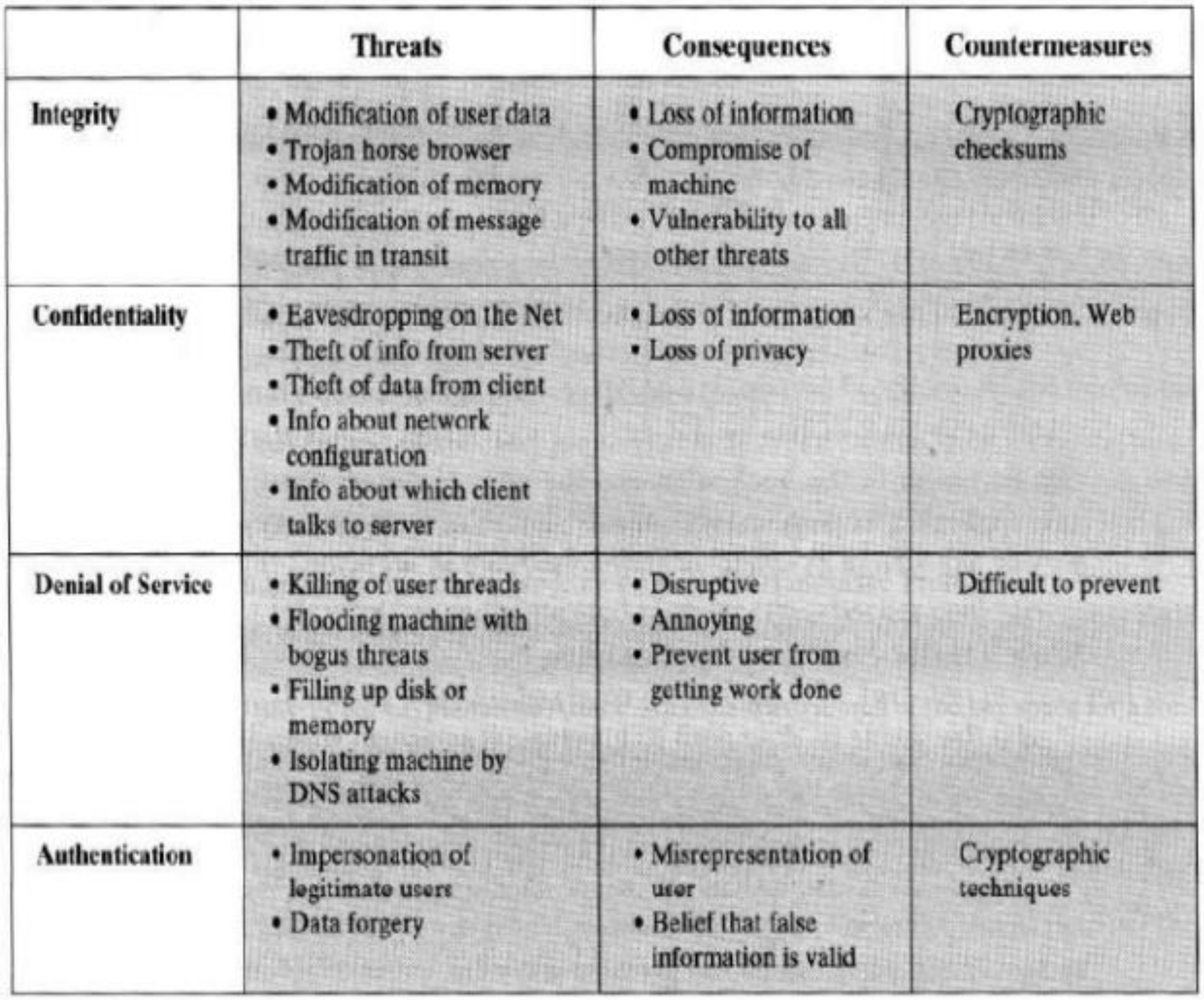
Transport-level Security:Web security considerations, Secure Socket Layer and Transport Layer Security, HTTPS, Secure Shell (SSH)

Wireless Network Security: Wireless Security, Mobile Device Security, IEEE 802.11 Wireless LAN, IEEE 802.11i Wireless LAN Security

Web security considerations:



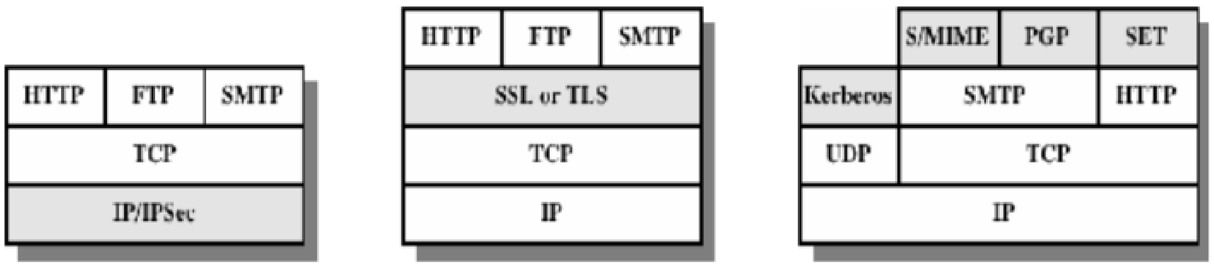
	Threats	Consequences	Countermeasures
Integrity	• Modification of user data • Trojan horse browser • Modification of memory • Modification of message traffic in transit	• Loss of information • Compromise of machine • Vulnerability to all other threats	Cryptographic checksums
Confidentiality	• Eavesdropping 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 threats • 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

- One way of grouping the security threats is in terms of passive and active attacks.
- *Passive attacks* include eavesdropping on network traffic between browser and server and gaining access to information on a website that is supposed to be restricted.
- *Active attacks* include impersonating another user, altering messages in transit between client and server and altering information on a website.
- Another way of classifying these security threats is in terms of location of the threat: Web server, Web browser and network traffic between browser and server.

By providing Security mechanisms to the above attacks we can solve the issue s as well as using SSL/TLs/SSh protocols in the layers we solve the security issues permananetly.

Web Traffic Security Approaches

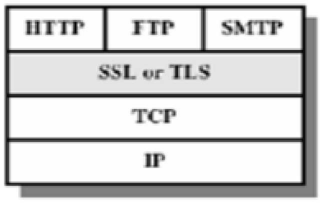
Various approaches for providing Web Security are available, where they are similar in the services they provide and also similar to some extent in the mechanisms they use. They differ with respect to their scope of applicability and their relative location within the TCP/IP protocol stack. The main approaches are IPSec, SSL or TLS and SET.



(a) Network Level (b) Transport Level (c) Application Level

- **IPSec** provides security at the network level and the main advantage is that it is transparent to end users and applications. In addition, IPSec includes a filtering capability so that only selected traffic can be processed.
- **Secure Socket Layer or Transport Layer Security (SSL/TLS)** provides security just above the TCP at transport layer. Two implementation choices are present here. Firstly, the SSL/TLS can be implemented as a part of TCP/IP protocol suite, thereby being transparent to applications. Alternatively, SSL can be embedded in specific packages like SSL being implemented by Netscape and Microsoft Explorer browsers.
- **Secure Electronic Transaction (SET)** approach provides application-specific services i.e., according to the security requirements of a particular application. The main advantage of this approach is that service can be tailored to the specific needs of a given application.

Secure Socket Layer/Transport Layer Security



(b) Transport Level

Before going to discuss about SSL Protocol we have to know about Two important SSL concepts are the SSL session and the SSL connection, which are defined in the specification as follows:

• **Connection**: A connection is a transport (in the OSI layering model definition) that provides a suitable type of service. For SSL, such connections are peer-to-peer relationships. The connections are transient. Every connection is associated with one session.

• **Session**: An SSL session is an association between a client and a server. Sessions are created by the Handshake Protocol. Sessions define a set of cryptographic security parameters, which can be shared among multiple connections. Sessions 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 (such as null, AES, etc.) and a hash algorithm (such as MD5 or SHA-1) used for MAC calculation. It also defines cryptographic attributes such as the hash_size.
- Master secret*:*48-byte secret shared between the client and 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 conventional encryption key for data encrypted by the server and decrypted by the client.
- Client write key: The conventional 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. Thereafter 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 264-1.

SSL Protocol

- SSL was developed by Netscape to provide security when transmitting information on the Internet.
- The Secure Sockets Layer protocol is a protocol layer which may be placed between a TCP and the application protocol layer HTTP.
- SSL provides for secure communication between client and server by allowing mutual authentication, the use of digital signatures for integrity and encryption for privacy.
- SSL protocol has different versions such as SSLv2.0, SSLv3.0,
- 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 shown below:

- Four higher-layer protocols are defined as part of SSL: the

  (a) The SSL Record Protocol provides basic security services to various higher-layer protocols.

  (b)Handshake Protocol

  (c) The Change Cipher Spec Protocol and

  (d) The Alert Protocol.

SSL Record Protocol

The SSL Record Protocol provides two services for SSL connections:

• Confidentiality: The Handshake Protocol defines a shared secret key that is used for conventional encryption of SSL payloads.
• Message Integrity: The Handshake Protocol also defines a shared secret key that is used to form a message authentication code (MAC).

The Record Protocol takes an application message to be transmitted, fragments the data into manageable blocks, optionally compresses the data, applies a MAC, encrypts, adds a header, and transmits the resulting unit in a TCP segment. Received data are decrypted, verified, decompressed, and reassembled and then delivered to higher-level users. The overall operation of the SSL Record Protocol is shown below:

- The first step is fragmentation.
- Each upper-layer message is fragmented into blocks of 214 bytes (16384 bytes) or less. Next, compression is optionally applied.
- Compression must be lossless and may not increase the content length by more than 1024 bytes.
- The next step in processing is to compute a message authentication code over the compressed data.
- For this purpose, a shared secret key is used. The calculation is defined as:

hash(MAC_write_secret || pad_2 || hash(MAC_write_secret || pad_1 || seq_num || SSLCompressed.type || SSLCompressed.length || SSLCompressed.fragment))

Where,

MAC_write_secret = Secret shared key pad_1	=	the byte 0x36 (0011 0110) repeated 48 times (384 bits) for MD5 and 40 times for
pad_2	=	the byte 0x5C (0101 1100) repeated 48 times for MD5 and 40 times for SHA-1

The main difference between HMAC and SSL protocol is that the two pads are concatenated in SSLv3 and are XORed in HMAC. The encryption algorithms allowed are AES-128/256, IDEA-128, DES-40, 3DES-168, RC2-40, Fortezza, RC4-40 and RC4-128. For stream encryption, the compressed message plus the MAC are encrypted whereas, for block encryption, padding may be added after the MAC prior to encryption.

The final step of SSL Record Protocol processing is to prepend a header, consisting of the following fields:

• Content Type (8 bits): The higher layer protocol used to process the enclosed fragment.

• Major Version (8 bits): Indicates major version of SSL in use. For SSLv3, the value is 3.

• Minor Version (8 bits): Indicates minor version in use. For SSLv3, the value is 0.

• Compressed Length (16 bits): The length in bytes of the plaintext fragment (or compressed fragment if compression is used). The maximum value is $2^{14} + 2048$.

The content types that have been defined are change_cipher_spec, alert, handshake, and application_data.

SSL Change Cipher Spec Protocol

The Change Cipher Spec Protocol is 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, 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.

SSL Alert Protocol

The Alert Protocol is 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.

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.

The fatal alerts are listed below

• unexpected_message: An inappropriate message was received.

• bad_record_mac: An incorrect MAC was received.

• decompression_failure: The decompression function received improper input (e.g.,

nable to decompress or decompress to greater than maximum allowable length).

• handshake_failure: Sender was unable to negotiate an acceptable set of security parameters given the options available. illegal_parameter: A field in a handshake message was out of range or inconsistent with other fields.

The remainder of the alerts are given below:

- close_notify: Notifies the recipient that the sender will not send any more messages on this connection. Each party is required to send a close_notify alert before closing the write side of a connection.
- no_certificate: May be sent in response to a certificate request if no appropriate certificate is available.
- bad_certificate: A received certificate was corrupt (e.g., contained a signature that did not verify).
- unsupported_certificate: The type of the received certificate is not supported.
- certificate_revoked: A certificate has been revoked by its signer.
- certificate_expired: A certificate has expired.
- certificate_unknown: Some other unspecified issue arose in processing the certificate, rendering it unacceptable.

SSL Handshake Protocol

SSL Handshake protocol ensures establishment of reliable and secure session between client and server and also allows server & client to:

- authenticate each other
- to negotiate encryption & MAC algorithms
- to negotiate cryptographic keys to be used

The Handshake Protocol consists of a series of messages exchanged by client and server. All of these have the format shown below and each message has three fields:

- **Type (1 byte**): Indicates one of 10 messages.
- **Length (3 bytes):** The length of the message in bytes.
- **Content (>=0 bytes):** The parameters associated with this message

The following figure shows the initial exchange needed to establish a logical connection between client and server. The exchange can be viewed as having four phases.in phases

o Establish Security Capabilities

o Server Authentication and Key Exchange

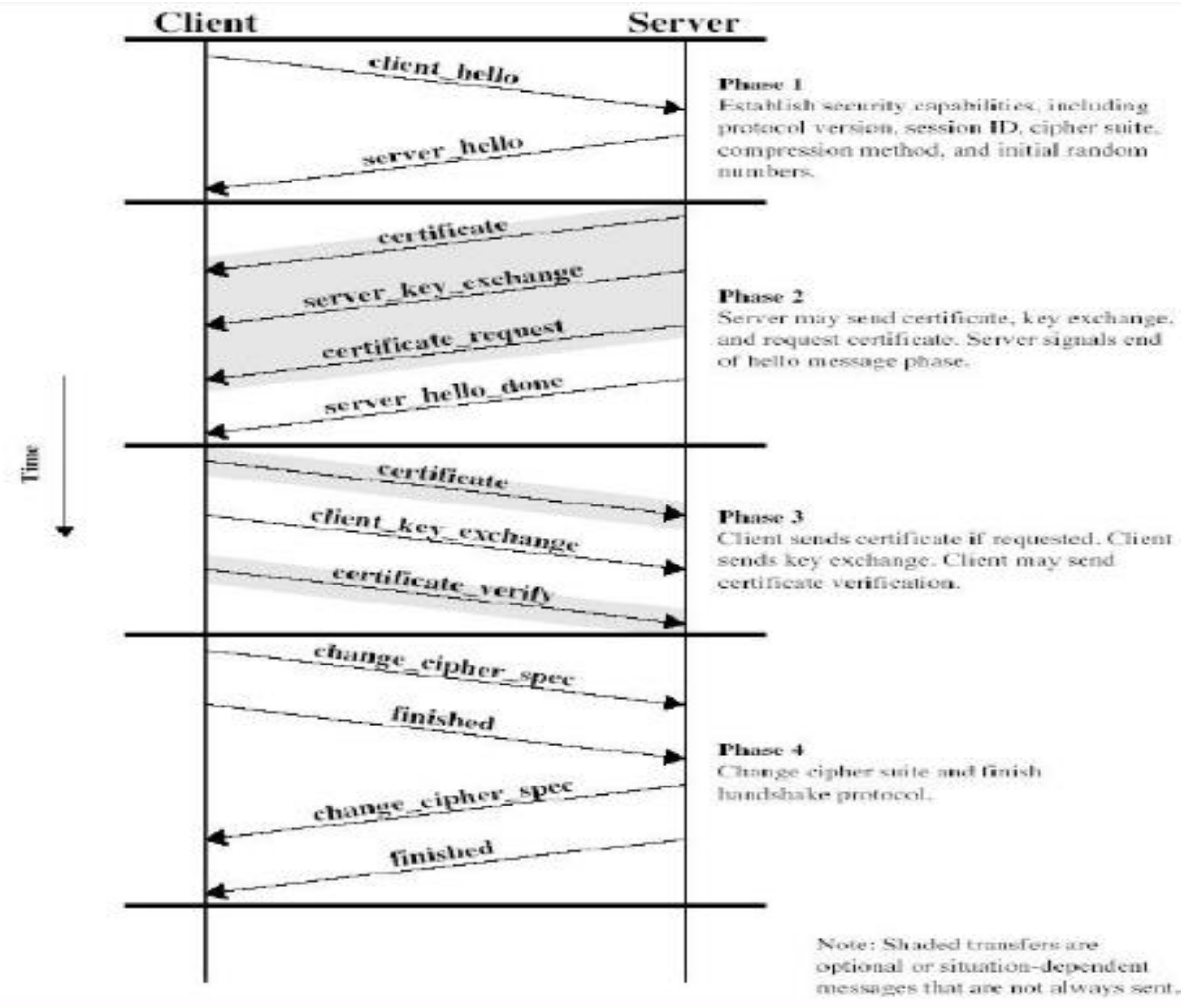
o Client Authentication and Key Exchange

o Finish

Phase 1. Establish Security Capabilities

This phase is used to initiate a logical connection and to establish the security capabilities that will be associated with it. The exchange is initiated by the client, which sends a client_hello message with the following 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 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.
- Compression Method: This is a list of the compression methods the client supports.


Client
Server
client_hello
server_hello
Phase 1
Establish security capabilities, including protocol version, session ID, cipher suite, compression method, and initial random numbers.
certificate
server_key_exchange
certificate_request
server_hello_done
Phase 2
Server may send certificate, key exchange, and request certificate. Server signals end of hello message phase.
Time
certificate
client_key_exchange
certificate_verify
Phase 3
Client sends certificate if requested. Client sends key exchange. Client may send certificate verification.
change_cipher_spec
finished
change_cipher_spec
finished
Phase 4
Change cipher suite and finish handshake protocol.
Note: Shaded transfers are optional or situation-dependent messages that are not always sent.

Phase 2. Server Authentication and Key Exchange

The server begins this phase by sending its certificate via a certificate message, which contains one or a chain of X.509 certificates. The **certificate message** is required for any agreed-on key exchange method except anonymous Diffie-Hellman. Next, a **server_key_exchange** message may be sent if it is required. It is not required in two instances: (1) The server has sent a certificate with fixed Diffie-Hellman parameters, or (2) RSA key exchange is to be used.

Phase 3. Client Authentication and Key Exchange

Once the server_done message is received by client, it should verify whether a valid certificate is provided and check that the server_hello parameters are acceptable. If all is satisfactory, the client sends one or more messages back to the server. If the server has requested a certificate, the client begins this phase by sending a **certificate message**.

If no suitable certificate is available, the client sends a no_certificate alert instead. Next is the **client_key_exchange** message, for which the content of the message depends on the type of key exchange.

Phase 4. Finish

This phase completes the setting up of a secure connection. The client sends a **change_cipher_spec** message and copies the pending CipherSpec into the current CipherSpec. The client then immediately sends the finished message under the new algorithms, keys, and secrets. The finished message verifies that the key exchange and authentication processes were successful.

TRANSPORT LAYER SECURITY

Version Number

The TLS Record Format is the same as that of the SSL Record Format (Figure 17.4), 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. HMAC is defined as

HMAC*K*(*M*) = H[(*K*+ _ opad) } H[(*K*+ _ ipad) } *M*]]

where

H = embedded hash function (for TLS, either MD5 or SHA-1)

M = message input to HMAC

K+ = secret key padded with zeros on the left so that the result is equal to the block length of the hash code (for MD5 and SHA-1, block length = 512 bits)

ipad = 00110110 (36 in hexadecimal) repeated 64 times (512 bits)

opad = 01011100 (5C in hexadecimal) repeated 64 times (512 bits)

SSLv3 uses the same algorithm, except that the padding bytes are concatenated with the secret key rather than being XORed with the secret key padded to the block length. The level of security should be about the same in both cases. For TLS, the MAC calculation encompasses the fields indicated in the following expression:

MAC(MAC_write_secret,seq_num } TLSCompressed.type }
TLSCompressed.version } TLSCompressed.length }
TLSCompressed.fragment)

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.

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; of these, the following are always fatal.

• **record_overflow:** A TLS record was received with a payload (ciphertext) whose length exceeds 214 + 2048 bytes, or the ciphertext decrypted to a length of greater than 214 + 1024 bytes.
• **unknown_ca:** A valid certificate chain or partial chain was received, but the certificate was not accepted because the CA certificate could not be located or could not be matched with a known, trusted CA.

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; of these, the following are always fatal.

• **record_overflow:** A TLS record was received with a payload (ciphertext) whose length exceeds 214 + 2048 bytes, or the ciphertext decrypted to a length of greater than 214 + 1024 bytes.
• **unknown_ca:** A valid certificate chain or partial chain was received, but the certificate was not accepted because the CA certificate could not be located or could not be matched with a known, trusted CA.

• **access_denied:** A valid certificate was received, but when access control was applied, the sender decided not to proceed with the negotiation.

• **decode_error:** A message could not be decoded, because either a field was out of its specified range or the length of the message was incorrect.

• **protocol_version:** The protocol version the client attempted to negotiate is recognized but not supported.

• **insufficient_security:** Returned instead of handshake_failure when a negotiation has failed specifically because the server requires ciphers more secure than those supported by the client.

• **unsupported_extension:** Sent by clients that receive an extended server hello containing an extension not in the corresponding client hello.

• **internal_error:** An internal error unrelated to the peer or the correctness of the protocol makes it impossible to continue.

• **decrypt_error:** A handshake cryptographic operation failed, including being unable to verify a signature, decrypt a key exchange, or validate a finished message.

Pseudorandom Function

TLS makes use of a pseudorandom function referred to as PRF to expand secrets into blocks data for purposes of key generation or validation. The objective is to make use of a relatively small shar secret value but to generate longer blocks of data in a way that is secure from the kinds of attacks mad hash functions and MACs. The PRF is based on the data expansion function (Figure 17.7) given as

P_hash(secret, seed)= HMAC_hash(secret,A(1) } seed) }
HMAC_hash(secret,A(2) } seed) }
HMAC_hash(secret,A(3) } seed) } . . .

where A() is defined as A(0) = seed
A(i) = HMAC_hash(secret, A(i - 1))

The data expansion function makes use of the HMAC algorithm with either MD5 or SHA-1 as the underlying hash function. As can be seen, P_hash can be iterated as many times as necessary to produce the required quantity of data. For example, if P_SHA-1 was used to generate 64 b of data, it would have to be iterated four times, producing 80 bytes of data of which the last 16 would be discarded. In this case, P_MD5 would also have to be iterated four times, producing exactly 64 bytes of data. Note that each iteration involves two executions of HMAC—each of which in turn involves two executions of the underlying hash algorithm.

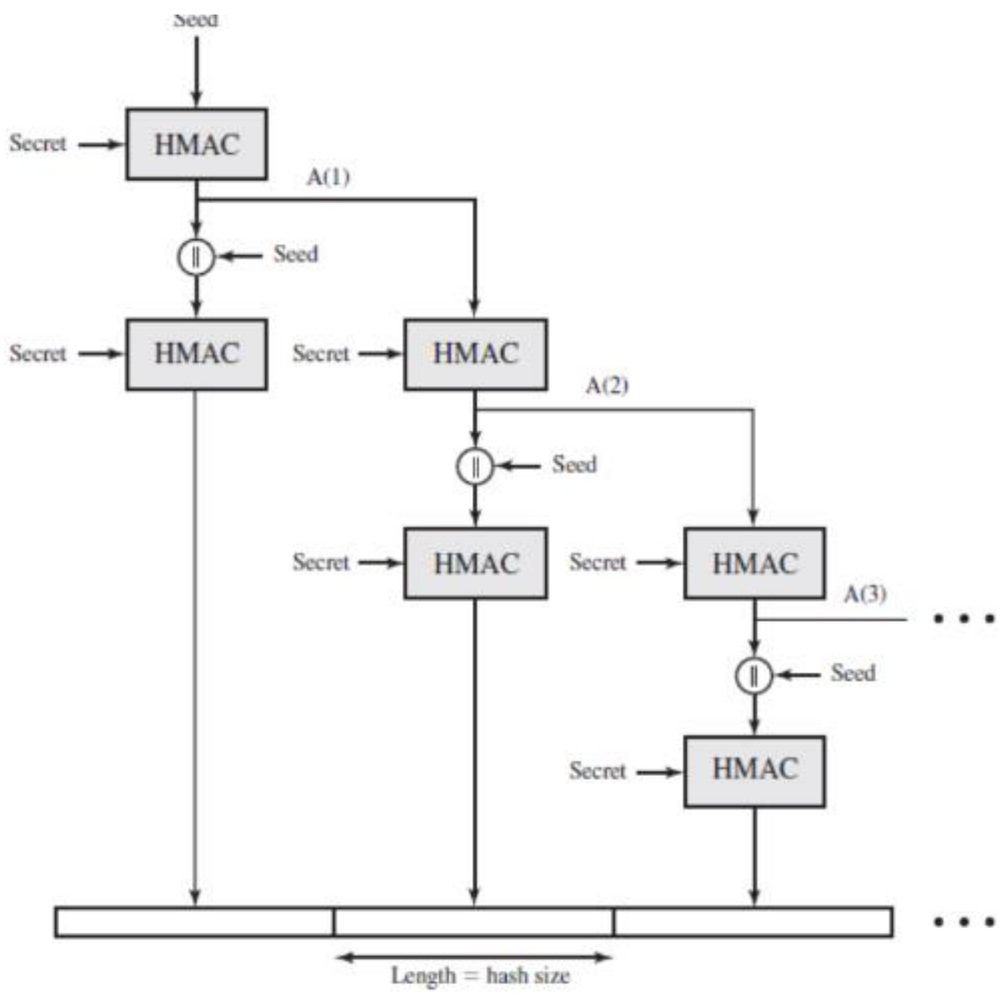


Figure 17.7 TLS Function `P_hash(secret, seed)`

HTTPS

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. Its use depends on the Web server supporting HTTPS communication. For example, some search engines do not support HTTPS. Google provides HTTPS as an option: https://google.com. The principal difference seen by a user of a Web browser is that URL (uniform resource locator) addresses begin with https:// rather than http://. A normal HTTP connection uses port 80. If HTTPS is specified, port 443 is used, which invokes SSL. 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 (filled in by browser user)
- Cookies sent from browser to server and from server to browser
- Contents of HTTP header

HTTPS is 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.

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. Normal HTTP behavior, including retained connections, should be followed.

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 also may 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. As we have seen, 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 following line in an HTTP record: Connection: close. This indicates that the connection will be closed after this record is delivered. 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. At the TLS level, the proper way to close a connection is for each side to use the TLS alert protocol to send a close_notify alert. TLS implementations must initiate an exchange of closure alerts before closing a connection.

Secure Shell (SSH)

Secure Shell (SSH) is 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. SSH client and server applications are widely available for most operating systems. SSH is organized as three protocols that typically run on top of TCP
(Figure 17.8):

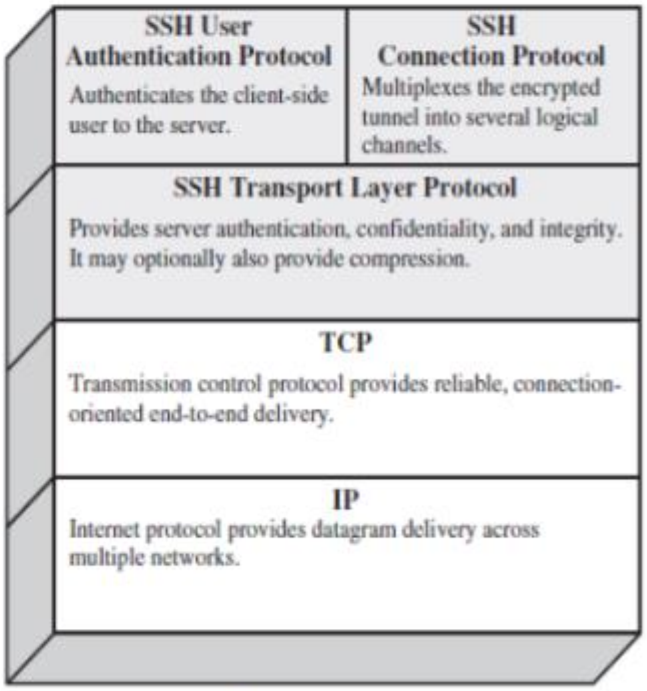

SSH User Authentication Protocol
Authenticates the client-side user to the server.
SSH Connection Protocol
Multiplexes the encrypted tunnel into several logical channels.
SSH Transport Layer Protocol
Provides server authentication, confidentiality, and integrity. It may optionally also provide compression.
TCP
Transmission control protocol provides reliable, connection-oriented end-to-end delivery.
IP
Internet protocol provides datagram delivery across multiple networks.

Figure 17.8 SSH Protocol Stack

- **Transport Layer Protocol:** Provides server authentication, data confidentiality, and data integrity with forward secrecy (i.e., if a key is compromised during one session, the knowledge does not affect the security of earlier sessions). The transport layer may optionally provide compression.
- **User Authentication Protocol:** Authenticates the user to the server.
- **Connection Protocol:** Multiplexes multiple logical communications channels over a single, underlying SSH connection.

Transport Layer Protocol

Host Keys : Server authentication occurs at the transport layer, based on the server possessing a public/private key pair. The server host key is used during key exchange to authenticate the identity of the host. For this to be possible, the client must have a priori knowledge of the server's public host key. Next, consider events in the SSH Transport Layer Protocol. First, a client establishes a TCP connection to the server. Once the connection is established, the client and server exchange data, referred to as packets, in the data field of a TCP segment.

Packet Exchange :Figure 17.9 illustrates the sequence of events in the SSH Transport Layer Protocol. First, the client establishes a TCP connection to the server. This is done via the TCP protocol and is not part of the Transport Layer Protocol. Once the connection is established, the client and server exchange data, referred to as packets, in the data field of a TCP segment

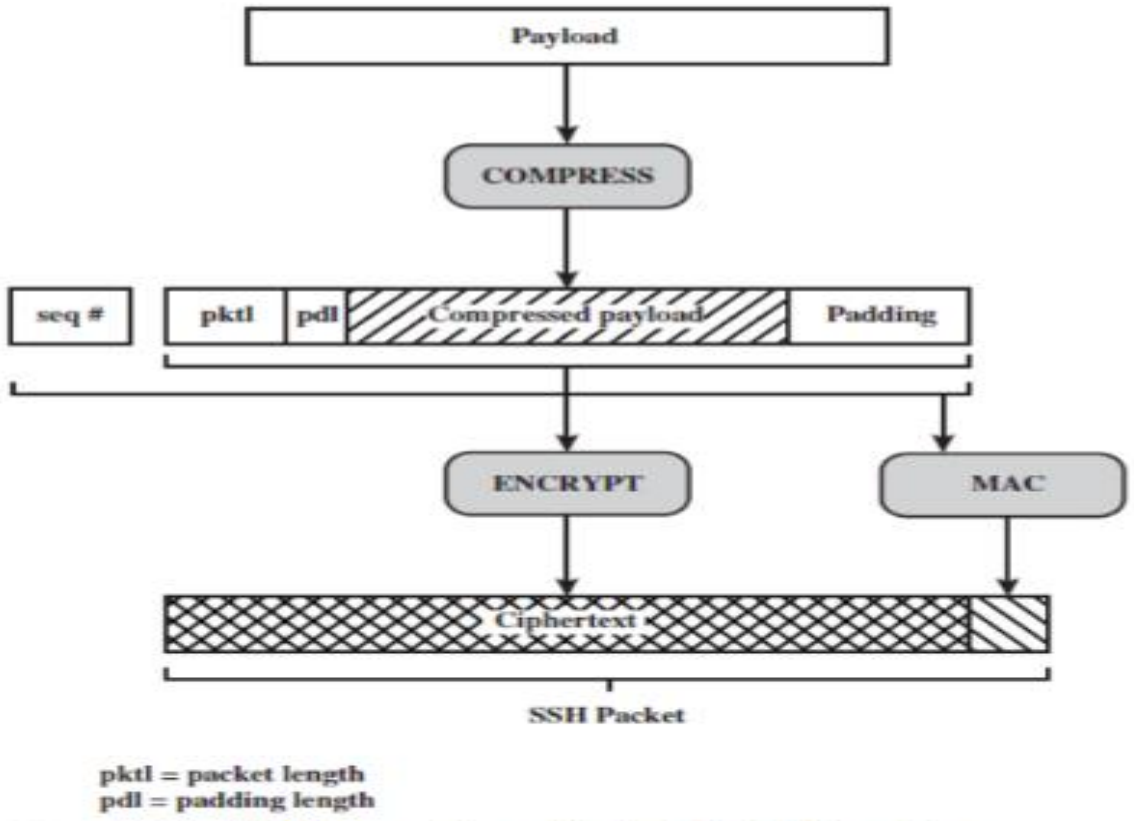


Figure 17.10 SSH Transport Layer Protocol Packet Formation

Each packet is in the following format (Figure 17.10).

• **Packet length:** Length of the packet in bytes, not including the packet length and MAC fields.
• **Padding length:** Length of the random padding field.
• **Payload:** Useful contents of the packet. Prior to algorithm negotiation, this field is uncompressed. If compression is negotiated, then in subsequent packets, this field is compressed.
• **Random padding:** Once an encryption algorithm has been negotiated, this field is added. It contains random bytes of padding so that that total length of the packet (excluding the MAC field) is a multiple of the cipher block size, or 8 bytes for a stream cipher.
• **Message authentication code (MAC):** If message authentication has been negotiated, this field contains the MAC value. The MAC value is computed over the entire packet plus a sequence number, excluding the MAC field. The sequence number is an implicit 32-bit packet sequence that is initialized to zero for the first packet and incremented for every packet. The sequence number is not included in the packet sent over the TCP connection.

Once an encryption algorithm has been negotiated, the entire packet (excluding the MAC field) is encrypted after the MAC value is calculated.

The SSH Transport Layer packet exchange consists of a sequence of steps (Figure 17.9).

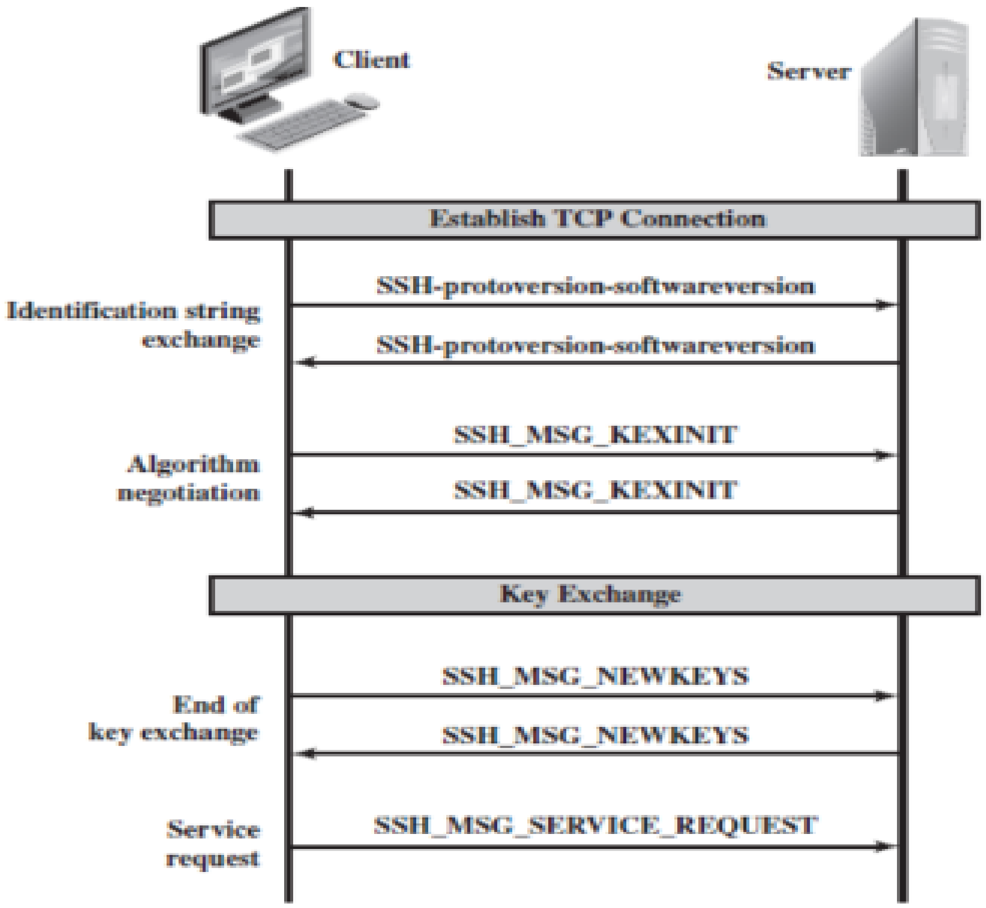


Figure 17.9 SSH Transport Layer Protocol Packet Exchanges

- The first step, the **identification string exchange**, begins with the client sending a packet with an identification string of the form:

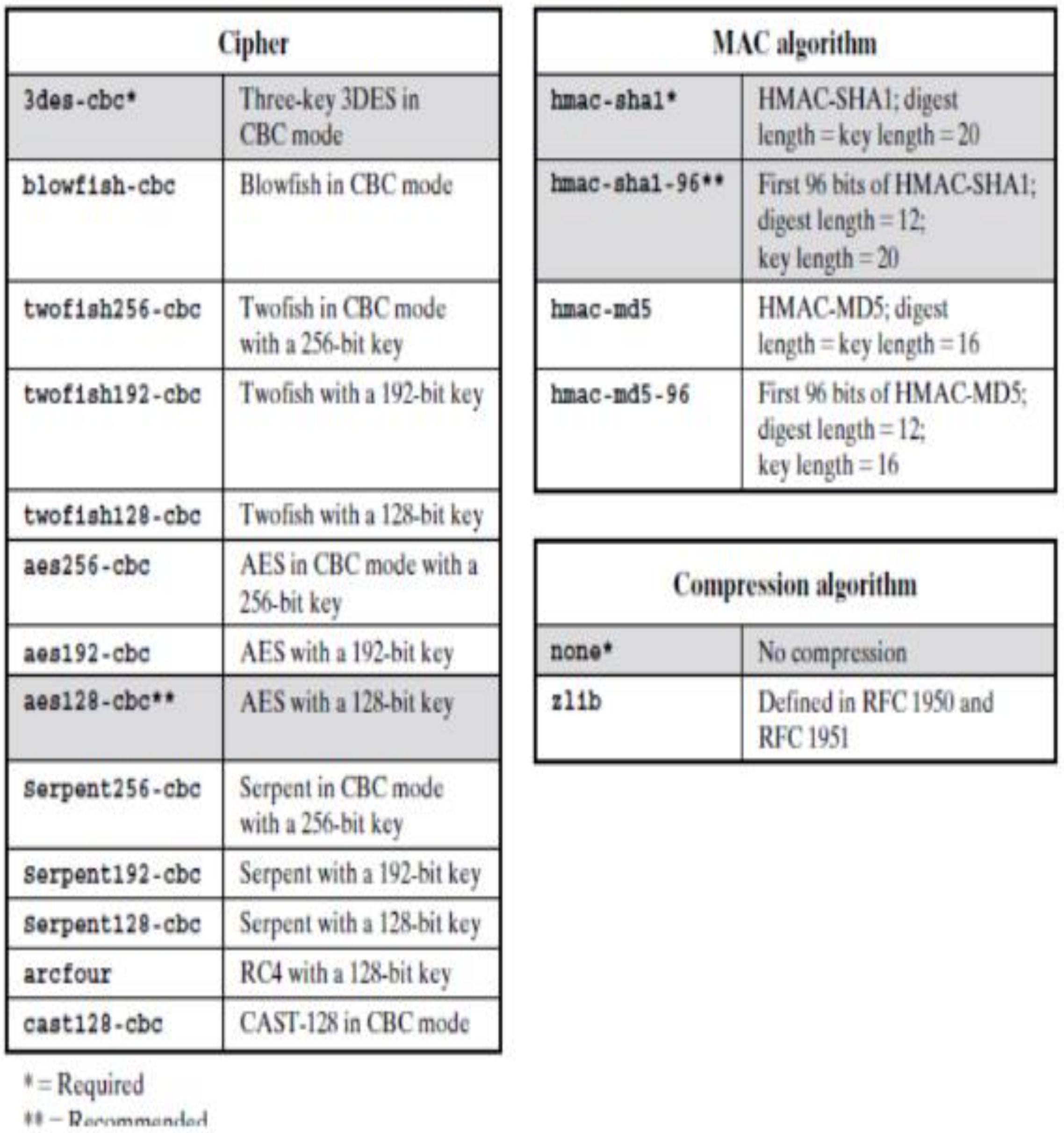
SSH-protoversion-softwareversion SP comments CR LF

where SP, CR, and LF are space character, carriage return, and line feed, respectively.

An example of a valid string is SSH-2.0-billsSSH_3.6.3q3<CR><LF>. The server responds with its own identification string. These strings are used in the Diffie-Hellman key exchange.

- Next comes **algorithm negotiation**. Each side sends an SSH_MSG_KEXINIT containing lists of supported algorithms in the order of preference to the sender. There is one list for each type of cryptographic algorithm. The algorithms include key exchange, encryption, MAC algorithm, and compression algorithm. Table 17.3 shows the allowable options for encryption, MAC, and compression. For each category, the algorithm chosen is the first algorithm on the client's list that is also supported by the server.

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

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

- The next step is **key exchange**. The specification allows for alternative methods of key exchange, but at present, only two versions of Diffie-Hellman key exchange are specified. Both versions are defined in RFC 2409 and require only one packet in each direction.

User Authentication Protocol

In this Protocol the client is authenticated to the server by using the following 3 methods .

(1) Message Types and Formats

(2) Message Exchange

(3) Authentication Methods

(1) Message Types and Formats : The client will send the msg in the following format

SSH_MSG_USERAUTH_REQUEST (50)

string user name	- (authorization identity)
string service name	- (service type)
string method name	- (authentication method)

… method specific fields

. **(2) Message Exchange :**

1. The client sends a SSH_MSG_USERAUTH_REQUEST with a requested method of none.

2. 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.

3. The server returns SSH_MSG_USERAUTH_FAILURE with a list of one or more authentication methods to be used.

4. The client selects one of the acceptable authentication methods and sends a SSH_MSG_USERAUTH_REQUEST with that method name and the required specific fields.

3) Authentication Methods:

The server may require one or more of the following authentication methods like Public key, password , host based signature

SSH Connection Protocol

- It runs on top of SSH Transport Layer Protocol and assumes secure authentication connection is established. That secure authentication connection is called Tunnel.
- Tunnel is used by the connection protocol to multiplex a number of logical channels.
- The life of a channel progresses through three stages: opening a channel, data transfer, and closing a channel.

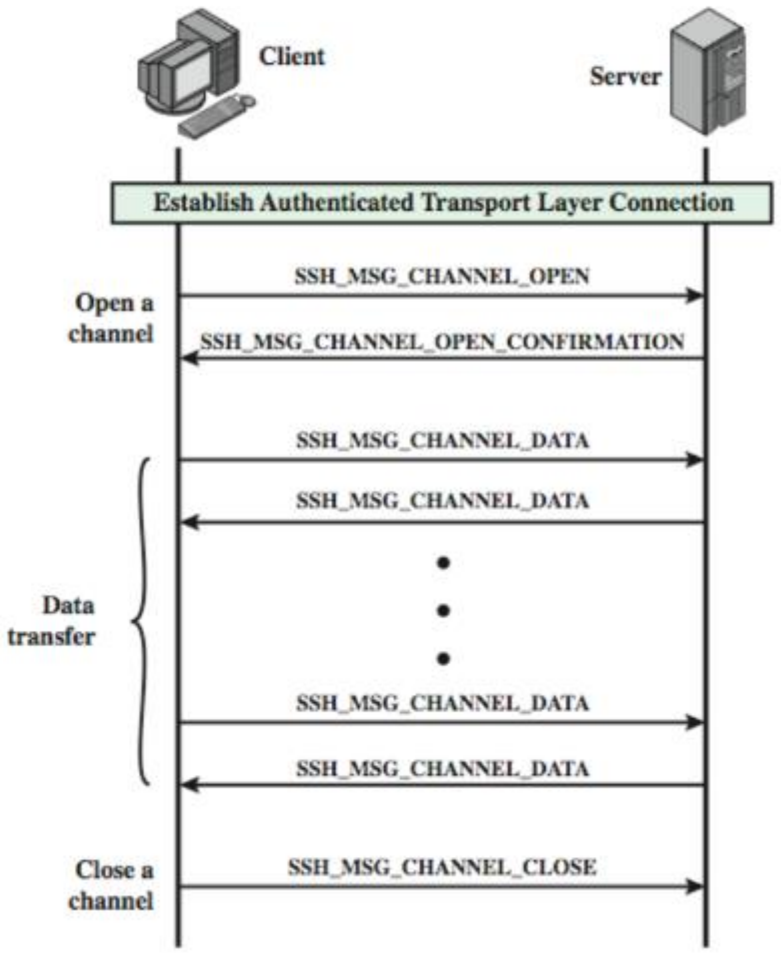
Channel Mechanism All types of communication using SSH, such as a terminal session, are supported using separate channels. Either side may open a channel. For each channel, each side associates a unique channel number, which need not be the same on both ends. Channels are flow controlled using a window 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.

- When either side wishes to **open a new channel,** it allocates a local number for the channel and then sends a message of the form:

byte SSH_MSG_CHANNEL_OPEN string channel type

- If the remote side is able to open the channel, it returns a SSH_MSG_ CHANNEL_ OPEN_ CONFIRMATION message, which includes the sender channel number, the recipient channel number, and window and packet size values for incoming traffic. Otherwise, the remote side returns a SSH_MSG_CHANNEL_OPEN_FAILURE message with a reason code indicating the reason for failure.
- Once a channel is open, **data transfer** is performed using a SSH_MSG_CHANNEL_DATA message, which includes the recipient channel number and a block of data. These messages, in both directions, may continue as long as the channel is open.
- When either side wishes to **close a channel,** it sends a SSH_MSG_CHANNEL_CLOSE message, which includes the recipient channel number. Figure 17.11 provides an example.



Port Forwarding

- One of the most useful features of SSH is port forwarding.
- In essence, port forwarding provides the ability to convert any insecure TCP connection into a secure SSH connection. This is also referred to as SSH tunneling.
- A **port** is an identifier of a user of TCP. So, any application that runs on top of TCP has a port number. Incoming TCP traffic is delivered to the appropriate application on the basis of the port number.
- An application may employ multiple port numbers. For example, for the Simple Mail Transfer Protocol (smtp), the server side generally listens on port 25, so an incoming SMTP request uses TCP and addresses the data to destination port 25.
- TCP recognizes that this is the SMTP server address and routes the data to the SMTP server application.

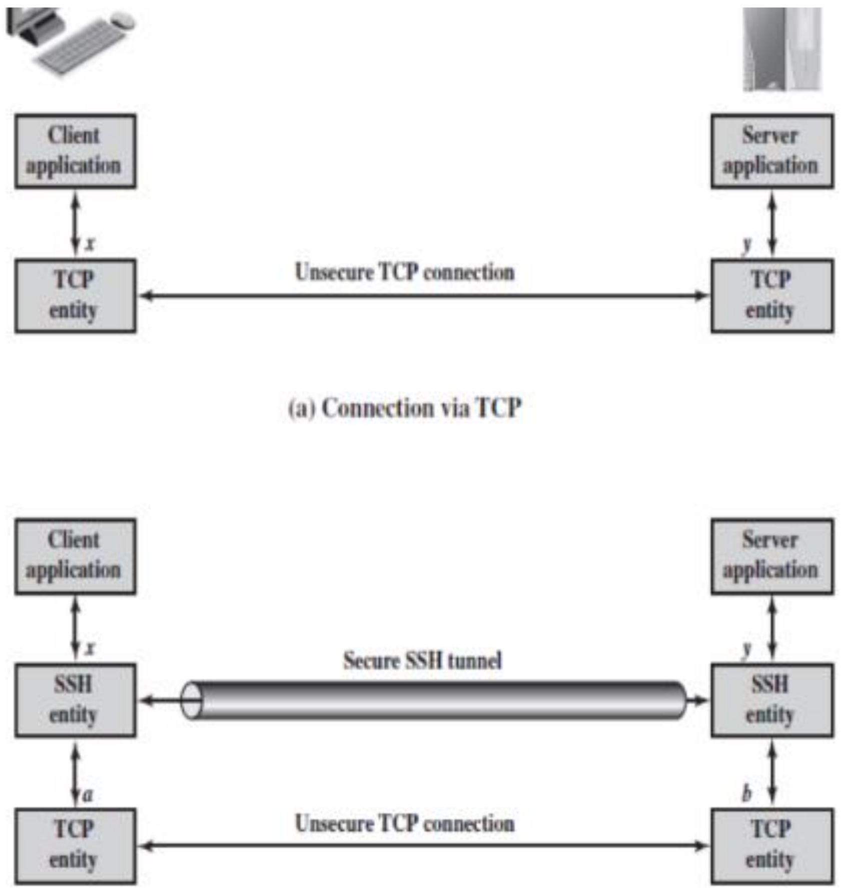
Figure 17.12 illustrates the basic concept behind port forwarding.

SSH supports two types of port forwarding: local forwarding and remote forwarding.

(a) **Local forwarding** allows the client to set up a “hijacker” process. This will intercept selected application-level traffic and redirect it from an unsecured TCP connection to a secure SSH tunnel. SSH is configured to listen on selected ports. SSH grabs all traffic using a selected port and sends it through an SSH tunnel. On the other end, the SSH server sends the incoming traffic to the destination port dictated by the client application.

The following example should help clarify local forwarding. Suppose you have an e-mail client on your desktop and use it to get e-mail from your mail server via the Post Office Protocol (POP). The assigned port number for POP3 is port 110. We can secure this traffic in the following way:

1. The SSH client sets up a connection to the remote server.
2. Select an unused local port number, say 9999, and configure SSH to accept traffic from this port destined for port 110 on the server.
3. The SSH client informs the SSH server to create a connection to the destination, in this case mailserver port 110.
4. The client takes any bits sent to local port 9999 and sends them to the server inside the encrypted SSH session. The SSH server decrypts the incoming bits and sends the plaintext to port 110.
5. In the other direction, the SSH server takes any bits received on port 110 and sends them inside the SSH session back to the client, who decrypts and sends them to the process connected to port 9999.



(b) Connection via SSH tunnel

Figure 17.12 SSH Transport Layer Packet Exchanges

(b)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.

A typical example of remote forwarding is the following. You wish to access a server at work from your home computer. Because the work server is behind a firewall, it will not accept an SSH request from your home computer. However, from work you can set up an SSH tunnel using remote forwarding. This involves the following steps.

1. From the work computer, set up an SSH connection to your home computer. The firewall will allow this, because it is a protected outgoing connection.

2. Configure the SSH server to listen on a local port, say 22, and to deliver data across the SSH connection addressed to remote port, say 2222.
3. You can now go to your home computer, and configure SSH to accept traffic on port 2222.
4. You now have an SSH tunnel that can be used for remote logon to the work server.

Wireless Network security

Why Wireless is Insecure?

Channel: Broadcast ⇒ Eavesdropping, Jamming, Active attacks on protocols

Mobility: Portable devices ⇒ Not physically secured

Resources: Limited memory and processing resources⇒ Need simpler security

Accessibility: May be left unattended

Wireless Network Threats

1. Accidental Association: Overlapping networks ⇒ unintentionally connect to neighbors
2. Malicious Association: Malicious access points (Free public WiFi) can steal passwords
3. Ad-Hoc Networks: Two computers can exchange data
4. Nontraditional Networks: Bluetooth can be used to eavesdrop
5. MAC Spoofing: Change MAC address to match a privileged computer
6. Man-In-The-Middle Attacks: Using rogue access point between the user and the real access point
7. Denial of Service (DoS): Keep the media busy
8. Network Injection: Spoof routing/management messages

Wireless Security Measures

wireless security measures are :

• **Signal-hiding techniques:** Organizations can take a number of measures to make it more difficult for an attacker to locate their wireless access points, including turning off service set identifier (SSID) broadcasting by wireless access points; assigning cryptic names to SSIDs; reducing signal strength to the lowest level that still provides requisite coverage; and locating wireless access points

in the interior of the building, away from windows and exterior walls. Greater security can be achieved by the use of directional antennas and of signal-shielding techniques.

• **Encryption:** Encryption of all wireless transmission is effective against eavesdropping to the extent that the encryption keys are secured. The use of encryption and authentication protocols is the standard method of countering attempts to alter or insert transmissions.

Mobile Device security

Mobile Device Security

Mobile Security Threats

1. Lack of Physical security: Mobiles cannot be locked
2. Not all devices can be trusted
3. Untrusted networks between device and the organization
4. Wide variety of contents on mobiles than on other computers (music, video, games, ...)
5. Apps from untrusted vendors
6. Data may get on unsecured device
7. Location information may be used for attack

Mobile Device Security Strategy

With the threats listed in the preceding discussion in mind, we outline the principal elements of a mobile device security strategy. They fall into three categories: device security, client/server traffic security, and barrier security (Figure 18.2).

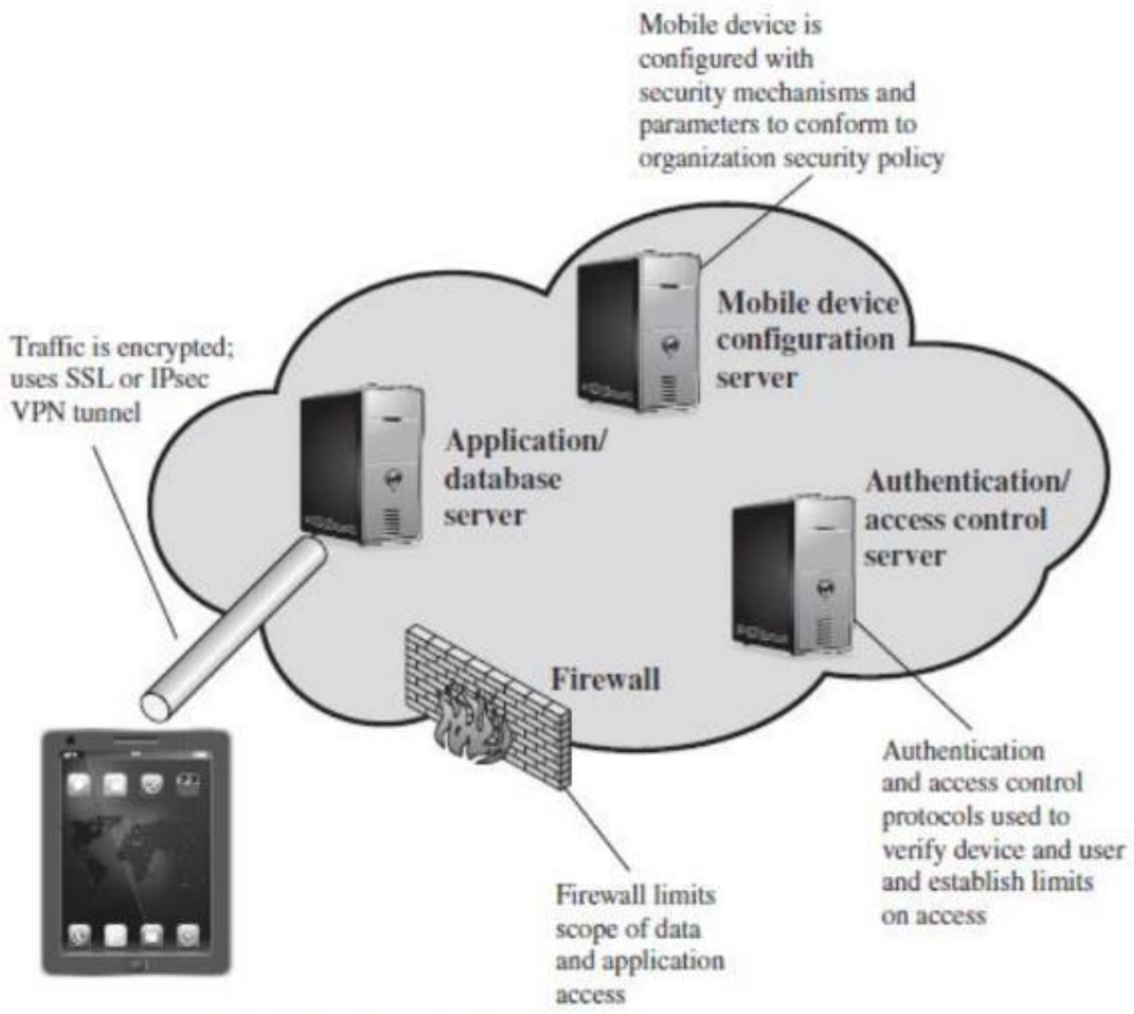

Mobile device is configured with security mechanisms and parameters to conform to organization security policy
Traffic is encrypted; uses SSL or IPsec VPN tunnel
Mobile device configuration server
Application/ database server
Authentication/ access control server
Firewall
Authentication and access control protocols used to verify device and user and establish limits on access
Firewall limits scope of data and application access

Figure 18.2 Mobile Device Security Elements

Device Security

- A number of organizations will supply mobile devices for employee use and preconfigure those devices to conform to the enterprise security policy.
- IT managers should be able to inspect each device before allowing network access. IT will want to establish configuration guidelines for operating systems and applications. For example, "rooted" or "jail-broken" devices are not permitted on the network, and mobile devices cannot store corporate contacts on local storage.
- Whether a device is owned by the organization or BYOD, the organization should configure the device with security controls, including the following:

- Enable auto-lock, which causes the device to lock if it has not been used for a given amount of time, requiring the user to re-enter a four-digit PIN or a password to re-activate the device.
- Enable password or PIN protection. The PIN or password is needed to unlock the device. In addition, it can be configured so that e-mail and other data on the device are encrypted using the PIN or password and can only be retrieved with the PIN or password.
- Avoid using auto-complete features that remember user names or passwords.
- Ensure that SSL protection is enabled, if available.

    - Make sure that software, including operating systems and applications, is upto date.
    - Install antivirus software as it becomes available.
    - Either sensitive data should be prohibited from storage on the mobile device or it should be encrypted.

• The organization may prohibit all installation of third-party applications, implement whitelisting to prohibit installation of all unapproved applications, or implement a secure sandbox that isolates the organization's data and applications from all other data and applications on the mobile device. Any application that is on an approved list should be accompanied by a digital signature and a public-key certificate from an approved authority.

• The organization can implement and enforce restrictions on what devices can synchronize and on the use of cloud-based storage.

• To deal with the threat of untrusted content, security responses can include training of personnel on the risks inherent in untrusted content and disabling camera use on corporate mobile devices.

Traffic Security

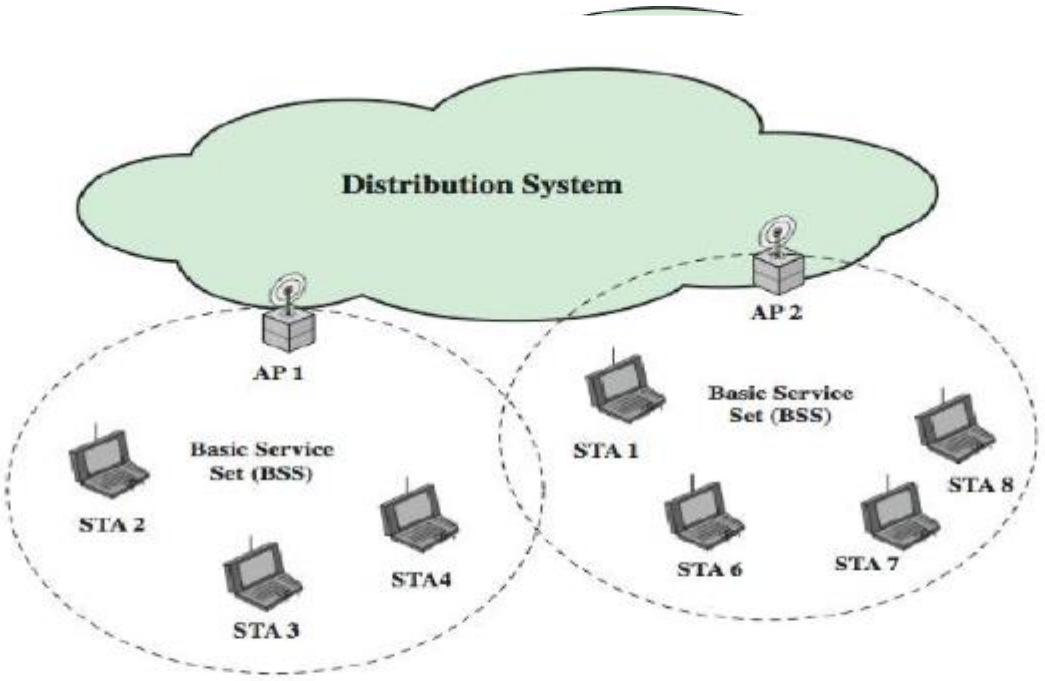
Traffic security is based on the usual mechanisms for encryption and authentication. All traffic should be encrypted and travel by secure means, such as SSL or IPv6. Virtual private networks (VPNs) can be configured so that all traffic between the mobile device and the organization's network is via a VPN. A strong authentication protocol should be used to limit the access from the device to the resources of the organization. Often, a mobile device has a single device-specific authenticator, because it is assumed that the device has only one user. A preferable strategy is to have a two-layer authentication mechanism, which involves authenticating the device and then authenticating the user of the device.

Barrier Security

The organization should have security mechanisms to protect the network from unauthorized access. The security strategy can also include firewall policies specific to mobile device traffic. Firewall policies can limit the scope of data and application access for all mobile devices. Similarly, intrusion detection and intrusion prevention systems can be configured to have tighter rules for mobile device traffic.

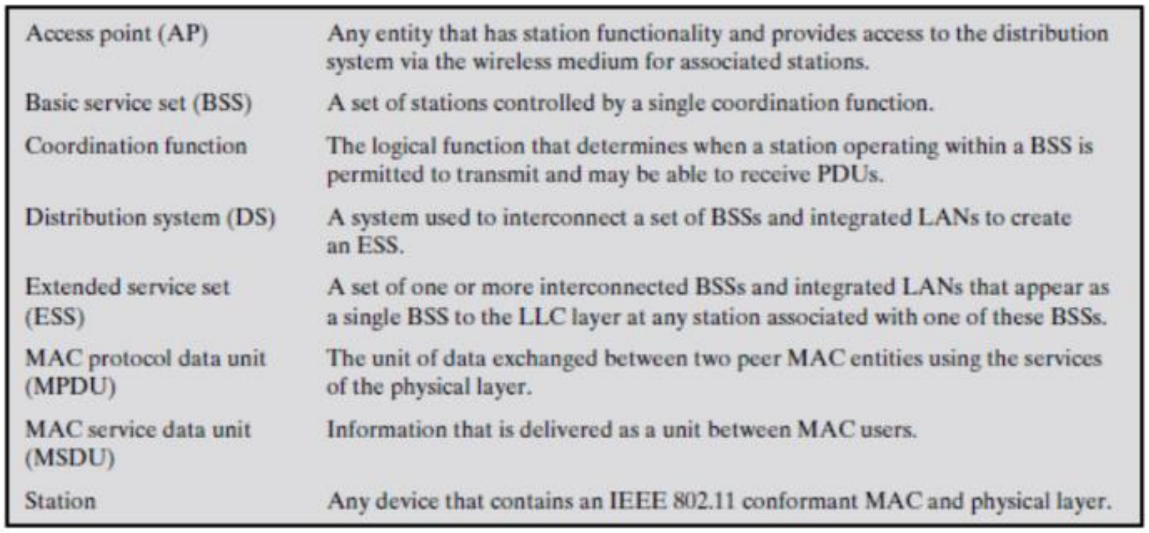
IEEE 802.11 Wireless LAN overview

- IEEE 802 is a committee that has developed standards for a local area networks (LANs).
- In 1990, the IEEE 802 Committee formed a new working group, IEEE 802.11, with a charter to develop a protocol and transmission specifications for wireless LANs (WLANs).
- Since that time, the demand for WLANs, at different frequencies and data rates, has exploded. Keeping pace with this demand, the IEEE 802.11 working group has issued an ever-expanding list of standards.



IEEE 802 Terminology

Table 18.1 IEEE 802.11 Terminology



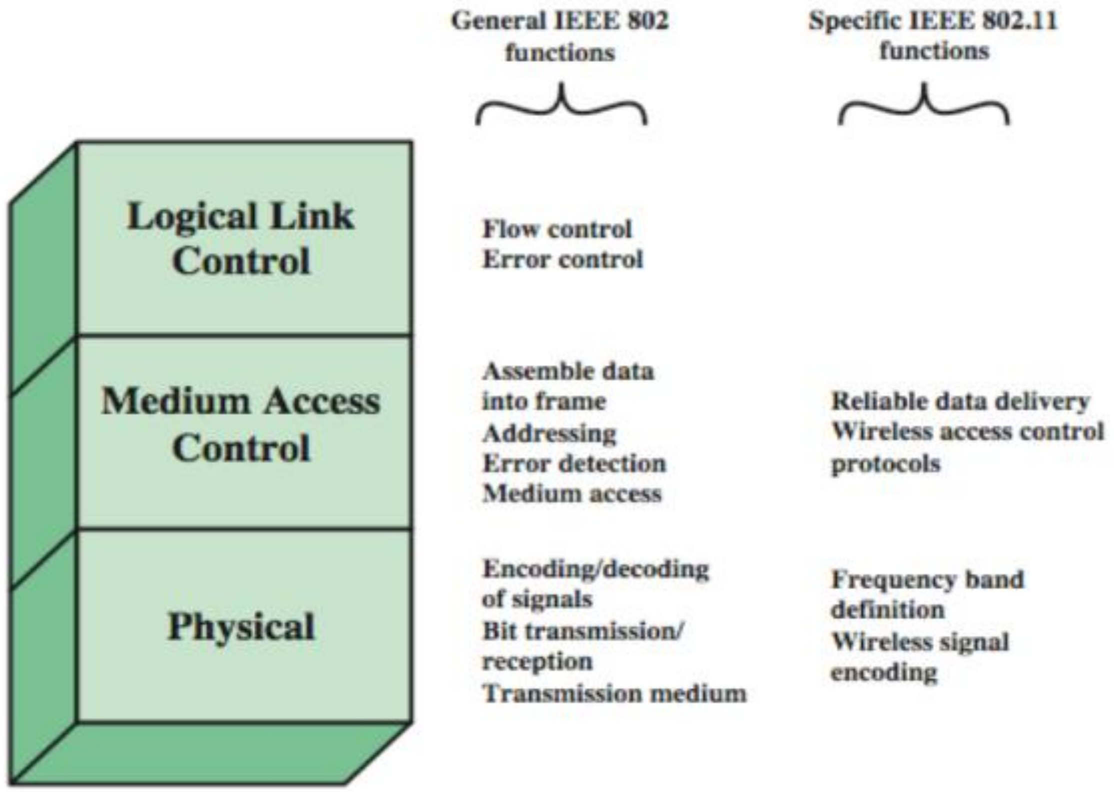
Access point (AP)	Any entity that has station functionality and provides access to the distribution system via the wireless medium for associated stations.
Basic service set (BSS)	A set of stations controlled by a single coordination function.
Coordination function	The logical function that determines when a station operating within a BSS is permitted to transmit and may be able to receive PDUs.
Distribution system (DS)	A system used to interconnect a set of BSSs and integrated LANs to create an ESS.
Extended service set (ESS)	A set of one or more interconnected BSSs and integrated LANs that appear as a single BSS to the LLC layer at any station associated with one of these BSSs.
MAC protocol data unit (MPDU)	The unit of data exchanged between two peer MAC entities using the services of the physical layer.
MAC service data unit (MSDU)	Information that is delivered as a unit between MAC users.
Station	Any device that contains an IEEE 802.11 conformant MAC and physical layer.

Wi-Fi Alliance

- The first 802.11 standard to gain broad industry acceptance was 802.11b. Although 802.11b products are all based on the same standard, there is always a concern whether products from different vendors will successfully interoperate.
- To meet this concern, the Wireless Ethernet Compatibility Alliance (WECA), an industry consortium, was formed in 1999. T
- his organization, subsequently renamed the Wi-Fi (Wireless Fidelity) Alliance, created a test suite to certify interoperability for 802.11b products.
- The term used for certified 802.11b products is *Wi- Fi*. Wi-Fi certification has been extended to 802.11g products,. The Wi-Fi Alliance has also developed a certification process for 802.11a products, called *Wi-Fi5*.
- The Wi-Fi Alliance is concerned with a range of market areas for WLANs, including enterprise, home, and hot spots.

IEEE 802 Protocol Architecture

we need to briefly preview the IEEE 802 protocol architecture. IEEE 802.11 standards are defined within the structure of a layered set of protocols. This structure, used for all IEEE 802 standards, is illustrated in Stallings Figure 17.1.



The lowest layer of the IEEE 802 reference model is the physical layer, which includes such functions as encoding/decoding of signals and bit transmission/reception. In addition, the physical layer includes a specification of the transmission medium. In the case of IEEE 802.11, the physical layer also defines frequency bands and antenna characteristics.

Next is the media access control (MAC) layer, which controls access to the transmission medium to provide an orderly and efficient use of that capacity. The MAC layer receives data from a higher-layer protocol, typically the Logical Link Control (LLC) layer, in the form of a block of data known as the MAC service data unit (MSDU). The exact format of the MPDU differs somewhat for the various MAC protocols in use.

In most data link control protocols, the data link protocol entity is responsible not only for detecting errors using the CRC, but for recovering from those errors by retransmitting damaged frames. In the LAN protocol architecture, these two functions are split between the MAC and LLC layers. The MAC layer is responsible for detecting errors and discarding any frames that contain errors. The LLC layer optionally keeps track of which frames have been successfully received and retransmits unsuccessful frames.

IEEE 802 .11 Network Components & Architecture

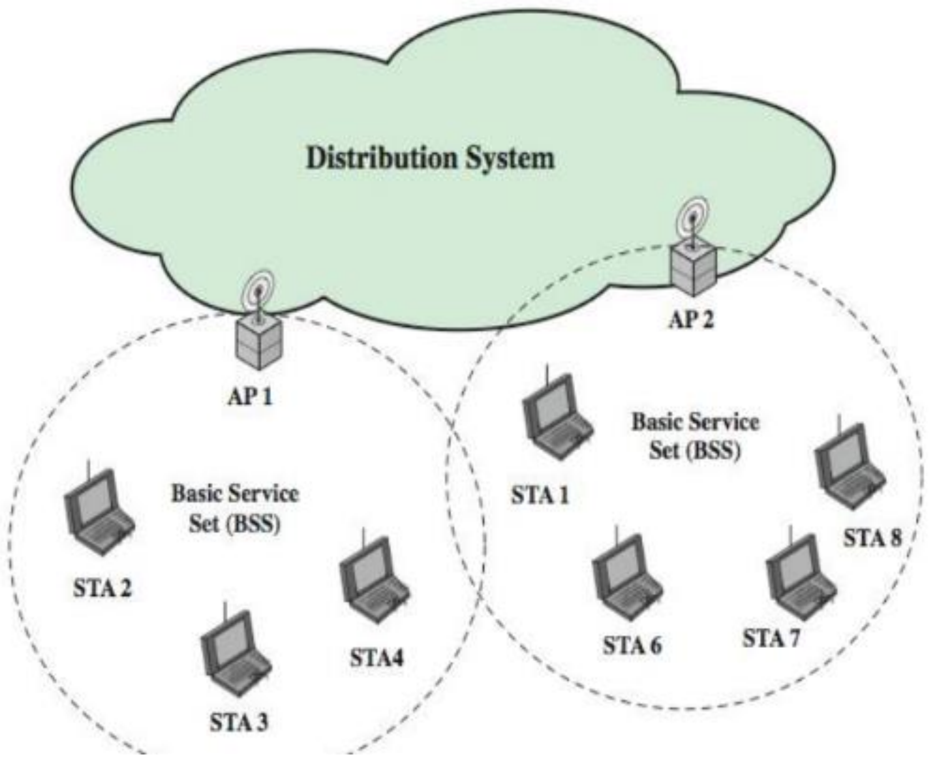


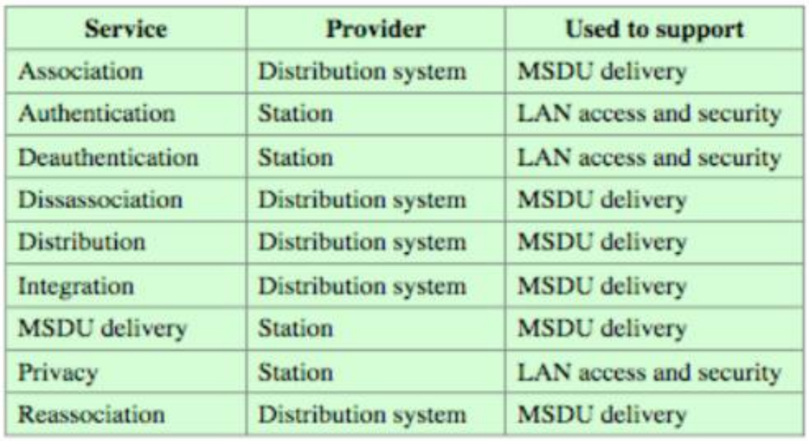
Figure 17.3 illustrates the model developed by the 802.11 working group.

- The smallest building block of a wireless LAN is a **basic service set (BSS)**, which consists of wireless stations executing the same MAC protocol and competing for access to the same shared wireless medium.
- A BSS may be isolated or it may connect to a backbone **distribution system (DS)** through an **access point (AP)**. The AP functions as a bridge and a relay point.
- In a BSS, client stations do not communicate directly with one another. Rather the MAC frame is first sent from the originating station to the AP, and then from the AP to the destination station. Similarly, a MAC frame from a station in the BSS to a remote station is sent from the local station to the AP and then relayed by the AP over the DS on its way to the destination station.
- The BSS generally corresponds to what is referred to as a cell. The DS can be a switch, a wired network, or a wireless network.
- When all the stations in the BSS are mobile stations that communicate directly with one another, not using an AP, the BSS is called an **independent BSS (IBSS)**. An IBSS is typically an ad hoc network. In an IBSS, the stations all communicate directly, and no AP is involved.

• A simple configuration is shown in Figure 17.3, in which each station belongs to a single BSS; that is, each station is within wireless range only of other stations within the same BSS. It is also possible for two BSSs to overlap geographically, so that a single station could participate in more than one BSS. Further, the association between a station and a BSS is dynamic. Stations may turn off, come within range, and go out of range.

• An **extended service set (ESS)** consists of two or more basic service sets interconnected by a distribution system. The extended service set appears as a single logical LAN to the logical link control (LLC) level.

IEEE 802.11 Services



Service	Provider	Used to support
Association	Distribution system	MSDU delivery
Authentication	Station	LAN access and security
Deauthentication	Station	LAN access and security
Dissassociation	Distribution system	MSDU delivery
Distribution	Distribution system	MSDU delivery
Integration	Distribution system	MSDU delivery
MSDU delivery	Station	MSDU delivery
Privacy	Station	LAN access and security
Reassociation	Distribution system	MSDU delivery

IEEE 802.11 defines nine services that need to be provided by the wireless LAN to achieve functionality equivalent to that which is inherent to wired LANs. Stallings Table 17.2 lists the services & notes two ways of categorizing them.

1. The service provider can be either the station or the DS. Station services are implemented in every 802.11 station, including AP stations. Distribution services are provided between BSSs; these may be implemented in an AP or in another special-purpose device attached to the distribution system.

2. Three of the services are used to control IEEE 802.11 LAN access and confidentiality. Six of the services are used to support delivery of MSDUs between stations. If the MSDU is too large to be transmitted in a single MPDU, it may be fragmented and transmitted in a series of MPDUs.

Distribution is the primary service used by stations to exchange MPDUs when the MPDUs must traverse the DS to get from a station in one BSS to a station in another BSS.

Integration enables transfer of data between a station on an IEEE 802.11 LAN and a station on an integrated (wired) IEEE 802.x LAN. To deliver a message within a DS, the distribution service needs to know where the destination station is located.

Association establishes an initial association between a station and an AP.

Reassociation enables an established association to be transferred from one AP to another, allowing a mobile station to move from one BSS to another.

Disassociation is a notification from either a station or an AP that an existing association is terminated.

802.11i Wireless LAN Security

The differences between wired and wireless LANs (in that wireless traffic can be monitored by any radio in range, and need not be physically connected) suggest the increased need for robust security services and mechanisms for wireless LANs.

The original 802.11 specification included a set of security features for privacy and authentication that were quite weak. For privacy, 802.11 defined the **Wired Equivalent Privacy (WEP)** algorithm. The privacy portion of the 802.11 standard contained major weaknesses. Subsequent to the development of WEP, the 802.11i task group has developed a set of capabilities to address the WLAN security issues. In order to accelerate the introduction of strong security into WLANs, the Wi-Fi Alliance promulgated **Wi-Fi Protected Access (WPA)** as a Wi-Fi standard. WPA is a set of security mechanisms that eliminates most 802.11 security issues and was based on the current state of the 802.11i standard. The final form of the 802.11i standard is referred to as **Robust Security Network (RSN)**. The Wi-Fi Alliance certifies vendors in compliance with the full 802.11i specification under the WPA 2 program.

<u>802.11i RSN Services and Protocols</u>

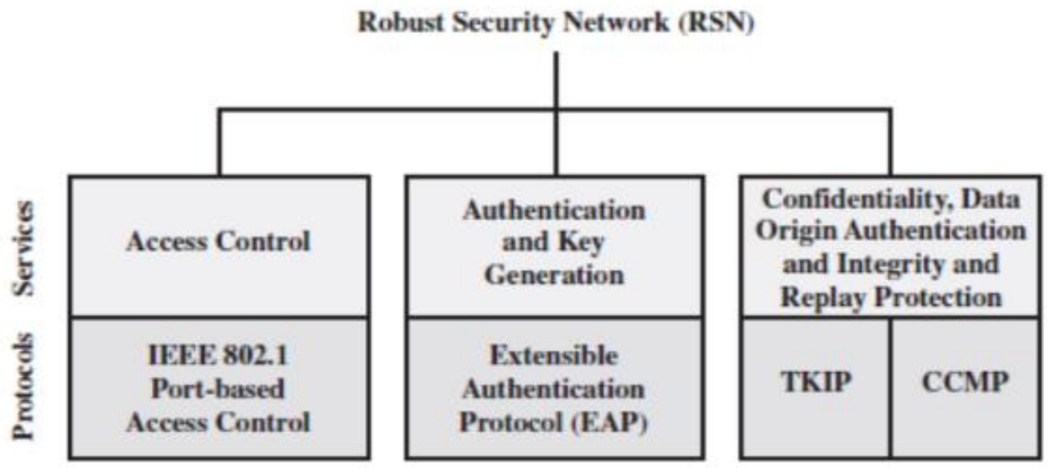
The 802.11i RSN security specification defines the following services:

• Authentication: A protocol is used to define an exchange between a user and an AS that provides mutual authentication and generates temporary keys to be used between the client and the AP over the wireless link.

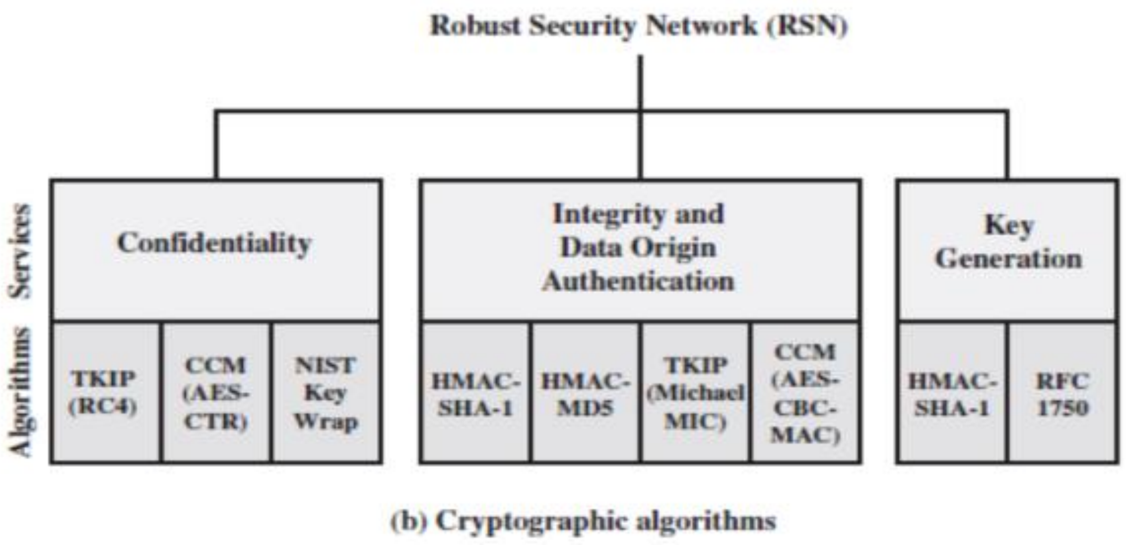
• Access control: This function enforces the use of the authentication function, routes the messages properly, and facilitates key exchange. It can work with a variety of authentication protocols.

• Privacy with message integrity: MAC-level data (e.g., an LLC PDU) are encrypted, along with a message integrity code that ensures that the data have not been altered.

Figure 17.4a indicates the security protocols used to support these services.


Robust Security Network (RSN)
Services
Protocols
Access Control
Authentication and Key Generation
Confidentiality, Data Origin Authentication and Integrity and Replay Protection
IEEE 802.1 Port-based Access Control
Extensible Authentication Protocol (EAP)
TKIP
CCMP

(a) Services and protocols



CBC-MAC	=	Cipher Block Chaining Message Authentication Code (MAC)
CCM	=	Counter Mode with Cipher Block Chaining Message Authentication Code
CCMP	=	Counter Mode with Cipher Block Chaining MAC Protocol
TKIP	=	Temporal Key Integrity Protocol

Figure 18.6 Elements of IEEE 802.11i

802.11i Phases of Operation

The operation of an IEEE 802.11i RSN can be broken down into five distinct phases of operation, as shown in Stallings Figure 17.5. One new component is the authentication server (AS). The five phase are:

• **Discovery:** An AP uses messages called Beacons and Probe Responses to advertise its IEEE 802.11i security policy. The STA uses these to identify an AP for a WLAN with which it wishes to communicate. The STA associates with the AP, which it uses to select the cipher suite and authentication mechanism when the Beacons and Probe Responses present a choice.

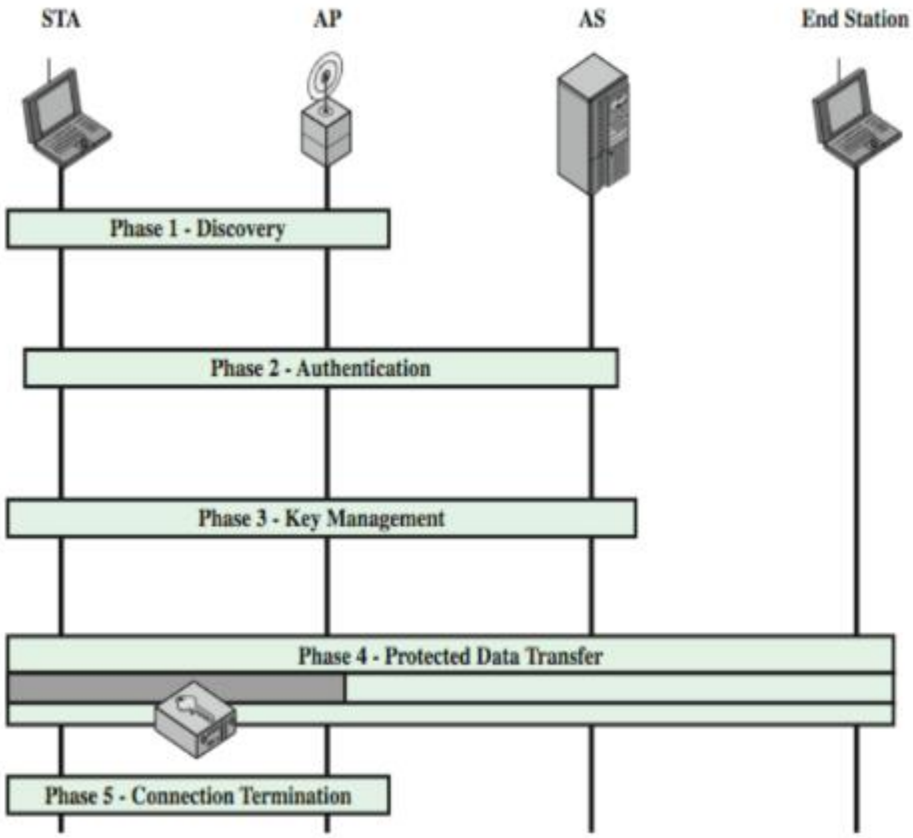
• **Authentication:** During this phase, the STA and AS prove their identities to each other. The AP blocks non-authentication traffic between the STA and AS until the authentication transaction is successful. The

AP does not participate in the authentication transaction other than forwarding traffic between the STA and AS.

• **Key generation and distribution**: The AP and the STA perform several operations that cause cryptographic keys to be generated and placed on the AP and the STA. Frames are exchanged between the AP and STA only

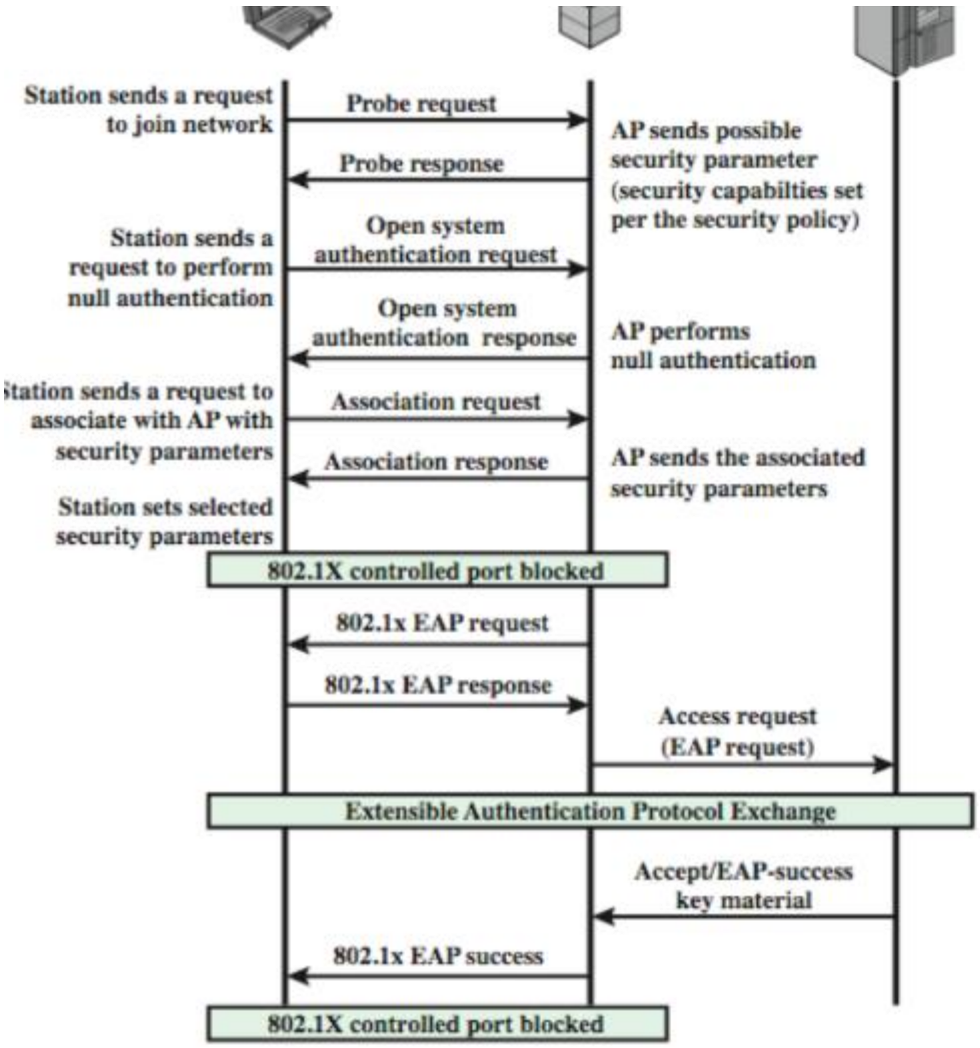
• **Protected data transfer**: Frames are exchanged between the STA and the end station through the AP. As denoted by the shading and the encryption module icon, secure data transfer occurs between the STA and the AP only; security is not provided end-to-end.

• **Connection termination**: The AP and STA exchange frames. During this phase, the secure connection is torn down and the connection is restored to the original state.



Discovery and Authentication Phases

We now look in more detail at the RSN phases of operation, beginning with the discovery phase, which is illustrated in the upper portion of Figure 17.6.



Station sends a request to join network
Probe request
AP sends possible security parameter (security capabilties set per the security policy)
Probe response
Station sends a request to perform null authentication
Open system authentication request
Open system authentication response
AP performs null authentication
Station sends a request to associate with AP with security parameters
Association request
Association response
AP sends the associated security parameters
Station sets selected security parameters
802.1X controlled port blocked
802.1x EAP request
802.1x EAP response
Access request (EAP request)
Extensible Authentication Protocol Exchange
Accept/EAP-success key material
802.1x EAP success
802.1X controlled port blocked

MPDU Exchange The discovery phase consists of three exchanges.

• Network and security capability discovery: During this exchange, STAs discover the existence of a network with which to communicate. The AP either periodically broadcasts its security capabilities (not shown in figure), indicated by RSN IE (Robust Security Network Information Element), in a specific channel through the Beacon frame; or responds to a station's Probe Request through a Probe Response frame. A wireless station may discover available access points and corresponding security capabilities b either passively monitoring the Beacon frames or actively probing every channel.

• Open system authentication: The purpose of this frame sequence, which provides no security, is simply to maintain backward compatibility with the IEEE 802.11 state machine, as implemented in existing IEEE 802.11 hardware. In essence, the two devices (STA and AP) simply exchange identifiers.

• Association: The purpose of this stage is to agree on a set of security capabilities to be used. The STA then sends an Association Request frame to the AP. In this frame, the STA specifies one set of matching capabilities (one authentication and key management suite, one pairwise cipher suite, and one group-ke cipher suite) from among those advertised by the AP. If thereis no match in capabilities between the AP and the STA, the AP refuses the Association Request. The STA blocks it too, in case it has associated with a rogue AP or someone is inserting frames illicitly on its channel. As shown in Figure 18.8, the IEEE 802.1X controlled ports are blocked, and no user traffic goes beyond the AP. The concept of blocked por is explained subsequently.

Authentication Phase : A s mentioned, there are a number of possible EAP exchanges that can b used during the authentication phase. Typically, the message flow between STA and AP employs the EA over LAN (EAPOL) protocol, and the message flow between the AP and AS uses the Remote Authentication Dial In User Service (RADIUS) protocol, although other options are available for both STA-to-AP and AP-to-AS exchanges. [FRAN07] provides the following summary of the authentication exchange using EAPOL and RADIUS.

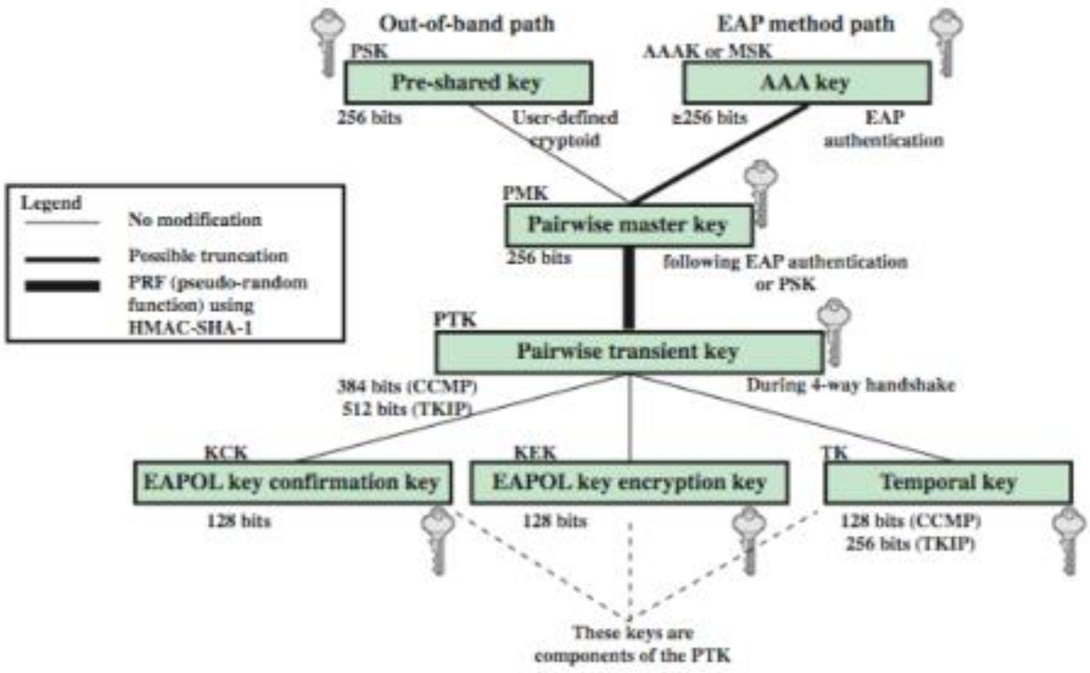
1. The EAP exchange begins with the AP issuing an EAP-Request/Identity frame to the STA.
2. The STA replies with an EAP-Response/Identity frame, which the AP receives over the uncontrolled port. The packet is then encapsulated in RADIUS over EAP and passed on to the RADIUS server as a RADIUS-Access-Request packet.
3. The AAA server replies with a RADIUS-Access-Challenge packet, which is passed on to the STA as a EAP-Request. This request is of the appropriate authentication type and contains relevant challenge information.

4. The STA formulates an EAP-Response message and sends it to the AS. The response is translated b AP into a Radius-Access-Request with the response to the challenge as a data field. Steps 3 and 4 may repeated multiple times, depending on the EAP method in use. For TLS tunneling methods, it is common for authentication to require 10 to 20 round trips. 5. The AAA server grants access with a Radiu Access-Accept packet. The AP issues an EAP-Success frame. (Some protocols require confirmation of t EAP success inside the TLS tunnel for authenticity validation.) The controlled port is authorized, and the user may begin to access the network.

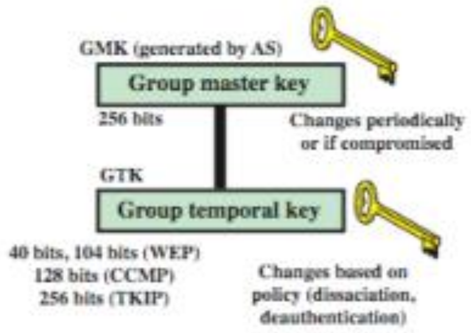
802.11i Key Management Phase

- During the key management phase, a variety of cryptographic keys are generated and distributed to STAs. There are two types of keys: pairwise keys, used for communication between an STA and an AP; and group keys, for multicast communication.
- Figure 17.8 shows the two key hierarchies.
- Pairwise keys are used for communication between a pair of devices, typically between an STA and an AP. These keys form a hierarchy, beginning with a master key from which other keys are derived dynamically and used for a limited period of time.
- A **pre-shared key (PSK)** is a secret key shared by the AP and a STA, and installed in some fashion outside the scope of IEEE 802.11i.
- The other alternative is the **master session key (MSK)**, also known as the AAAK, which is generated using the IEEE 802.1X protocol during the authentication phase, as described previously.

- The **pairwise master key (PMK)** is derived from the master key as follows: If a PSK is used, then the PSK is used as the PMK; if a MSK is used, then the PMK is derived from the MSK by truncation (if necessary).
- By the end of the authentication phase (on EAP Success message), both the AP and the STA have a copy of their shared PMK.
- The PMK is used to generate the **pairwise transient key (PTK)**, which in fact consists of three keys to be used for communication between an STA and AP after they have mutually authenticated.
- To derive the PTK, the PMK, the MAC addresses of the STA and AP, and nonces generated when needed are all input to the HMAC-SHA-1 function.
- Group keys are used for multicast communication when one STA sends MPDU's to multiple STAs.



(a) Pairwise key hierarchy



(b) Group key hierarchy

Pairwise Key Distribution

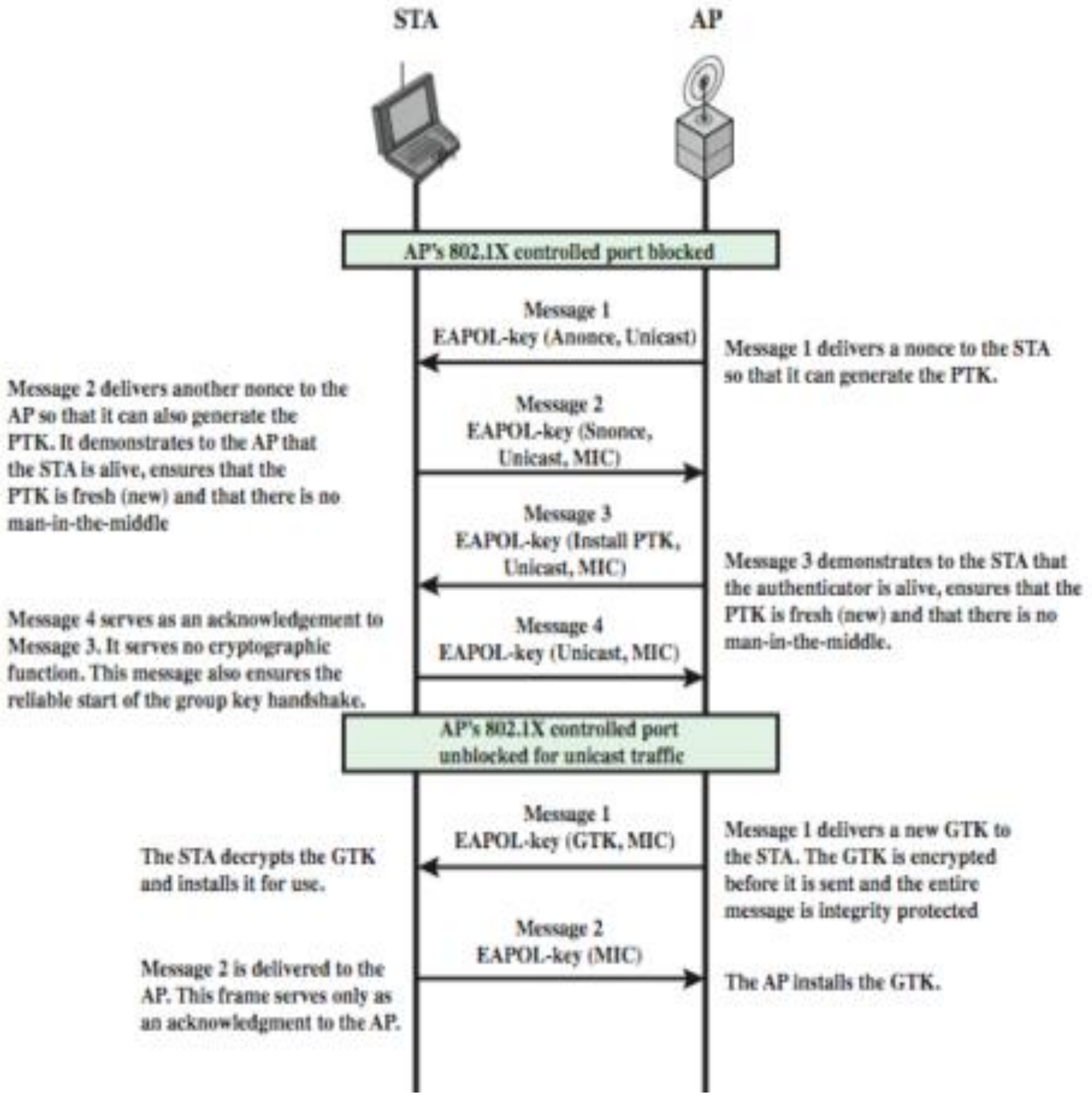
The upper part of Figure 18.10 shows the MPDU exchange for distributing pairwise keys. This exchange is known as the **4-way handshake.** The STA and AP use this handshake to confirm the existence of the PMK, verify the selection of the cipher suite, and derive a fresh PTK for the following data session. The four parts of the exchange are as follows.

- **AP → STA:** Message includes the MAC address of the AP and a nonce (Anonce)
- **STA → AP:** The STA generates its own nonce (Snonce) and uses both nonces and both MAC addresses, plus the PMK, to generate a PTK. The STA then sends a message containing its MAC address and Snonce, enabling the AP to generate the same PTK. This message includes a message integrity code (MIC)[2] using HMAC-MD5 or HMAC-SHA-1-128. The key used with the MIC is KCK.

- **AP → STA:** The AP is now able to generate the PTK. The AP then sends a message to the STA, containing the same information as in the first message, but this time including a MIC.
- **STA → AP:** This is merely an acknowledgment message, again protected by a MIC.

Group Key Distribution For group key distribution, the AP generates a GTK and distributes it to each STA in a multicast group. The two-message exchange with each STA consists of the following:

- **AP → STA:** This message includes the GTK, encrypted either with RC4 or with AES. The key used for encryption is KEK, using a key wrapping algorithm (as discussed in Chapter 12). A MIC value is appended.
- **STA → AP:** The STA acknowledges receipt of the GTK. This message includes a MIC value.


STA
AP
AP's 802.1X controlled port blocked
Message 1
EAPOL-key (Anonce, Unicast)
Message 1 delivers a nonce to the STA so that it can generate the PTK.
Message 2 delivers another nonce to the AP so that it can also generate the PTK. It demonstrates to the AP that the STA is alive, ensures that the PTK is fresh (new) and that there is no man-in-the-middle
Message 2
EAPOL-key (Snonce, Unicast, MIC)
Message 3
EAPOL-key (Install PTK, Unicast, MIC)
Message 3 demonstrates to the STA that the authenticator is alive, ensures that the PTK is fresh (new) and that there is no man-in-the-middle.
Message 4 serves as an acknowledgement to Message 3. It serves no cryptographic function. This message also ensures the reliable start of the group key handshake.
Message 4
EAPOL-key (Unicast, MIC)
AP's 802.1X controlled port unblocked for unicast traffic
Message 1
EAPOL-key (GTK, MIC)
The STA decrypts the GTK and installs it for use.
Message 1 delivers a new GTK to the STA. The GTK is encrypted before it is sent and the entire message is integrity protected
Message 2
EAPOL-key (MIC)
Message 2 is delivered to the AP. This frame serves only as an acknowledgment to the AP.
The AP installs the GTK.

802.11i Protected Data Transfer Phase

- IEEE 802.11i defines two schemes for protecting 802.11 MPDU data message integrity and confidentiality: the Temporal Key Integrity Protocol (TKIP), and the Counter Mode-CBC MAC Protocol (CCMP).
- TKIP is designed to require only software changes to devices that are implemented with the older wireless LAN security approach called Wired Equivalent Privacy (WEP). TKIP adds a 64-bit message integrity code (MIC), generated by an algorithm, called Michael, to the 802.11 MAC frame after the data field. TKIP provides data confidentiality by encrypting the MPDU plus MIC value using RC4.
- CCMP is intended for newer IEEE 802.11 devices that are equipped with the hardware to support this scheme. CCMP uses the cipher block chaining message authentication code (CBC-MAC), described in Chapter 12, to provide message integrity. CCMP uses the CRT block cipher mode of operation, described in Chapter 6, with AES for encryption. The same 128-bit AES key is used for both integrity and confidentiality. The scheme uses a 48-bit packet number to construct a nonce to prevent replay attacks.