanges

IEEE 802.11i Discovery and Association

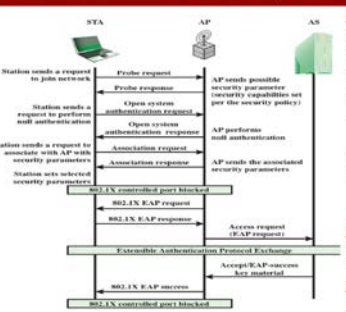


Figure 7.8 IEEE 802.11i Phases of Operation: Capability Discovery, Authentication, and Associatio

Discovery and Association Step 1:
Discovery: Periodic Beacon signals from the AP or responds to a request from the STA
Security Capability (sets of possible suites):

- Confidentiality and integrity protocols
- The specification of a protocol with a key is called a **cipher suite**
- Authentication method
- Key management approach
- Another negotiable suite is Authentication and Key management **AKM**

Discovery and Association Step 2:
Null Authentication: for backward compatibility with 802.11 hardware

Discovery and Association Step 3:

Association: Agree on specific set of security capabilities (one specific cipher suite and one specific AKM suite)

The Handshake Protocol

C→ S: client_hello

- Max TLS **supported** version
- Nonce, Session ID
- List of **supported** cipher suites and compression methods

S→ C: server_hello

- **Chosen** TLS version
- **Chosen** Cipher **suite** and compression method
- Nonce

S→C: Certificate (user authentication)

- Includes a Public Key

S→C: server_key_exchange

- Exchange of cipher **specs**
- This msg is hashed and signed with the sender's private key

S→C: certificate_request

S→C: certificate_hello_done

S→C: Certificate (user authentication)

- Includes a Public Key

S→C: client_key_exchange

- Diffie-Hellman parameters
- This msg is hashed and signed with the sender's private key

S→C: certificate_verify

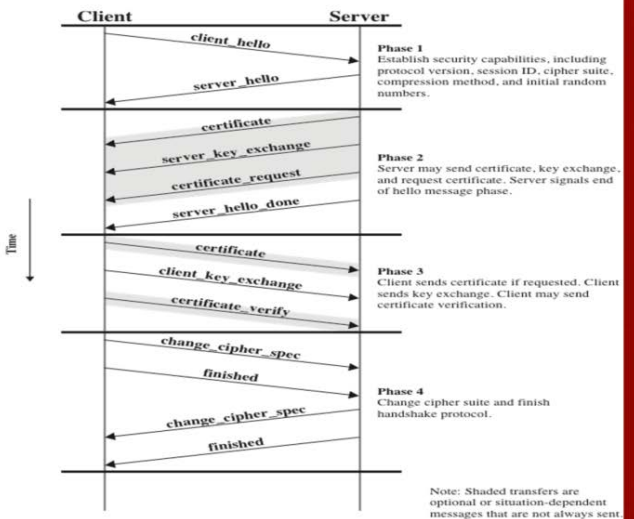


Figure 6.6 Handshake Protocol Action

IEEE 802.11i Authentication Phase

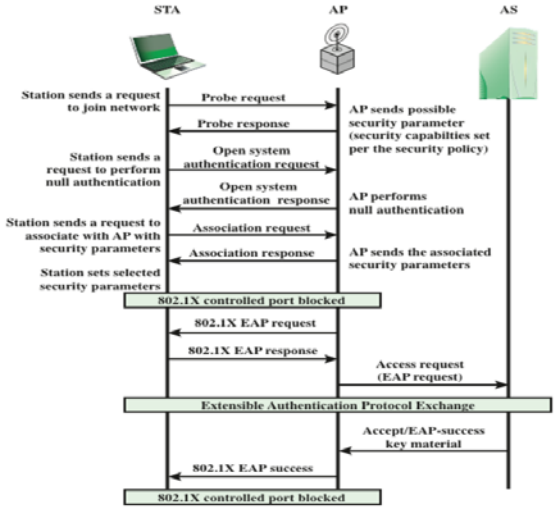


Figure 7.8 IEEE 802.11i Phases of Operation: Capability Discovery, Authentication, and Associatio

Authentication phase:

- Designed to allow only authorized STAs to use the network and to provide STAs with assurance that it is communication with a legitimate network
- Used 802.1x Port based access control and EAPOL

Authentication Step 1: Connect to AS

- STA request from the AP for connection to AS.
- AP acknowledges the request and sends an access request to the AS

Authentication Step 2: EAP Exchange

- Exchange of authentication messages

Authentication Step 3: Secure Key Delivery

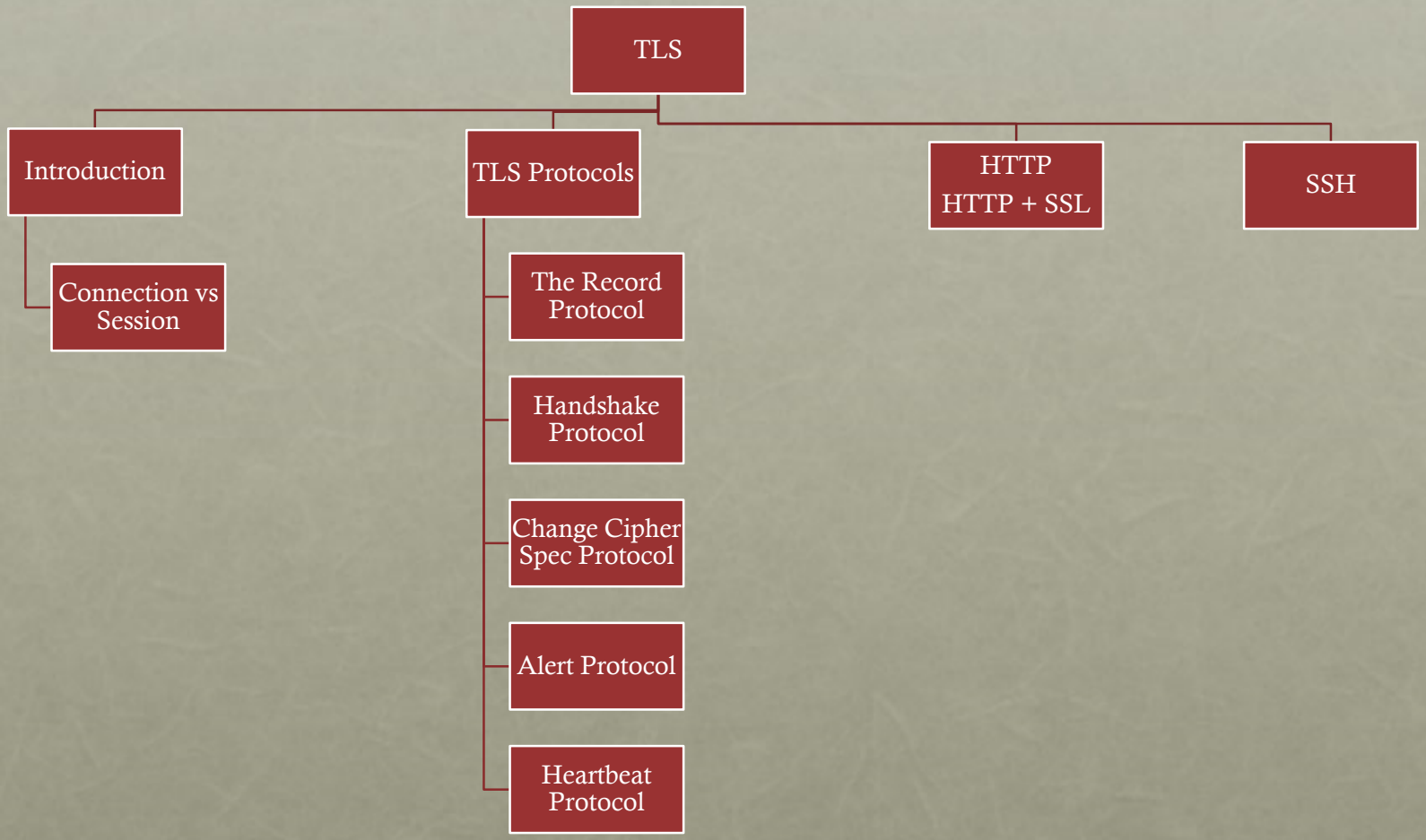
- After authentication is successful, the AS generates a Master Session Key (MSK), also known as AAA Key (Authentication, Authorization and accounting)
- 802.11i relies on EAP for the secure delivery of the MSK

SSL/TLS ATTACKS

Attack categories

- Attacks on the handshake protocol
- Attacks on the record and application data protocols
- Attacks on the PKI
- Other attacks

TLS


TLS
Introduction
Connection vs Session
TLS Protocols
The Record Protocol
Handshake Protocol
Change Cipher Spec Protocol
Alert Protocol
Heartbeat Protocol
HTTP
HTTP + SSL
SSH

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

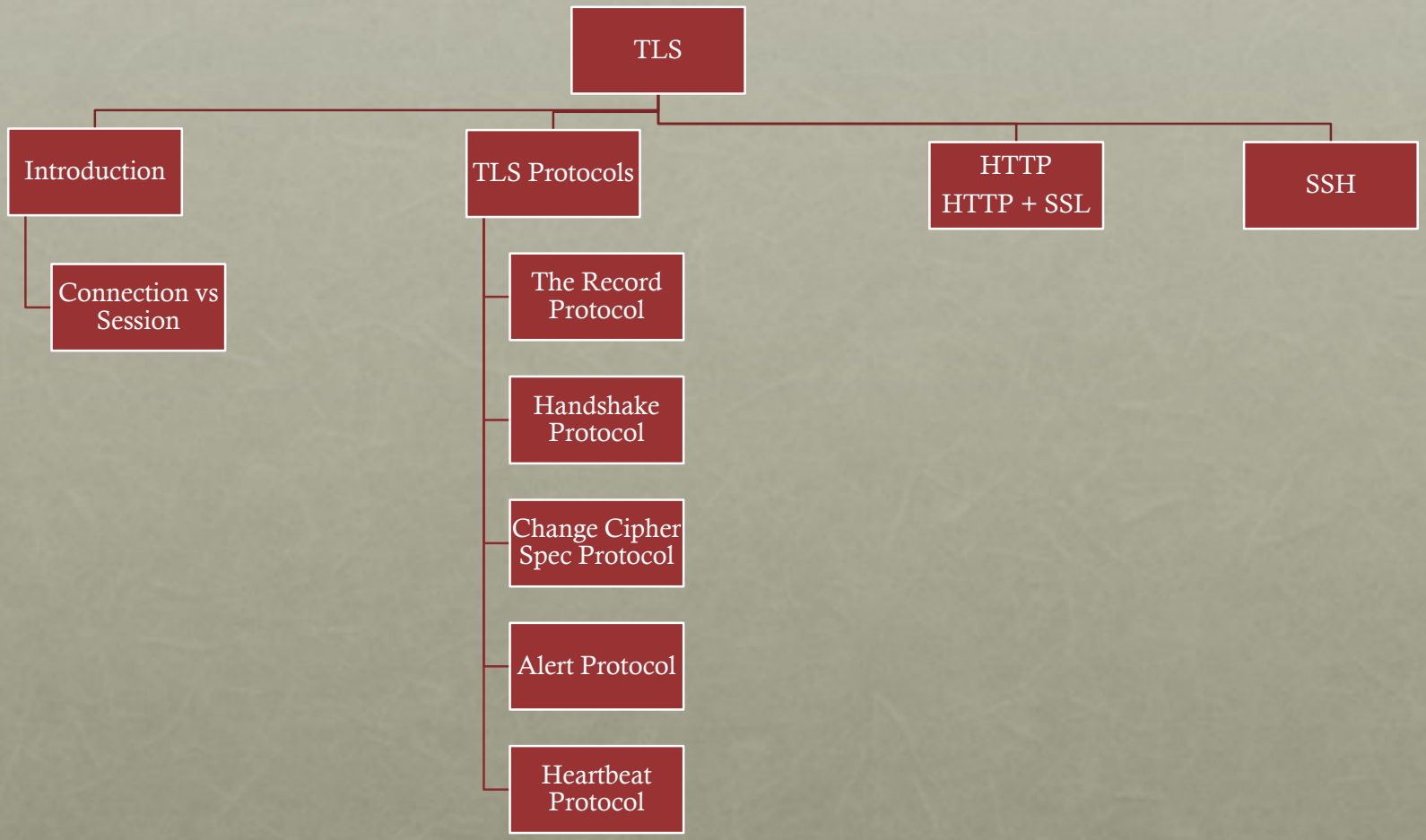
There are three levels of awareness of a connection in HTTPS:

- 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

TLS


TLS
Introduction
Connection vs Session
TLS Protocols
The Record Protocol
Handshake Protocol
Change Cipher Spec Protocol
Alert Protocol
Heartbeat Protocol
HTTP
HTTP + SSL
SSH

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

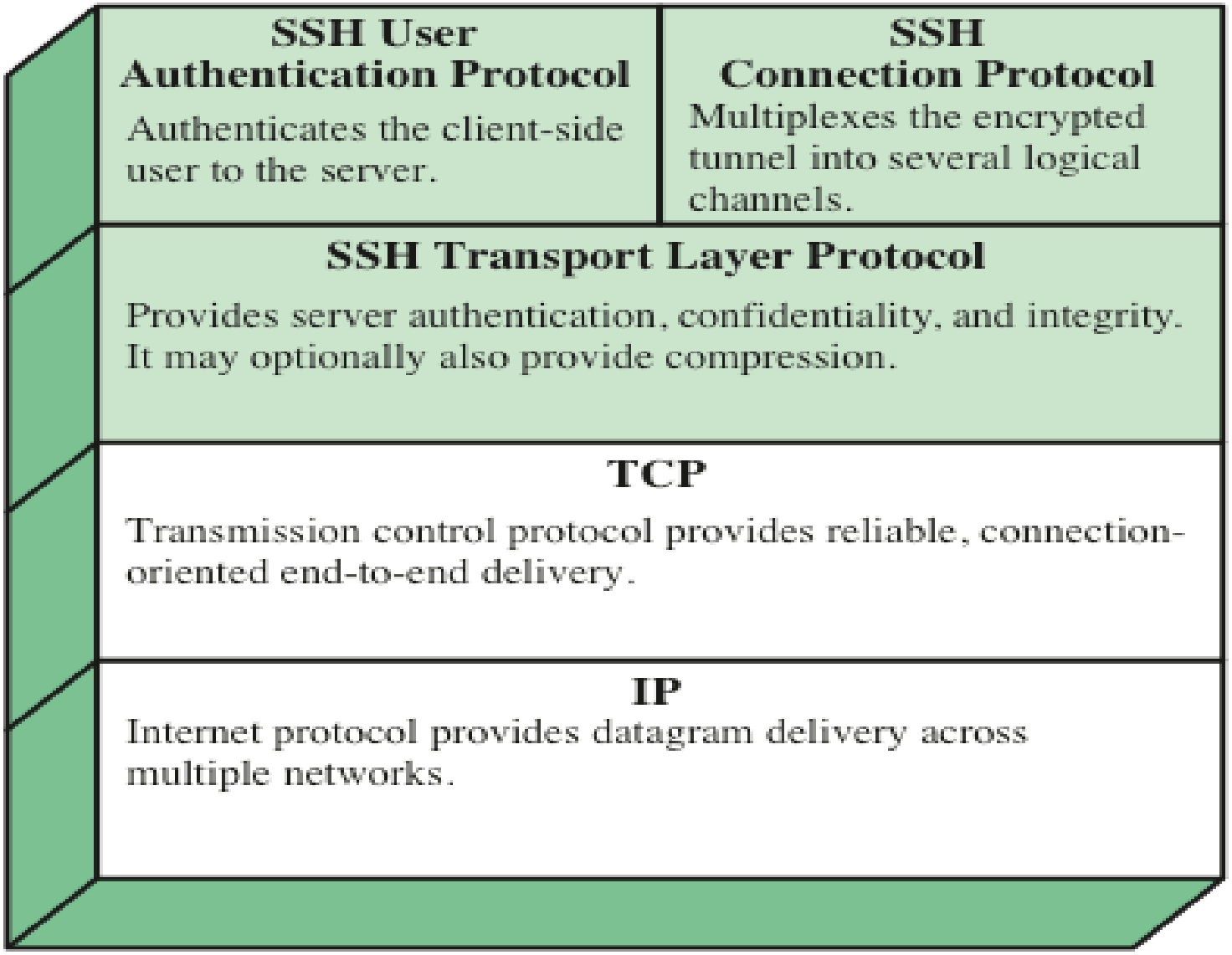


Figure 6.8 SSH Protocol Stack

SSH 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
- 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

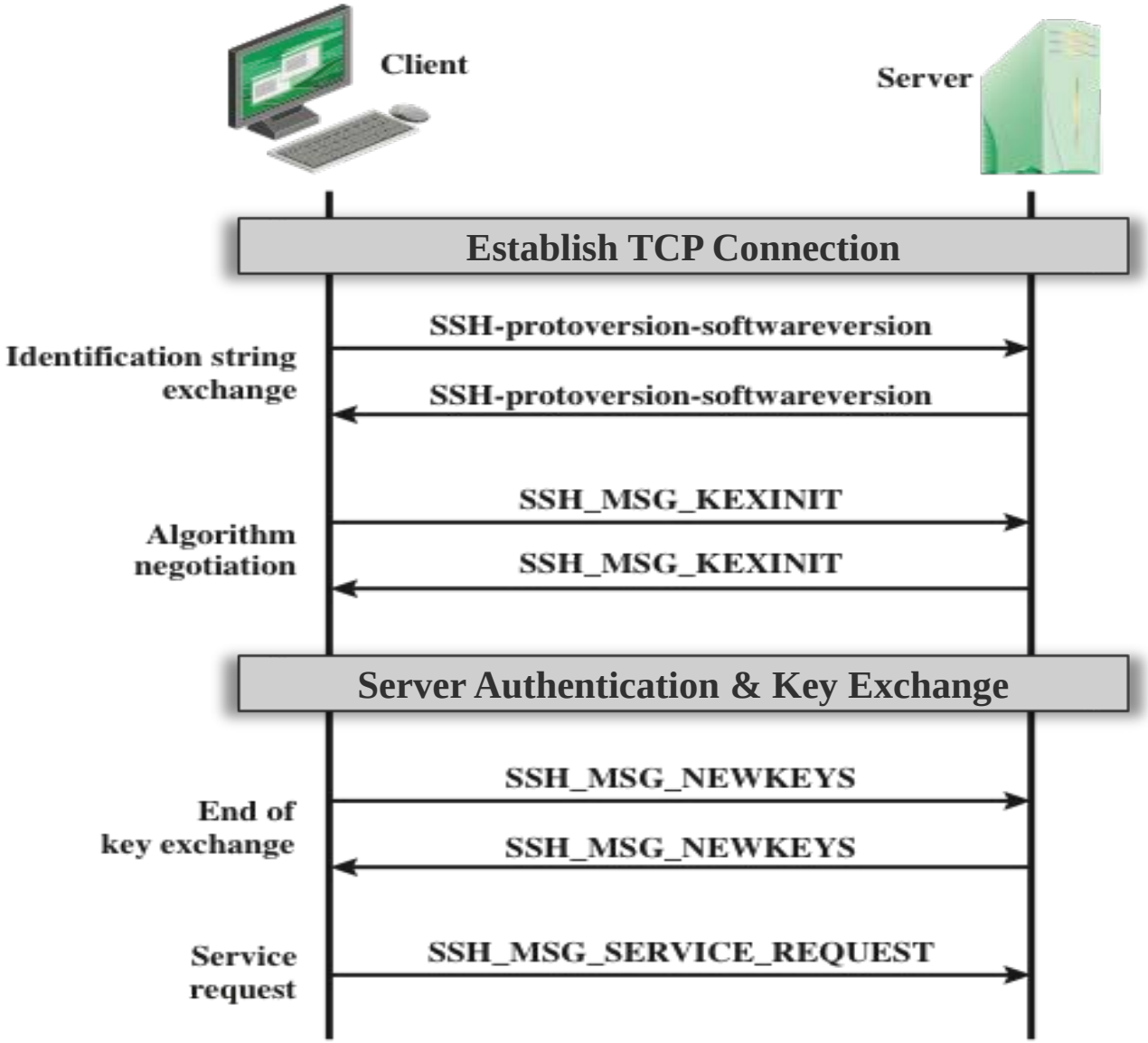


Figure 6.9 SSH Transport Layer Protocol Packet Exchanges

SSH TRANSPORT LAYER PROTOCOL

- The key exchange method allows to establish a shared secret K and the hash value h which are used to derived the SSH session keys:
  - The hash h of the initial key exchange is also taken as the session_id
  - IVClient2Server = Hash(K, h, “A”, session_id) // initialization vector
  - IVServer2Client = Hash(K, h, “B”, session_id) // initialization vector
  - EKClient2Server = Hash(K, h, “C”, session_id) // encryption key
  - EKServer2Client = Hash(K, h, “D”, session_id) // encryption key
  - IKClient2Server = Hash(K, h, “E”, session_id) // integrity key
  - IKServer2Client = Hash(K, h, “F”, session_id) // integrity key

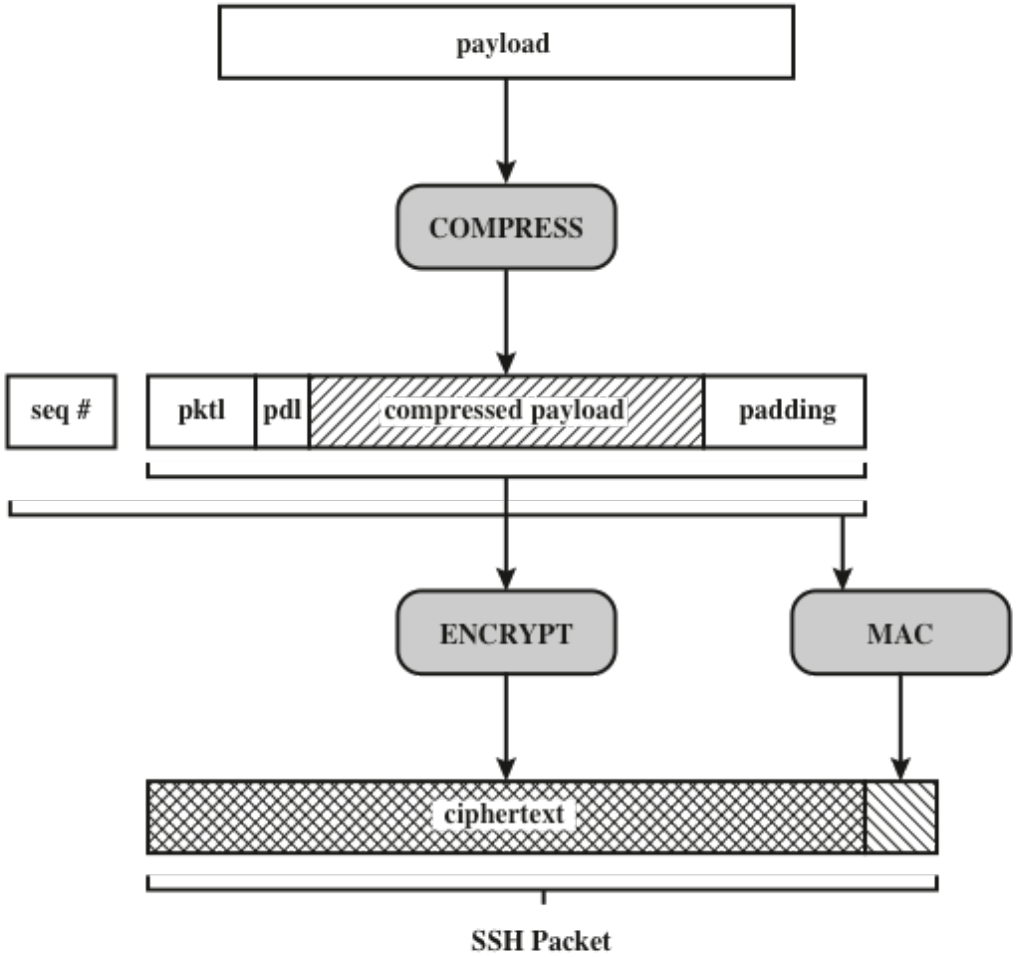


Figure 6.10 SSH Transport Layer Protocol Packet Formation

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

SSH USER 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

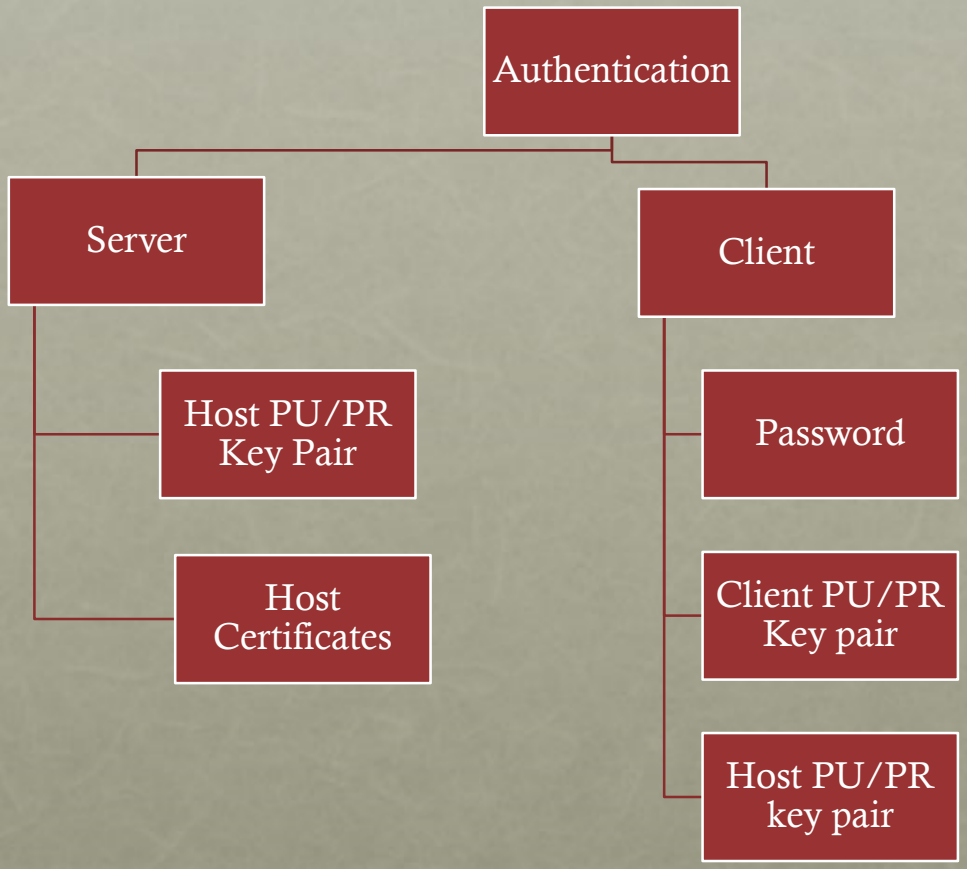
Password

- 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

AUTHENTICATION SUMMARY



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

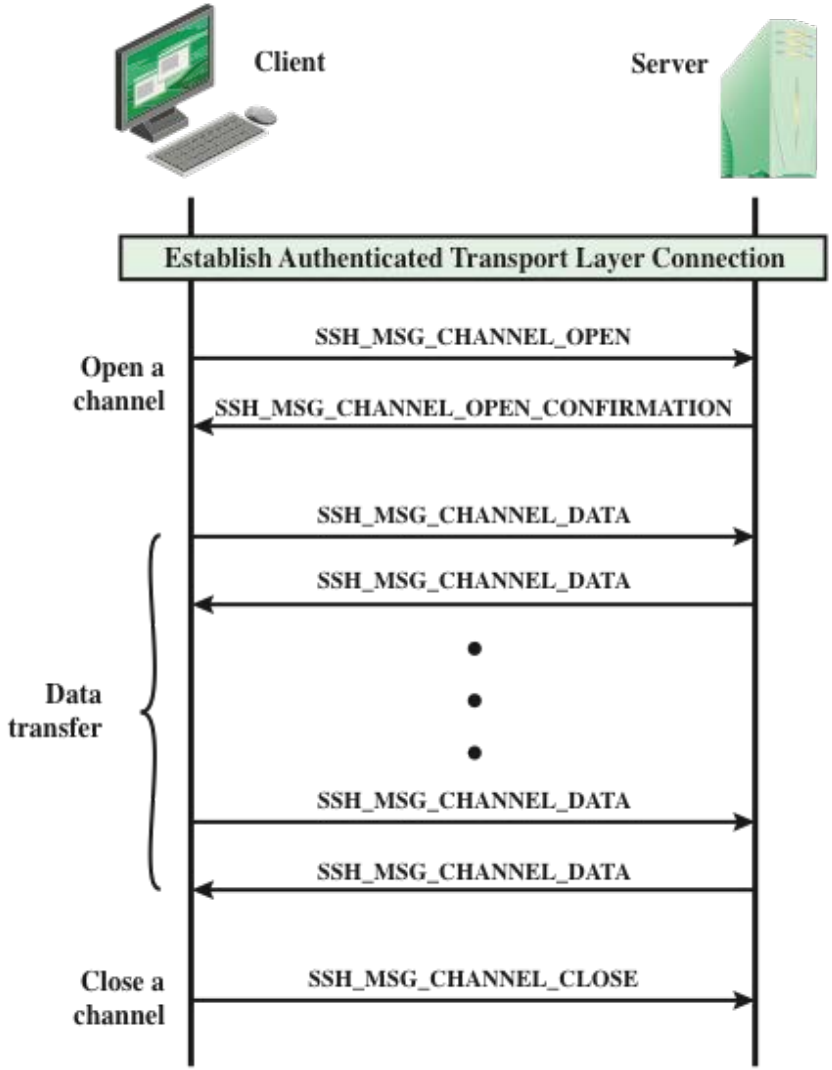


Figure 6.11 Example SSH Connection Protocol Message Exchange

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

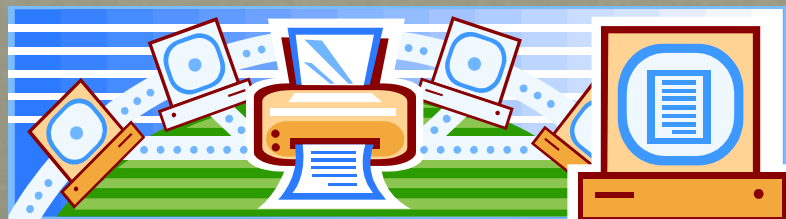
- Remote port forwarding

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



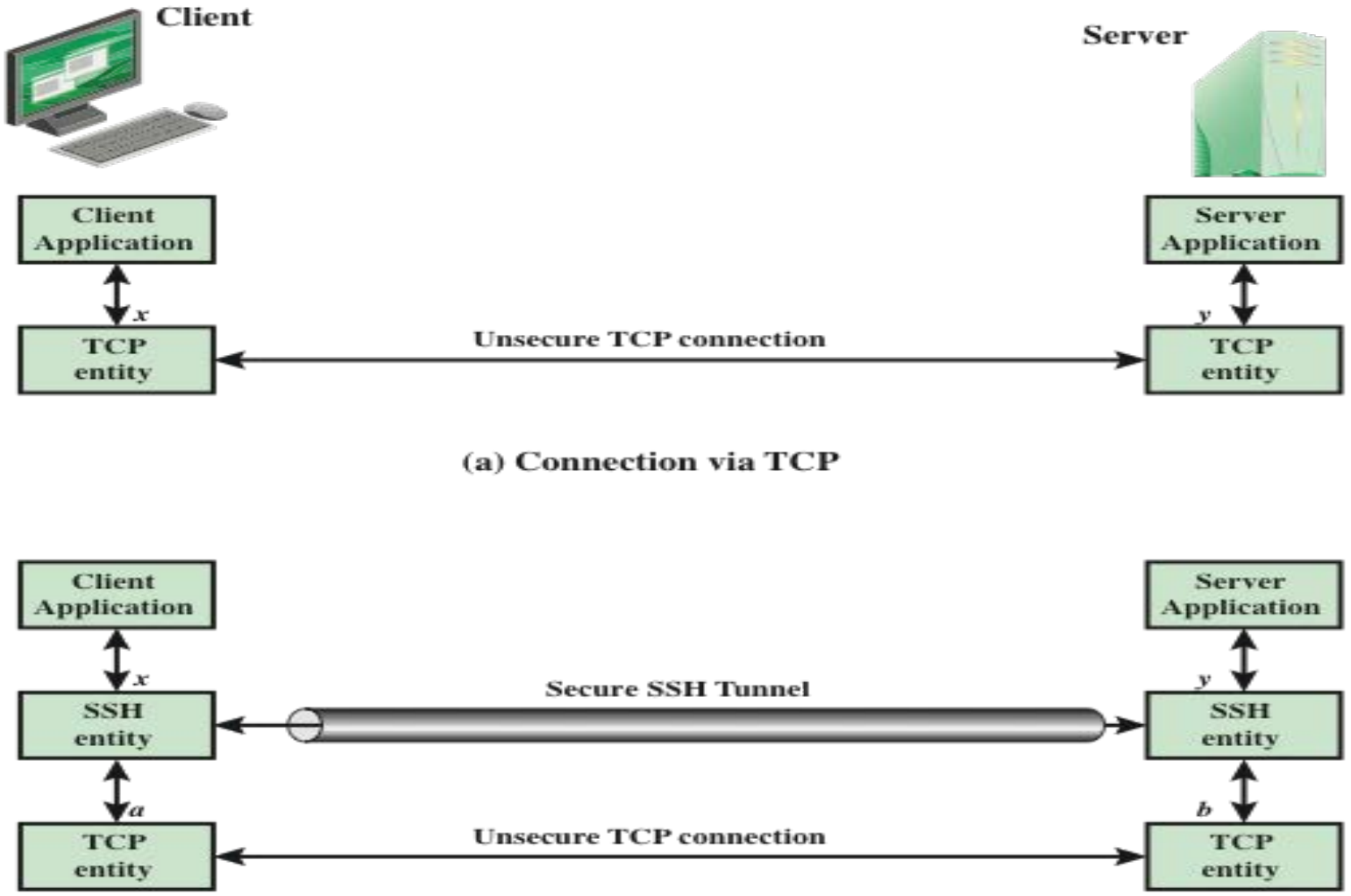


Figure 6.12 SSH Transport Layer Packet Exchanges

PORT FORWARDING

SSH supports two types of port forwarding:

- **Local forwarding:** SSH Client will intercept selected application-level traffic and redirect it from an unsecured TCP connection to a secure SSH tunnel. On the other end, the SSH server sends the incoming traffic to the destination port dictated by the client application.
- **Remote forwarding:** the user's SSH client acts on the server's behalf. The client receives traffic with a given destination port number, places the traffic on the correct port and sends it to the destination the user chooses. **Example:** access your home computer from the internet through firewall or through NAT.

SUMMARY

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