ECE Dept.
N&CS
VIII Sem
2018-19

Department of Electronics & Communication Engg.

Course: Network and Cyber Security - 15EC835. **Sem.: 8th (2018-19)**

Course Coordinator:

Prof. B. P. Khot

Web Security

- Web Now Widely Used By Business, Government, Individuals
- But Internet & Web Are Vulnerable
- Have A Variety Of Threats
  - Integrity
  - Confidentiality
  - Denial of Service
  - Authentication
- Need Added Security Mechanisms

Web Security Consideration

- WWW is fundamentally a client/server application running over the internet and TCP/IP

1. Web browser are very easy to use, web browser are relatively easy to configure and manage, and web content is increasingly easy to develop, but underlying software is extraordinarily complex. This complex software may hide many potential security flaws(weakness). That are vulnerable to variety of attacks.
2. Web server can be used as a launching pad to the corporation's or agency's computer. Once the web server is subverted(misused), an attacker may be able to gain access to data and systems connected to the server.
3. Casual and untrained users are common client for web-based services. They are not aware of attacks

Integrity

- **Threats (Damage)**

1. Modification of user data.
2. Trojan horse program.
3. Modification of memory
4. Modification of message traffic.

- **Consequences (Effects)**

1. Loss of information.
2. Compromise of machine(effecting Confidential of machine).
3. Vulnerable to system.

- **Countermeasures (Action Taken)**

1. Cryptographic Check Sum

Confidentiality

- **Threats (Damage)**

1. Theft of information from server.
2. Theft of data from browser.
3. Information about network configuration.
4. Information about which client talks to server.
5. Eavesdropping on the net (secretly observing /listening to network)

- **Consequences (Effects)**

1. Loss of information.
2. Loss of privacy.

- **Countermeasures (Action Taken)**

1. Encryption, web proxies(hide IP and protects online privacy)

Denial of Service

- **Threats (Damage)**

1. Killing of user threads.
2. Flooding machine with bogus requests.
3. Filling up disk or memory.
4. Spreading vulnerability

- **Consequences (Effects)**

1. Disruptive(trouble making).
2. Annoying.
3. Preventing user from getting work done.

- **Countermeasures (Action Taken)**

1. Difficult to prevent.

Authentication

- **Threats (Damage)**

1. Impersonation of legitimate user.
2. Data forgery.

- **Consequences (Effects)**

1. Misrepresentation of user.
2. Belief that false information is valid.

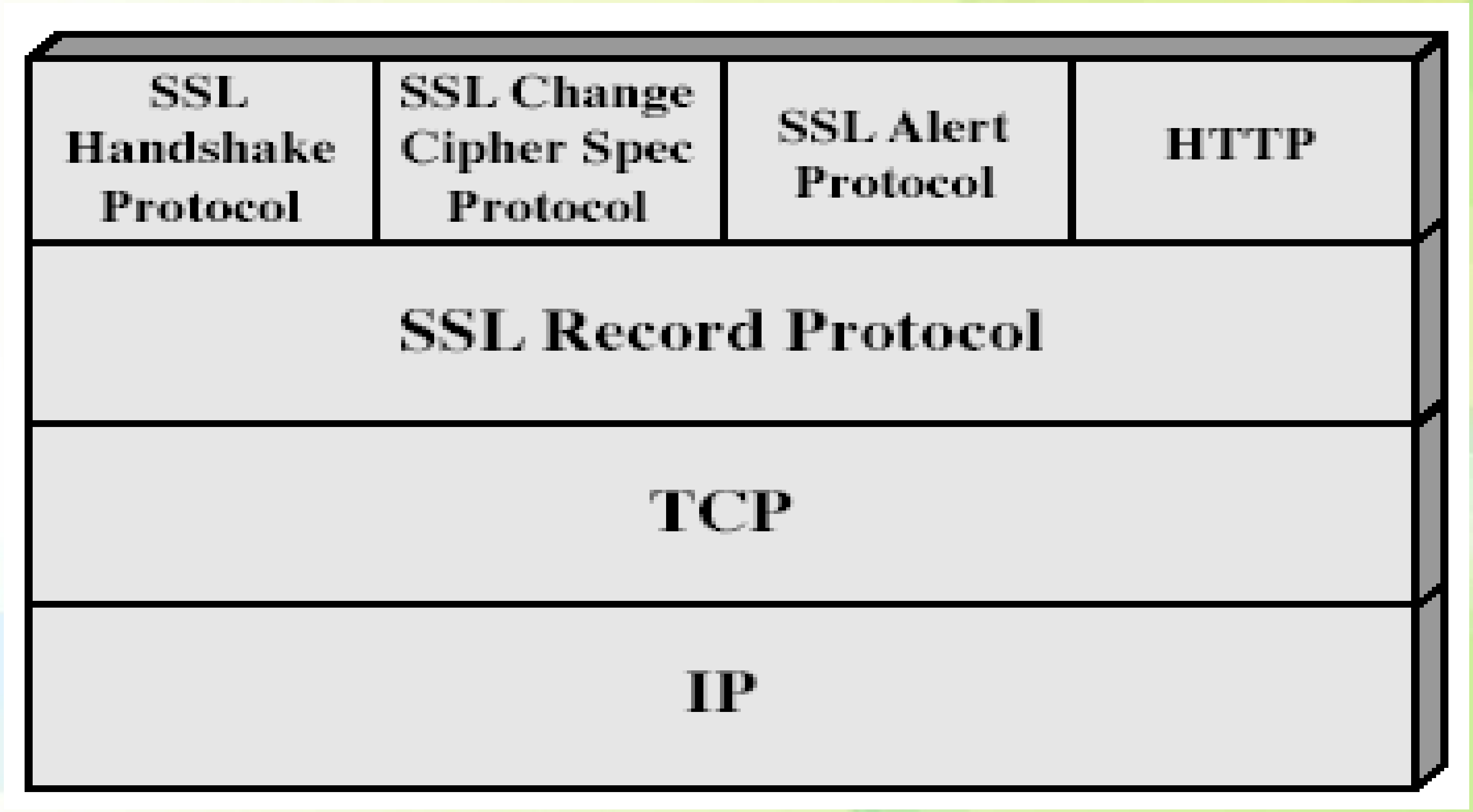
- **Countermeasures (Action Taken)**

1. Cryptographic techniques.

SSL (Secure Socket Layer)

- Most widely used security service
- Transport Layer Security Service(TLS)
- Originally Developed By Netscape
- SSL is a general purpose service provider, implemented as a set of protocol that relay/depend on TCP.
- Subsequently Became Internet Standard Known As TLS (Transport Layer Security)
- Uses TCP to provide a Reliable End-to-end Service
- SSL Has Two Layers Of Protocols

SSL Architecture



SSL Architecture

- SSL Record protocol provides basic security services to various higher layer protocol.
- HTTP provides transfer service for web client/server interaction.
- Three higher layer protocol are defined as part of SSL

1. SSL Handshake protocol
2. SSL Change cipher spec protocol.
3. SSL Alert protocol

- These SSL specific protocols are used in the management of SSL exchanges.

SSL Architecture

Two important SSL concept are

- **SSL session**
  - an association between client & server
  - created by the Handshake Protocol
  - define a set of cryptographic parameters
  - may be shared by multiple SSL connections
- **SSL connection**
  - a transient, peer-to-peer, communications link
  - associated with 1 SSL session

SSL Architecture

SSL connection

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

SSL session

- An association between a client and a server
- Created by the Handshake Protocol
- Define a set of cryptographic security parameters which can be shared among multiple connections
- Are used to avoid the expensive negotiation of new security parameters for each connection

A session state is defined by the following parameters

Session identifier	Peer certificate	Compression method	Cipher spec	Master secret	Is resumable
An arbitrary byte sequence chosen by the server to identify an active or resumable session state	An X509.v3 certificate of the peer; this element of the state may be null	The algorithm used to compress data prior to encryption	Specifies the bulk data encryption algorithm and a hash algorithm used for MAC calculation; also defines cryptographic attributes such as the hash_size	48-byte secret shared between the client and the server	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

Client write MAC secret

- The secret key used in MAC operations on data sent by the client

Server write MAC secret

- The secret key used in MAC operations on data sent by the server

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

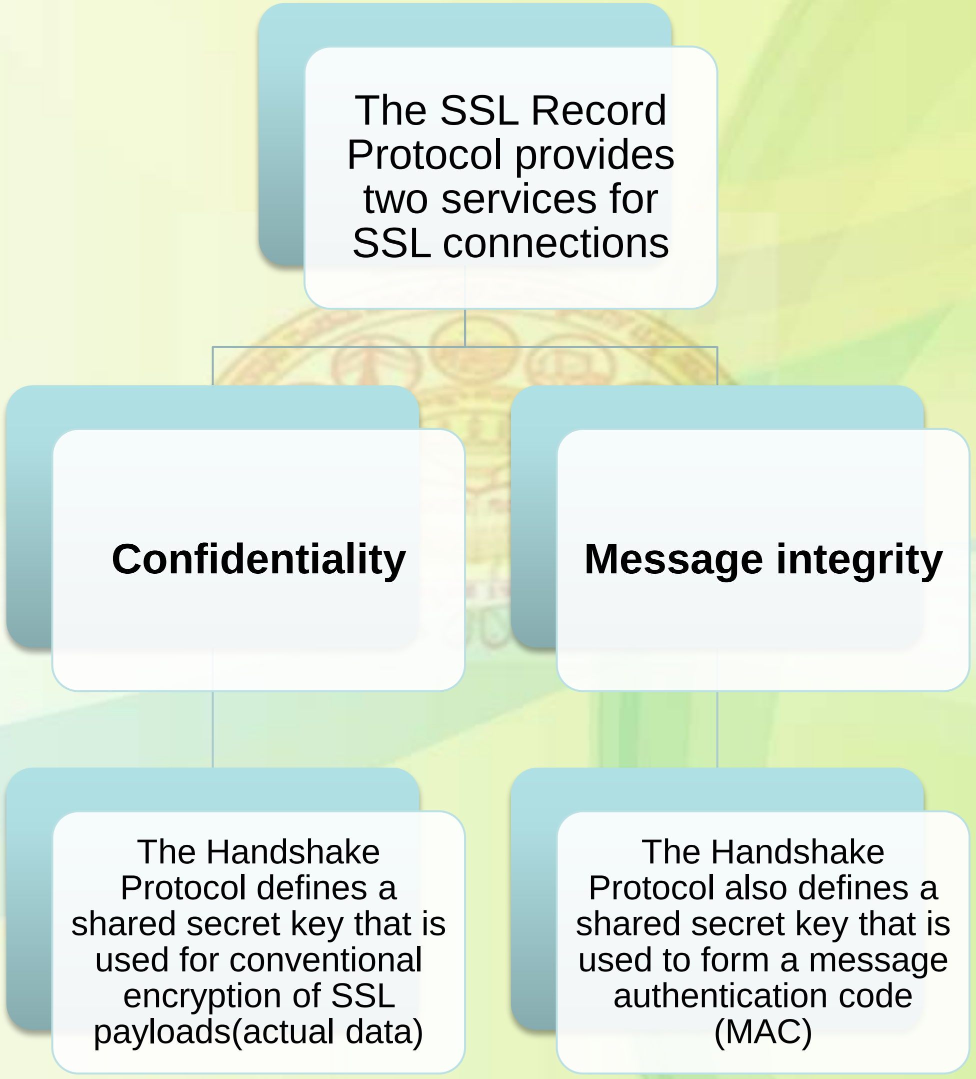
Initialization vectors

- For a block cipher in CBC mode , an initialization vector (IV) used for each key.
- This field is first initialized by the SSL Handshake Protocol
- The final ciphertext block from each record is preserved by IV

Sequence numbers

- Each party maintains separate sequence numbers for transmitted and received messages for each connection
- When a party sends or receives a change cipher spec message, the appropriate sequence number is set to zero
- Sequence numbers may not exceed $2^{64} - 1$

SSL Record Protocol



The SSL Record Protocol provides two services for SSL connections

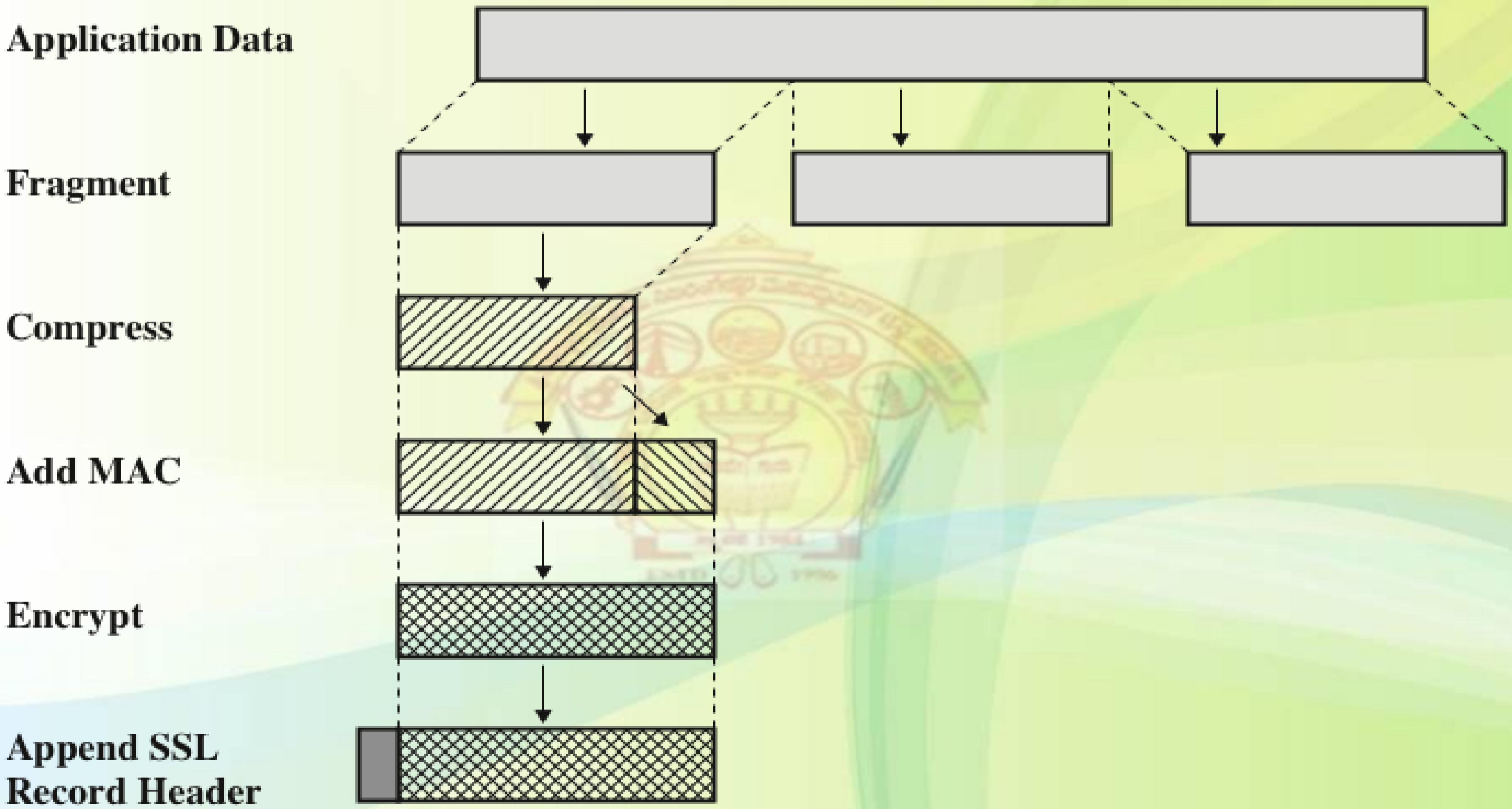
Confidentiality

The Handshake Protocol defines a shared secret key that is used for conventional encryption of SSL payloads(actual data)

Message integrity

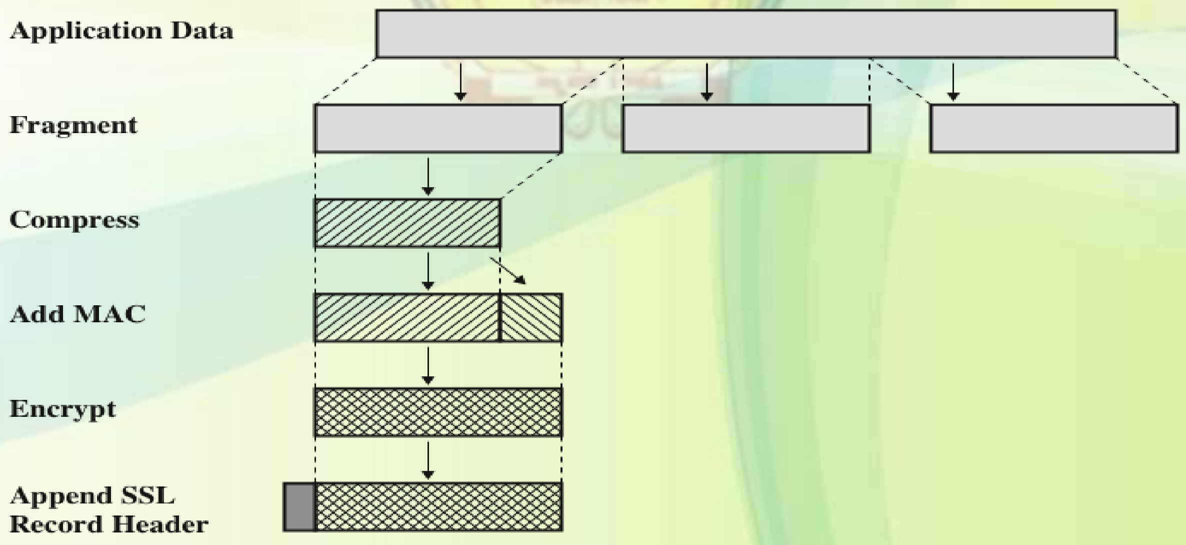
The Handshake Protocol also defines a shared secret key that is used to form a message authentication code (MAC)

SSL Record Protocol operation

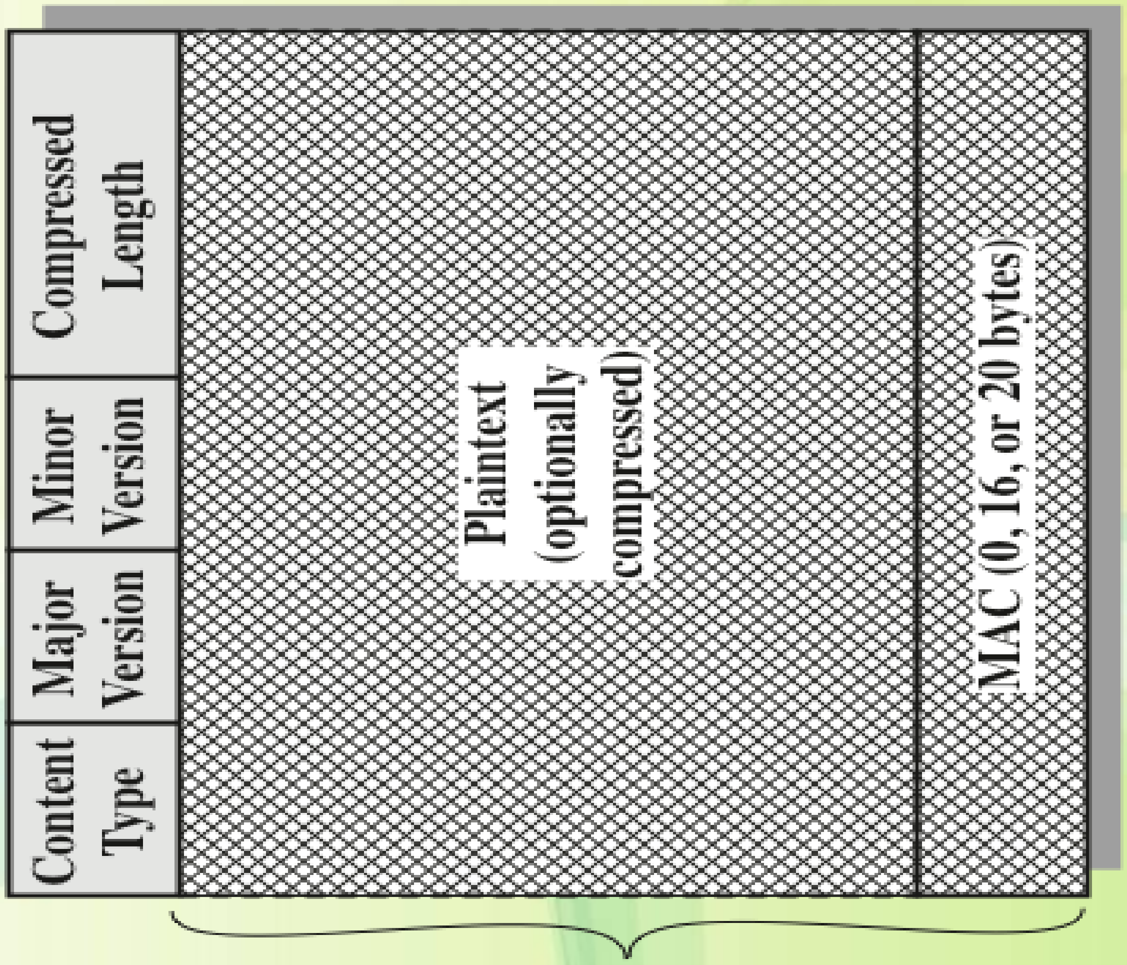


SSL Record Protocol operation

1. The Record protocol takes an application message to be transmitted
2. fragments the data into manageable blocks
3. Compress the data
4. Applies a MAC / add the MAC
5. Encrypts
6. Adds a header, and transmits the resulting unit with TCP segment.
7. Received data are decrypted, verified decompressed, reassembled before being delivered to higher level users



SSL Record format



Encrypted

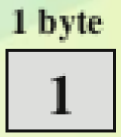
SSL Record format

- **SSL record protocol processing is to prepare a header consisting of the following fields.**

1. Content Type- 8 bits- The higher-layer protocol used to process the enclosed fragment.
2. Major Version- 8 bits- Indicates major version of SSL in use. For SSLv3, the value is 3.
3. Minor Version- 8 bits- Indicates minor version in use. For SSLv3, the value is 0.
4. Compressed length -16 bits-
5. The length in bytes of the plaintext fragment (or compressed fragment if compression is used). The maximum value is $2^{14} + 2048$.

SSL Change Cipher Spec Protocol

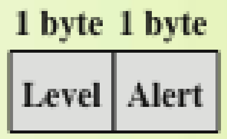
- One of 3 SSL specific protocols which use the SSL Record protocol
- It is the simplest
- It consist of a single message with single byte with value 1
- causes pending state to become into current state
- which updates the cipher suite(set of algorithms) used for connection.



(a) Change Cipher Spec Protocol

SSL Alert Protocol

- The alert protocol is used to conveys SSL-related alerts to peer connections
- The first byte takes the value (1) warning or (2) fatal to convey the message of severity. If the level is fatal, SSL immediately terminates the connection.
    - warning or fatal
- The second byte contains a code that indicates the specific alert.
- specific alert
    - Unexpected_message, bad_record_MAC, decompression_failure, handshake_failure, illegal_parameter
    - Close_notify, no_certificate, bad_certificate, unsupported _certificate, certificate_revoked, certificate_expired, certificate_unknown
- Alert messages are compressed & encrypted like all SSL data



(b) Alert Protocol

HTTPS

- HTTPS:
  - It is the combination of HTTP and SSL to Implement secure communication between web server and web client (browser)
  - Its use depends on the web server supporting HTTPS communications.
  - Ex: some search engine do not support HTTP, Google provides HTTPS as an option.
  - Principle difference seen by user of a web browser is that URL(Uniform resource locator) addresses begins with https://
  - A normal HTTP uses port 80 and HTTPS uses port 443
- When HTTPS is used following communication are encrypted:
  - URL of requested document
  - Contents of document
  - Contents of browser forms
  - Cookies(it allows server to store its own information)
  - Contents of HTTP header

- HTTPS is documented in RFC 2818, HTTP over TLS. There is no fundamental change in using HTTP over SSL or TLS

HTTPS Connection Initiation

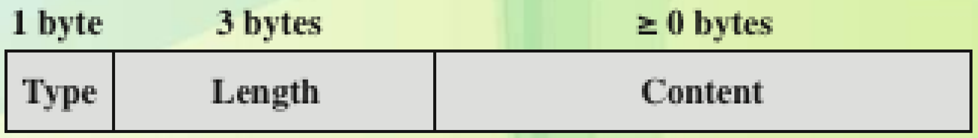
- For HTTPS, the agent acting as the HTTP client (Web browser) and also acts as TLS (Transport layer security) client
  - Client initiates a connection to the server and sends TLS client_Hello message to begin TLS Handshake.
  - Once the handshake finishes client may begin HTTP request.
  - Three levels of connection in HTTPS
    - HTTP level
    - TLS/SSL level
    - TCP level

HTTPS Connection Closure

- A HTTP client or server can indicates the closing of a connection by including: **connection: close** in HTTP record
  - Closure of an HTTPs connection require close of TLS connection
    - Use the TLS alert protocol to send close_notify alert;
    - May close the connection without waiting for the peer to send its closure alert
    - HTTP client must be able to cope with a situation in which underly TCP connection is terminated without a prior close_notify alert and without a connection close indicator such a situation could be due to a programming error.

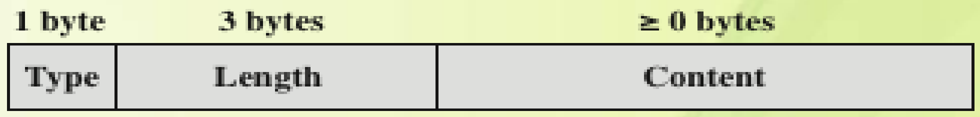
SSL Handshake Protocol

- Most complex part of SSL
- allows server & client to:
  - authenticate each other
  - To negotiate encryption & MAC algorithms
  - to negotiate cryptographic keys to be used
- Consists of a series of messages exchanged by server and client
- These exchanges can be viewed as four phases.
  - Establish Security Capabilities
  - Server Authentication and Key Exchange
  - Client Authentication and Key Exchange
  - Finish

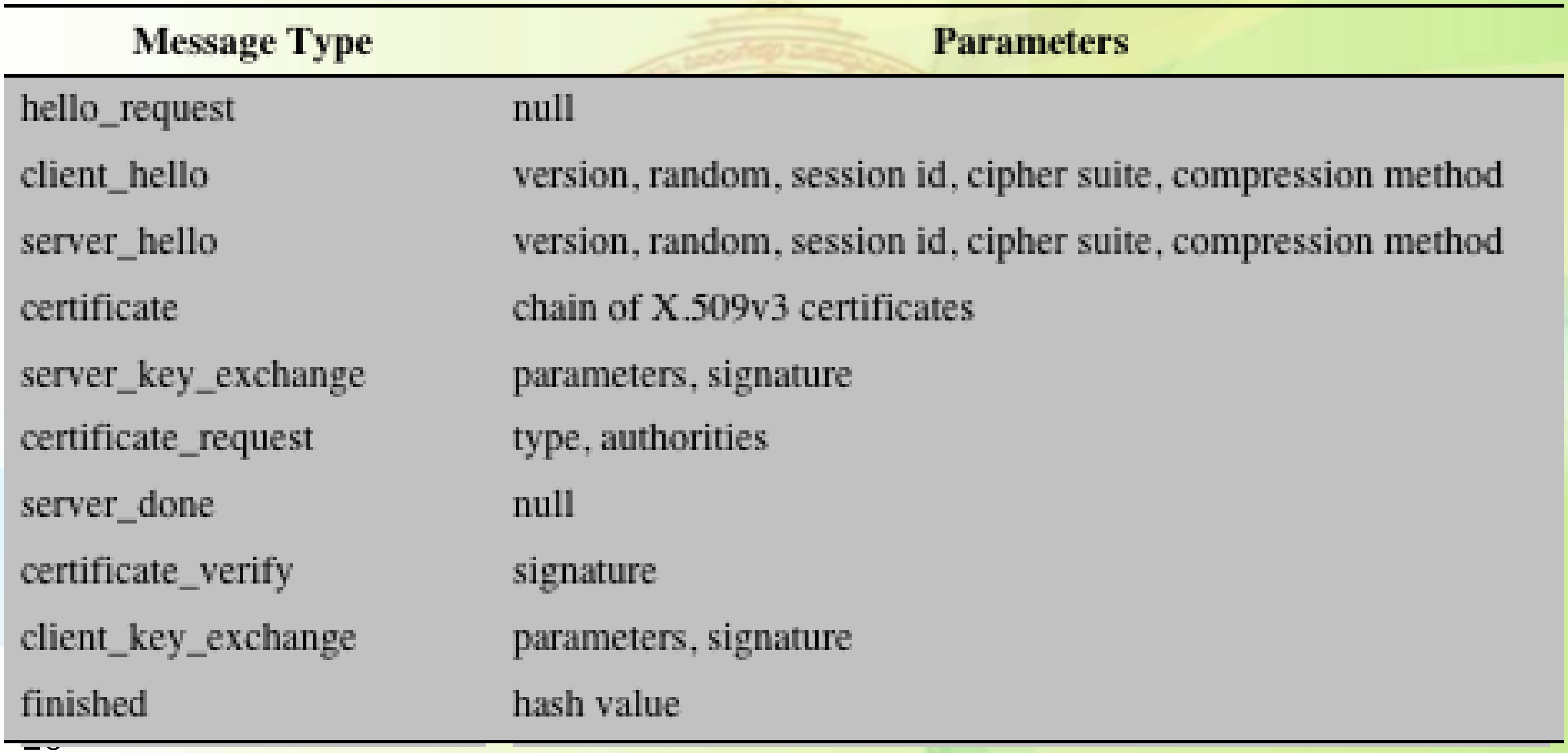


(c) Handshake Protocol

SSL Handshake Protocol

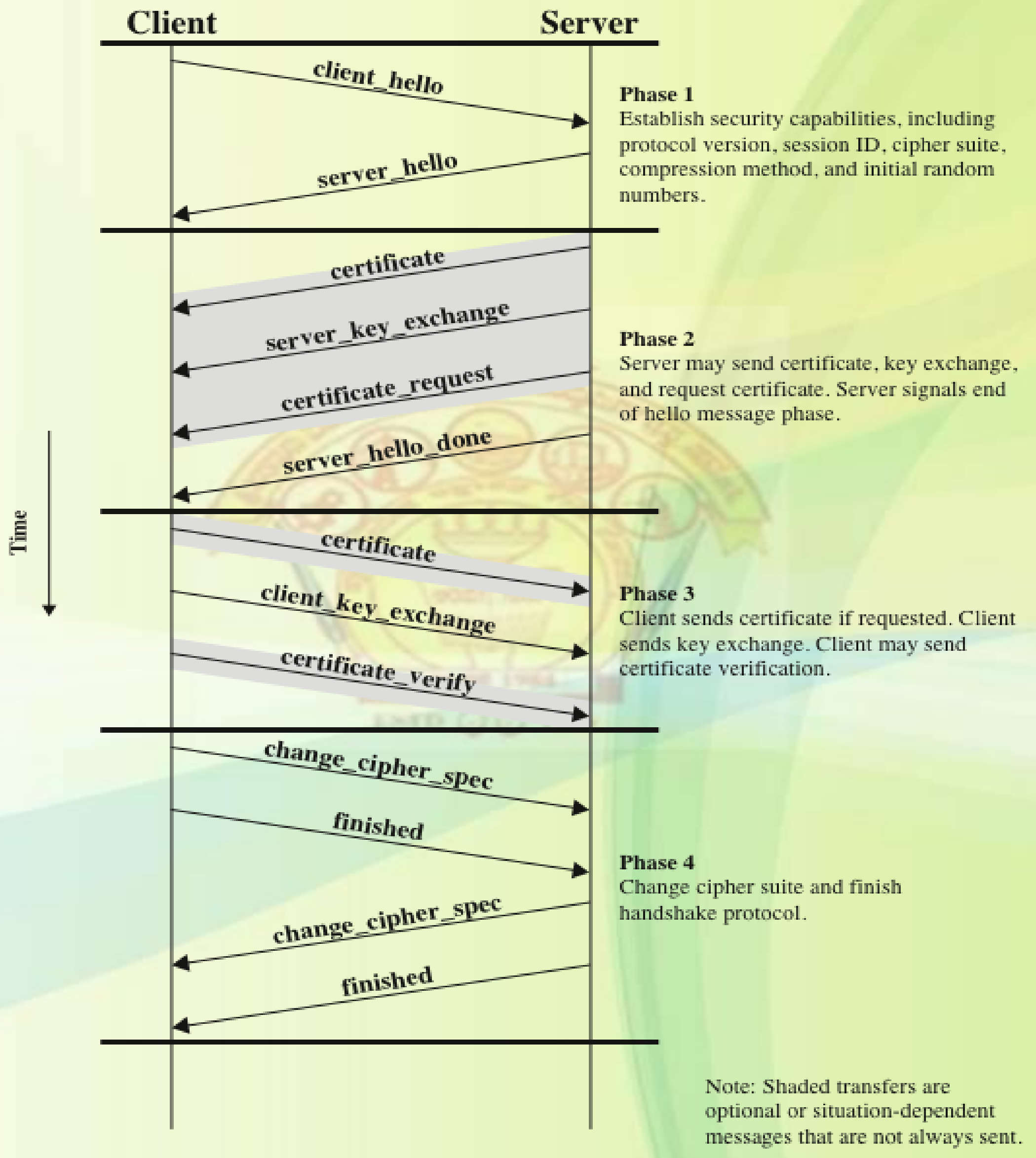


(c) Handshake Protocol



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

SSL Handshake Protocol

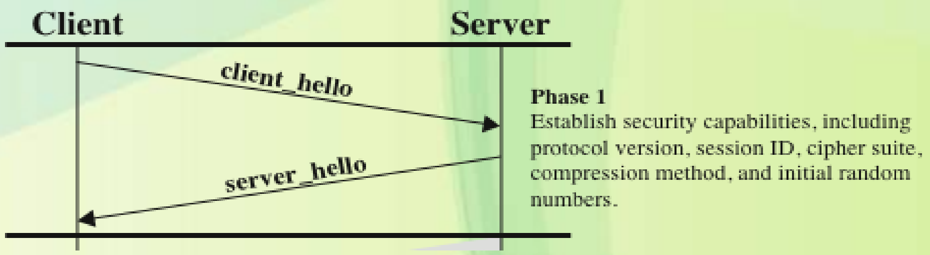


Client
Server
client_hello
server_hello
certificate
server_key_exchange
certificate_request
server_hello_done
certificate
client_key_exchange
certificate_verify
change_cipher_spec
finished
change_cipher_spec
finished
Time
Phase 1
Establish security capabilities, including protocol version, session ID,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

Figure 17.6 Handshake Protocol Action

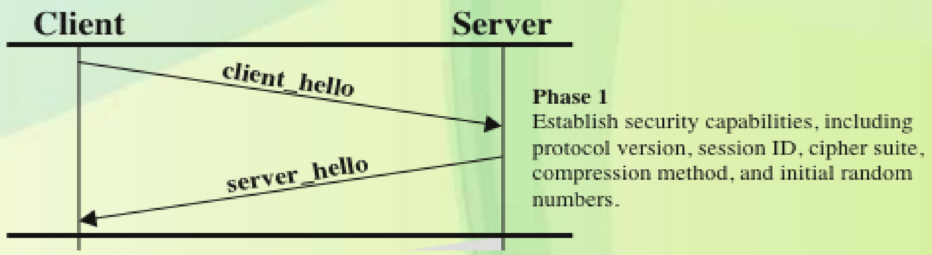
SSL Handshake Protocol
Phase 1: establish security capabilities

– The client initiates a logical connection “client_hello”
  - Parameters: version, random, session ID, cipher suite, compression method
  - Details of cipher suite: key exchange method
- “server hello”


Client
Server
client_hello
server_hello
Phase 1
Establish security capabilities, including protocol version, session ID, cipher suite, compression method, and initial random numbers.

SSL Handshake Protocol
Phase 1: establish security capabilities

- The client initiates a logical connection “client_hello”
- Version: The highest SSL version understood by client
- Random: A client generate random structure with 32 bit time stamp and 28 bytes secure random number generated by generator
- Session ID: A variable length session identifier.
- Cipher suite: Combination of cryptographic algorithms supported by client
- Compression method: The compression method supported by client
- After sending client_hello message client wait for the “server hello”


Client
Server
client_hello
server_hello
Phase 1
Establish security capabilities, including protocol version, session ID, cipher suite, compression method, and initial random numbers.

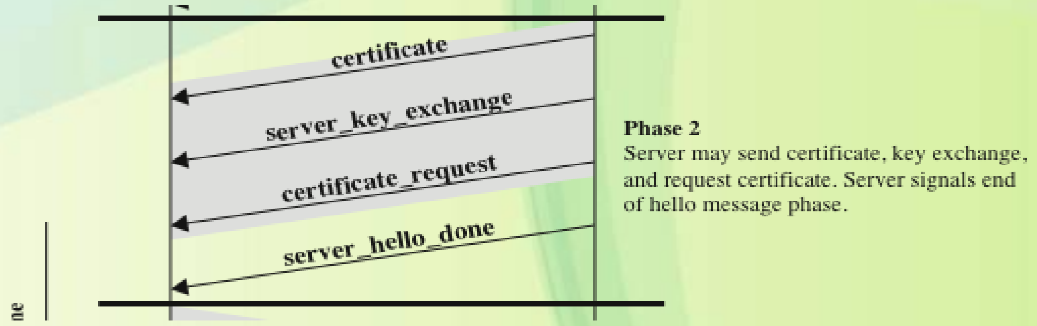
SSL Handshake Protocol
Phase 1: establish security capabilities

- Details of cipher suite: key exchanged method.
- The following key exchange methods are supported

- RSA(Ron Shamir Adleman)- the secret key is encrypted with reciever’s RSA public key
- Fixed Diffie-Hellmann- The server’s certificate contains the Diffie- Hellman public parameters signed by public certificate authority(CA).
- Ephemeral Diffie-Hellman: this technique is used to create one time/temporary key
- Anonymous Diffie-Hellman: the base Diffie-Hellman algorithm is used with no authentication.
- Fortezza: Information security system that uses the Fortezza cryptography card.

Phase 2: Server authentication and key exchange

1. Server sends its certificate: one or chain of X.509 certificates;
2. Server sends a server_key_exchange message;
   - E.g. 1 anonymous DH
   - E.g. 2 Ephemeral Diffie-Hellman
   - E.g 3 RSA key exchange
   - E.g. 4 Fortezza : Signature in this message
3. Server sends a certificate_request message
   - Certificate type and a list of CAs
4. Server sends a server_hello_done message

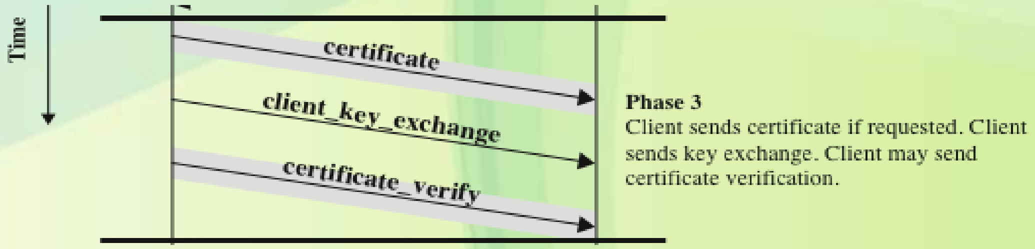

certificate
server_key_exchange
certificate_request
server_hello_done
Phase 2
Server may send certificate, key exchange, and request certificate. Server signals end of hello message phase.

Phase 3: Client authentication and key exchange

Client first verify server's certificate and parameters Received.

If all good →

1. If server requests a certificate, client sends a certificate message
2. Client sends a client_key_exchange message
   - E.g. 1 RSA: 48-byte pre-master secret, encrypted with server's public key or RSA key
   - E.g. 2 anonymous DH
   - E.g. 3 Fixed DH
3. Client sends a certificate_verify message

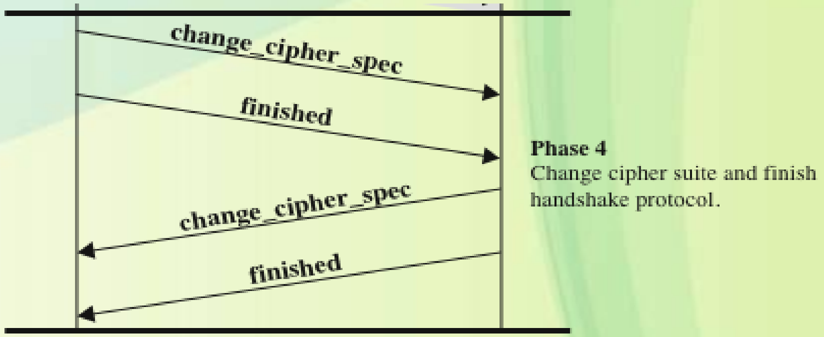

Time
certificate
client_key_exchange
certificate_verify
Phase 3
Client sends certificate if requested. Client sends key exchange. Client may send certificate verification.

Phase 4: Finish

1. Client sends a change_cipher_spec message
2. Client sends a finished message
   - Verify the key exchange and authentication process were successful

- Server sends a change_cipher_spec message
- Server send a finished message

--- handshake is complete ---



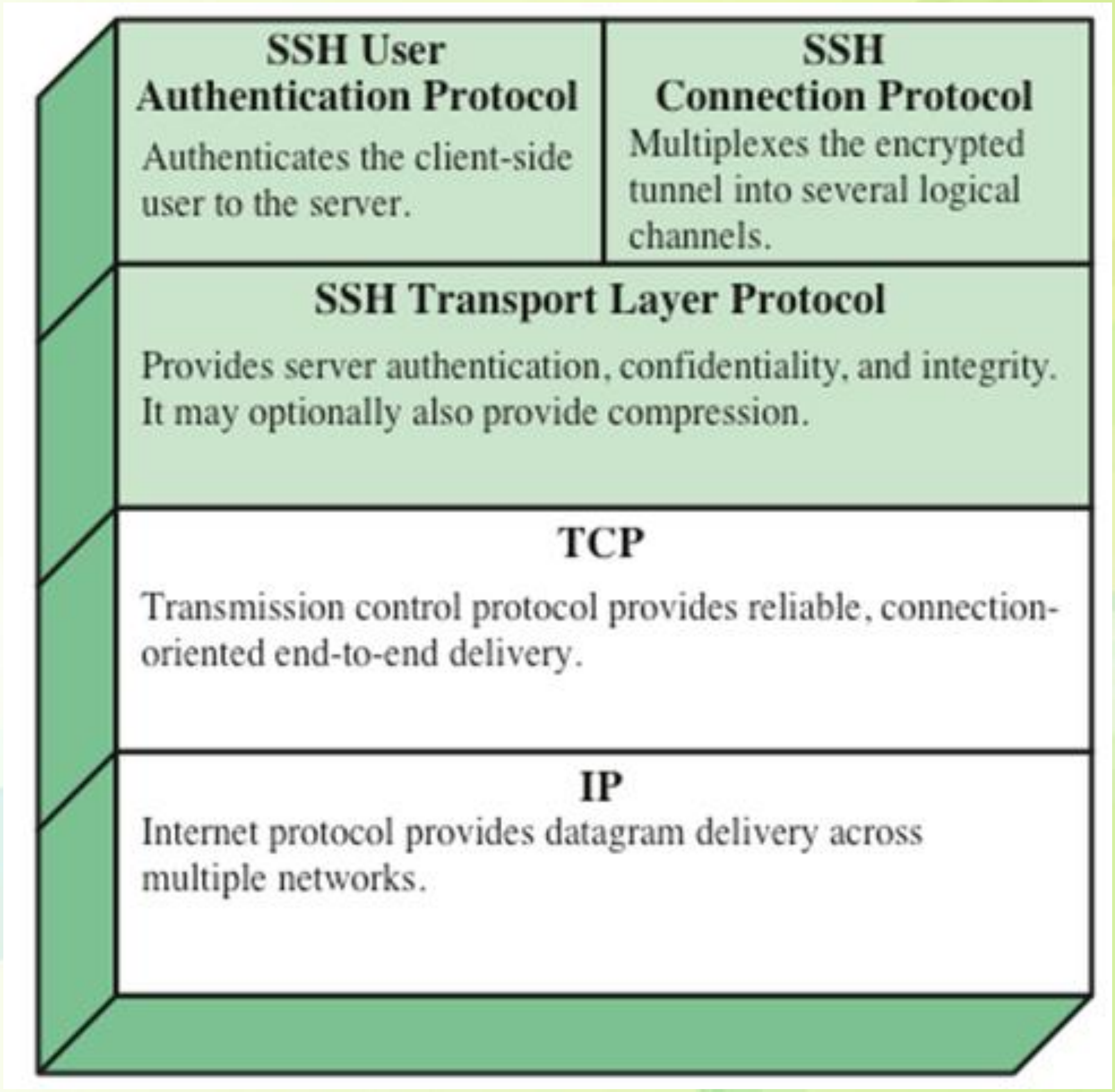
Secure Shell (SSH)

- SSH is a protocol for secure network communication
- Simple and inexpensive to implement.
  - SSH1 is designed to replace Telnet
  - security issues with Telnet
    - Sends all data in clear text.
    - Host between sender and receiver can see what the traffic is.
    - No security
  - SSH provides secure remote access between client/server and can be used for file transfer and e-mail.
  - Transmission can be compressed.

History of SSH

- Created by Tatu Ylönen in July 1995, a student of Helsinki University of Technology
  - Initial version is SSH1, provides secure logon facility.
  - A new version SSH 2 fixes a number of security flaws in SSH1 (RFC4250 – 4256)
- SSH client and server applications are widely available in most operating systems.
- SSH is organized as three protocols, run on top of TCP

1. Transport layer Protocol
2. User authentication Protocol
3. Connection Protocol


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.

Functions of SSH protocol stack

- Transport layer protocol
  - Provides server authentication, data confidentiality and integrity
- User authentication protocol
  - Authenticates the user to the server
- Connection protocol
  - Multiplex multiple logic communication channels over a single SSH connection

SSH Transport layer protocol

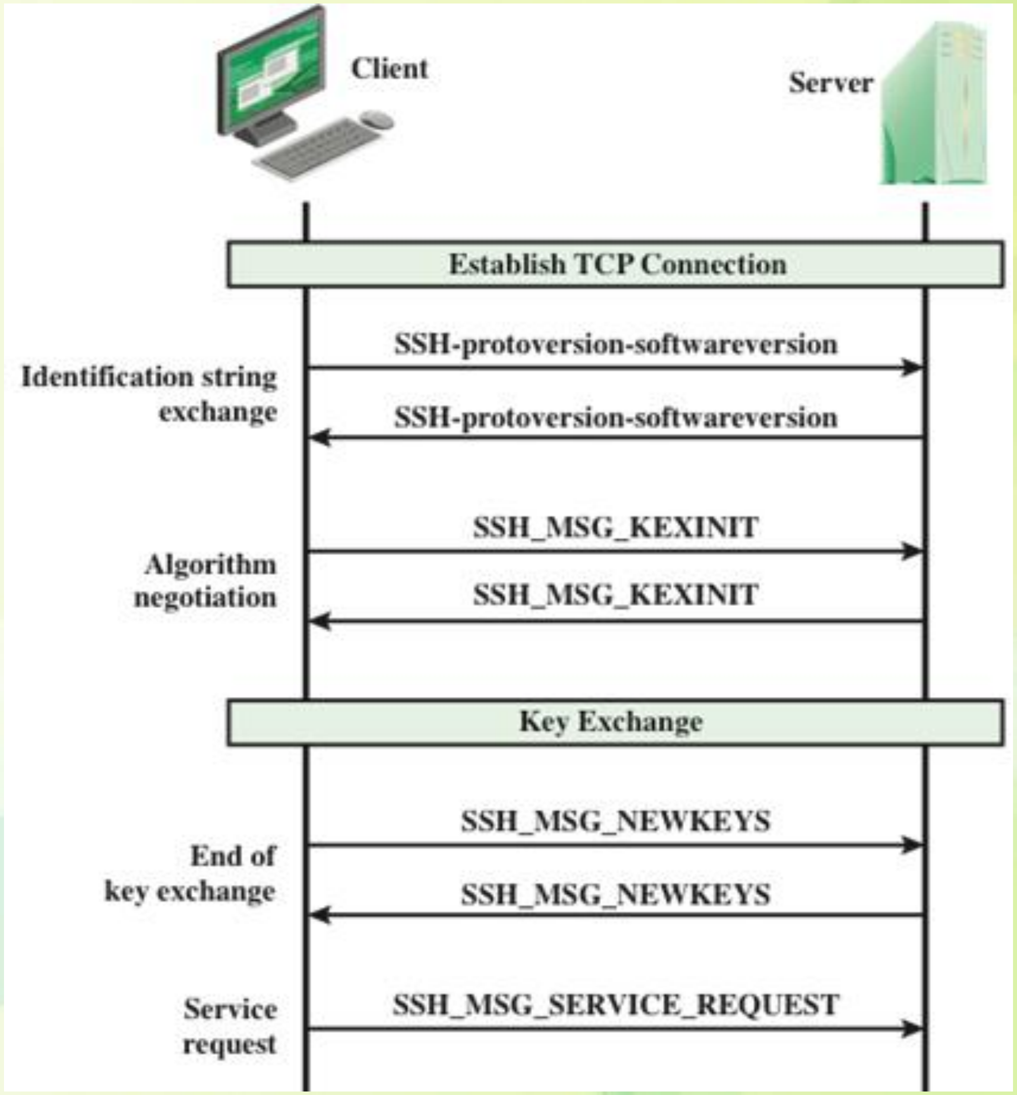
- Server authentication is based on the server's public/private key pair
  - Host Keys: a server may have one host or many hosts could share one key
  - Client must have the server's public key in advance!
- Two alternative trust models defined in RFC4251
    - The client has a local DB associates each server with public key
    - The host name to key association is certified by CA. The client only knows CA root key can verify validity of all host keys certified by CA.

SSH Packate exchange

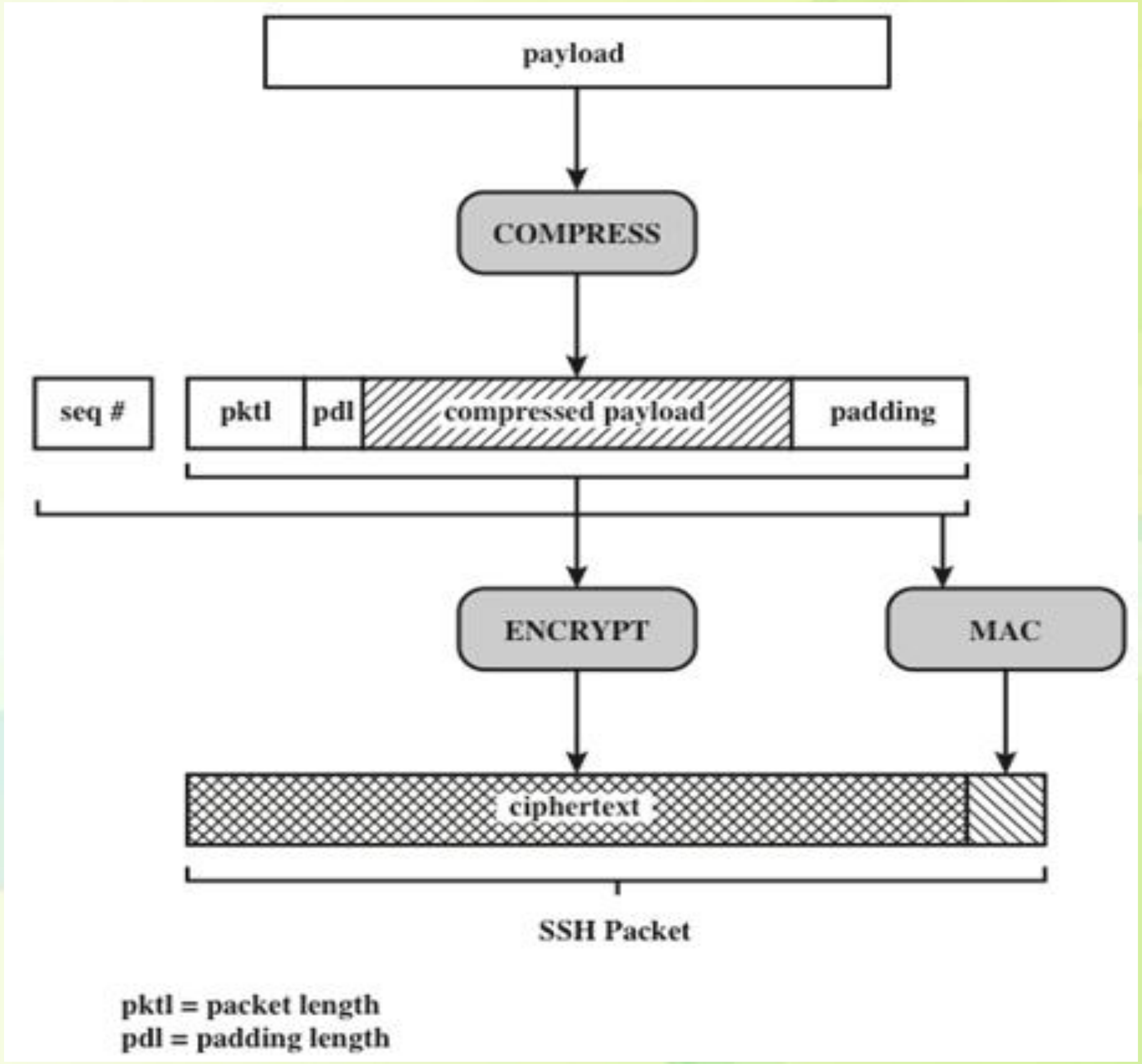
- Package exchange of SSH Transport Layer Protocol
  - First, client establish TCP connection to the server
  - Then starts SSH key exchange steps (next slide)
  - The client and server exchange data (packets)
    - Packet format (after next slide)

      pktl, pdl, payload (may be compressed), random padding, MAC,

SSH Transport Layer Protocol Packet exchange steps



SSH Transport Layer Protocol Packet Formation



SSH Transport Layer Protocol Packet Formation

- Packet Length : Length of the packet in bytes.
- Padding Length: Length of the random padding Field.
- Payload: Useful contents of the packet. Added before algorithm negotiation.
- Random Padding: Once an encryption algorithm has been negotiated this field is added. It contains random bytes of padding.
- Message Authentication Code (MAC): If a message authentication has added to encryption this field contains the MAC value.
- Sequence Number : It is 32 bit. Sequence Number is initiated to zero for the first Packet and incremented for every packet.

SSH Transport Layer Protocol key exchange steps

- Establish TCP Connection.
- Identification string Exchange
- Algorithm Negotiation
- Key Exchange
- End of Key Exchange
- Service request

SSH Transport Layer Protocol key exchange steps

1. Establish TCP Connection.
2. Identification string Exchange

- SSH-protoversion-softwareversion SP Comments CR LF
- SP-space character
- CR-carriage return
- LF-line feed
- Example: SSH-2.0-billsSSH_3.6.3q3<CR><LF>
- The client responds with its own identification string.
- The server responds with its own identification string.

SSH Transport Layer Protocol key exchange steps

3. Algorithm Negotiation

- Each side sends an SSH_MSG_KEXINIT containing list of supported algorithms.

4. Key Exchanges

5. End of key Exchanges: : SSH_MSG_NEWKEYS packets

6.Service Request : SSH_MSG_SERVICE_REQUEST

SSH Transport Layer Crypto 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

SSH User Authentication Protocol

- User Authentication Protocol provides client Authentication to the server
- Message Types and Formats used: Three types of messages are used
- byte SSH_MSG_USERAUTH_REQUEST (50)

string	user name
string	service name
string	method name

- byte SSH_MSG_USERAUTH_FAILURE (51)

name-list	authentication that can continue
boolean	partial success

- byte SSH_MSG_USERAUTH_SUCCESS (52)

SSH User Authentication Protocol

- Message exchange

1. Client sends request
2. Server checks if user name is valid → valid or NOT
3. Server returns result of step 2 and a list of authentication methods
4. Client selects one of authentication method in step 3 and reply its choice

A sequence of exchange to perform authentication

5. Based on authentication result, go to step 3 Or

6 when all required authentication methods succeeds, server sends a success message

Authentication methods in SSH User Authentication Protocol

- Public key
  - Client sends message to server. The message contains signature (message encrypted by client’s private key) and client’s public key
  - Server verify if the key is acceptable and if the signature is valid
- Password
  - Client sends a password encrypted by Transport layer protocol
- Hostbased
  - Client sends a signature created with private key of client host
  - Server verifies the identity of client host, and then believes the client host already authenticate that client

SSH Connection Protocol

- SSH connection protocol runs on the top of SSH Transport layer protocol
  - Secure authentication connection is called tunnel
  - Each side may open a channel, and each side associates a unique channel number.

- SSH Connection Protocol steps (next slide)
  1. Open a channel
  2. Data transfer
  3. Close a channel

SSH Connection Protocol

- The life of a channel progress through three stages
- Open a channel, data transfer and closing a channel
- **Open a channel**
- When either side wishes to open a channel, it allocates a local number of channel and sends a message of the form
- byte SSH_MSG_CHANNEL_OPEN

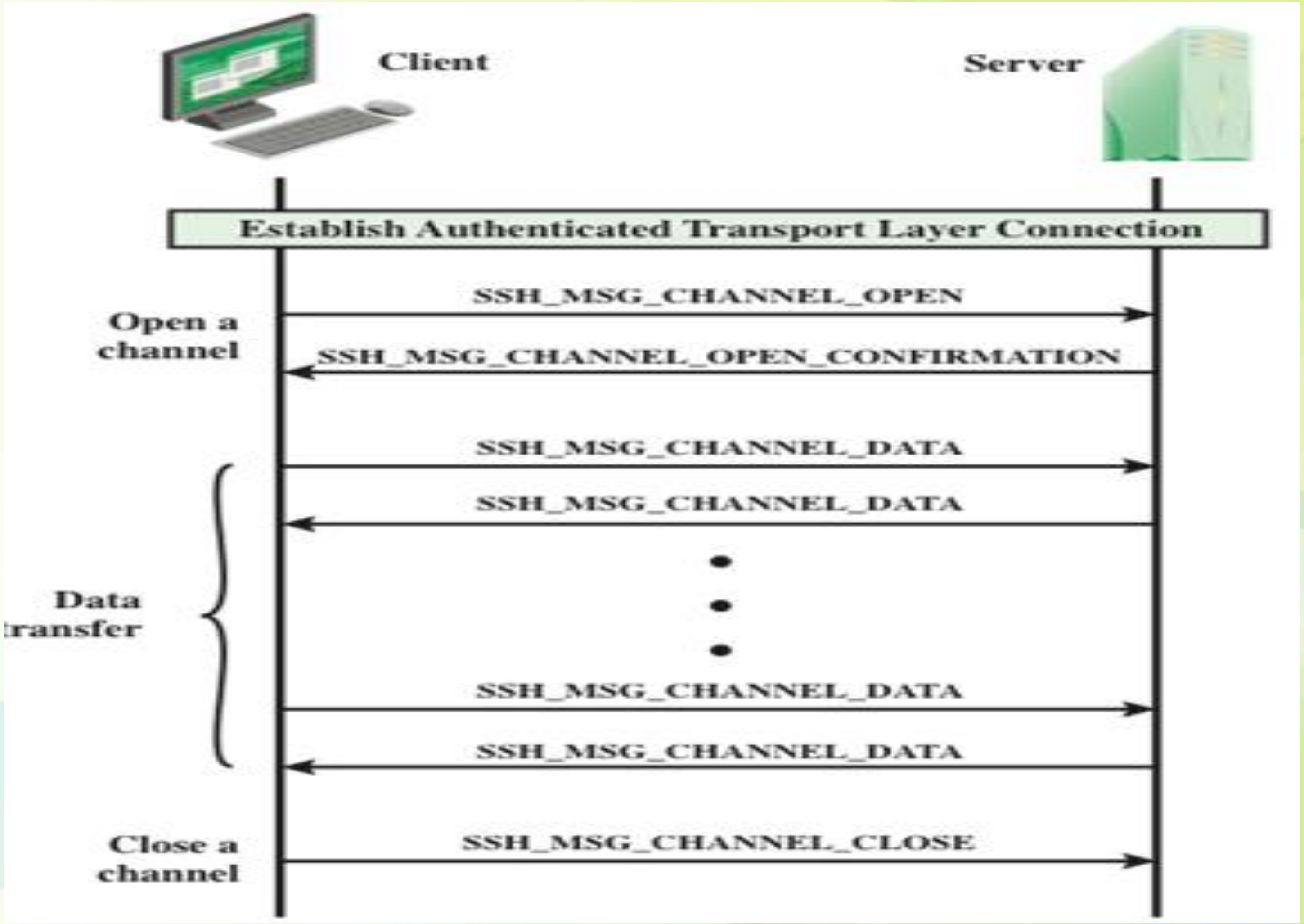
string	channel type	(application for this channel)
unit32	sender channel	(local channel number)
unit32	initial window	(bytes of channel data)
unit32	maximum packet size	

… channel type specific data follows

SSH Connection Protocol

- If the remote side is able to open channel, it returns a SSH_MSG_CHANNEL_CONFIRMATION message, which includes the sender channel number, and window and packet size values for incoming traffic. - Otherwise, the remote channel returns a SSH_MSG_CHANNEL_FAILURE message with a reason code indicating the reason of failure.
- Once 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.

SSH connection Protocol Message exchange



Secure Shell Connection Protocol

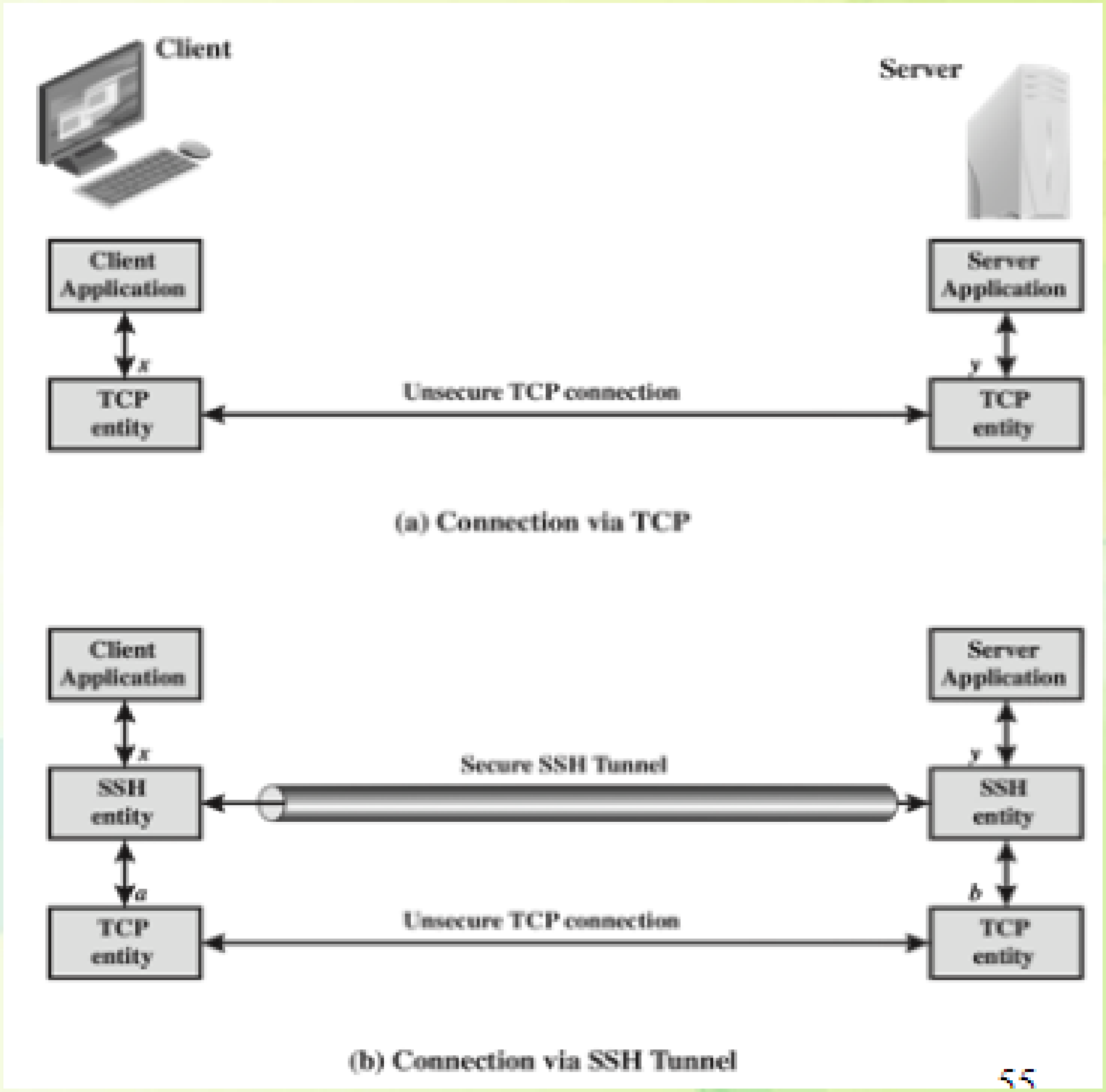
- Channel Types –
- Four channel types are recognized in the SSH connection protocol specification.
- session : The remote execution of a program. The program may be a shell, an application such as file transfer or e – mail, a system command, or some built – in subsystem. Once a session channel is opened, subsequent requests are used to start the remote program.
- x11 : This refers to the X window system, a computer software system and network protocol that provides a GUI for networked computers. X allows applications to run on a network server but to be displayed on a desktop machine..
- forwarded – tcpip : This is remote port forwarding.
- direct – tcpip : this is local port forwarding.

Secure Shell Connection Protocol

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

Connection via SSH Tunnel



Secure Shell Connection Protocol

Local Forwarding

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

Secure Shell Connection Protocol

Remote Forwarding

- With 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.

Secure Command Shell

- Allow you to edit files.
- View the contents of directories.
- Custom based applications.
- Create user accounts.
- Change permissions.
- Anything can be done from command prompt can be done remotely and securely.

Secure Shell Connection Protocol

Remote Forwarding

- With 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.

Port Forwarding

- A Powerful Tool.
  - provide security to TCP/IP applications including e-mail, sales and customer contact databases, and in-house applications.
  - allows data from normally unsecured TCP/IP applications to be secured.

Two Cryptographic Items in Handshake process (formula is Not required)

- The creation of a **shared master secret key** by key exchange

- Generation of **cryptographic parameters** from master secret;

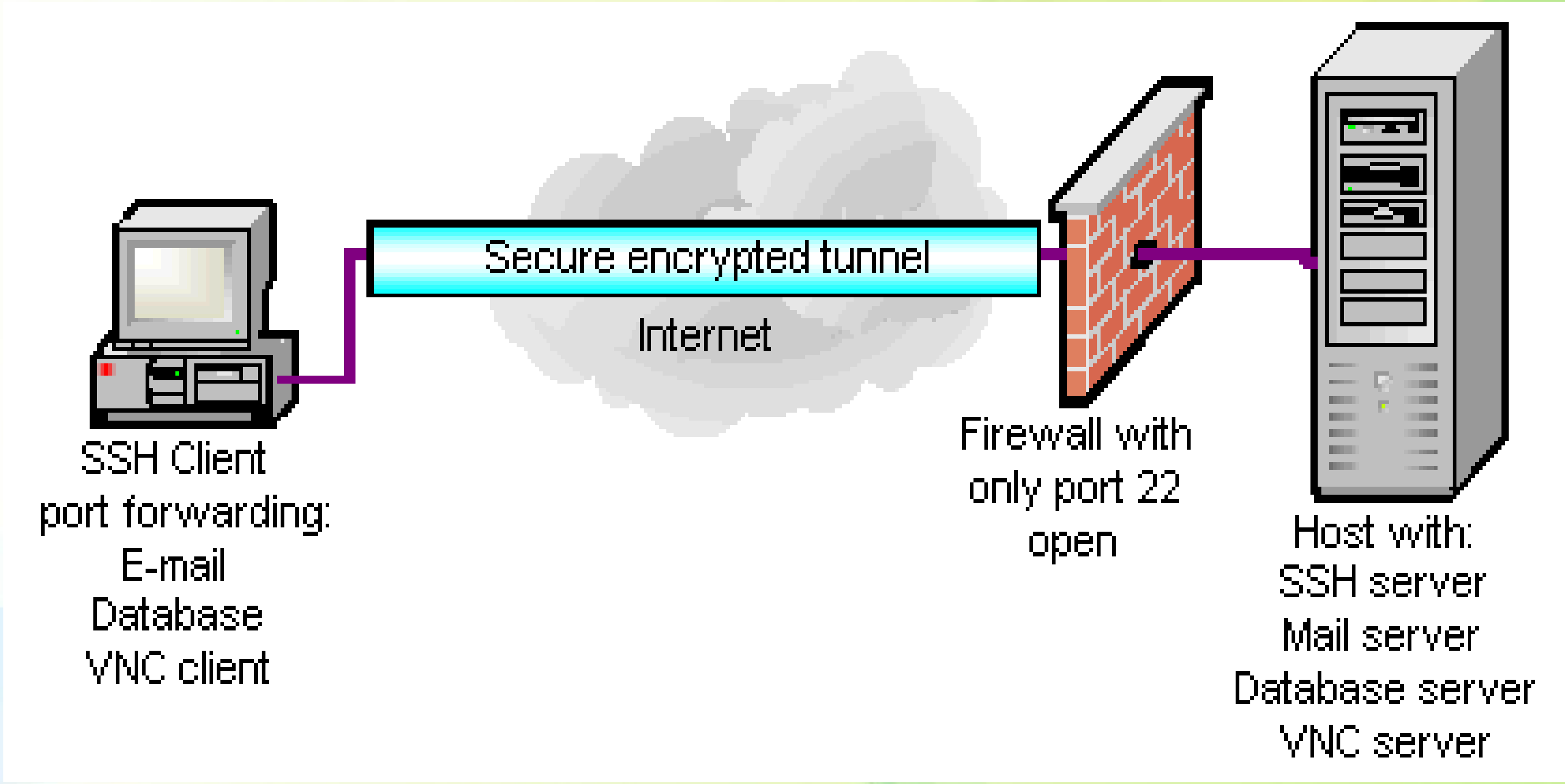
The creation of a shared master secret key by key exchange (formula is Not required)

- Shared master secret is one-time 48-byte for this session by secret key exchange
  1. Pre_master_secret is exchanged
  2. Master-secret is calculated by both parties;

  – E.g. 1 RSA

  – E.g. 2 DH

Port Forwarding



THANK YOU