2 : Transport Layer Security

IT6406 – Network Security

Level III - Semester 6

Overview

- In this topic we discuss about the essential topics related to transport level security through briefly introducing secure protocols such as Transport Layer Security (TLS), HTTPS, and Secure Shell (SSH).

Intended Learning Outcomes

- At the end of this lesson, you will be able to;
  - Summarize Web security threats and Web traffic security approaches
  - Describe Transport Layer Security (TLS) protocol suite
  - Describe the differences between Secure Sockets Layer and Transport Layer Security.
  - Discuss an overview of HTTPS (HTTP over SSL)
  - Discuss an overview of Secure Shell (SSH)

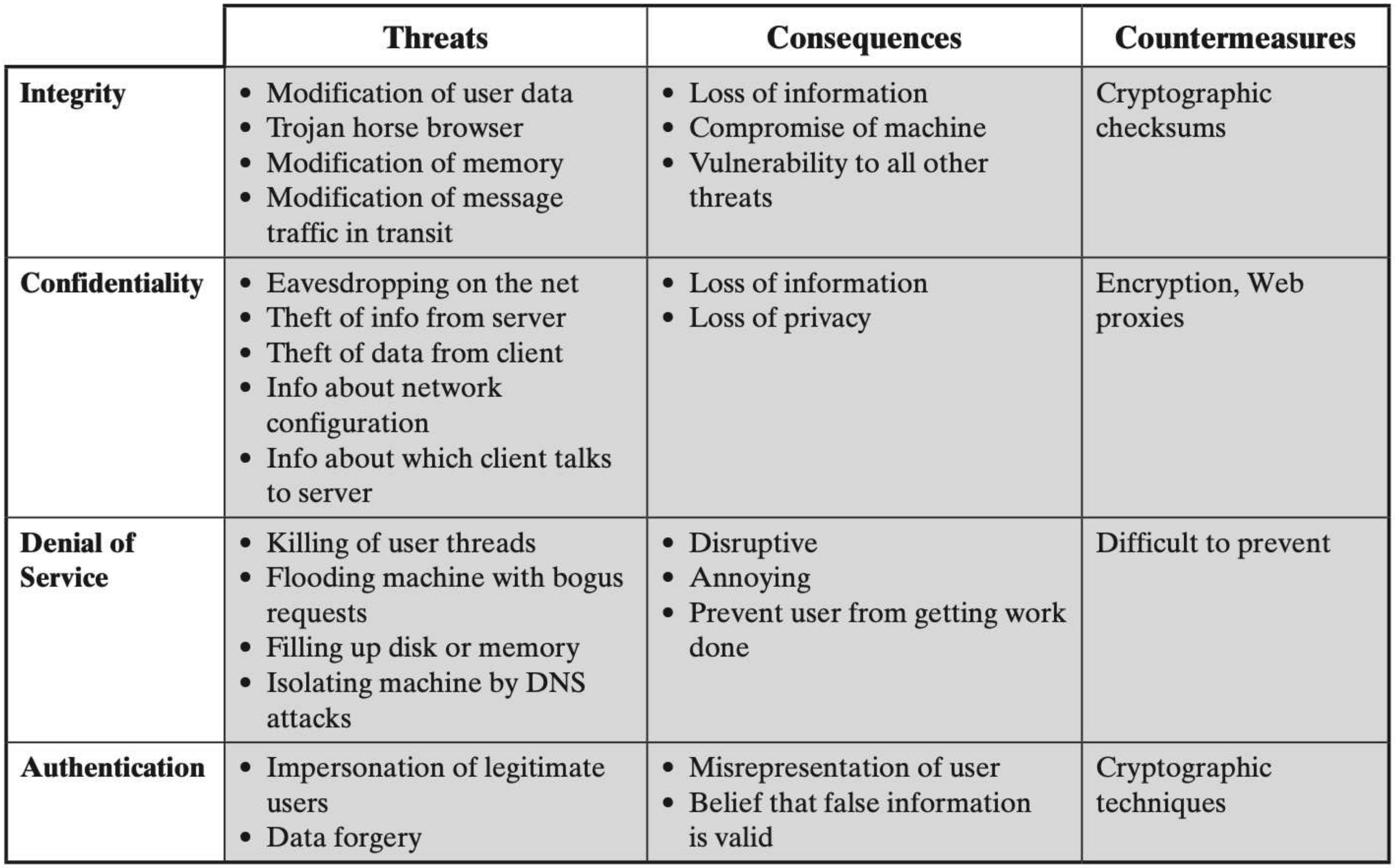
List of sub topics

2.1 Web Security Considerations

- The need of web security.
  - The underlying software for delivering web content is extraordinarily complex. This complex software may hide many potential security flaws.
  - A Web server can be exploited as a launching pad into the corporation's or agency's entire computer complex.
  - Casual and untrained (in security matters) users are common clients for Web- based services.

2.1 Web Security Considerations

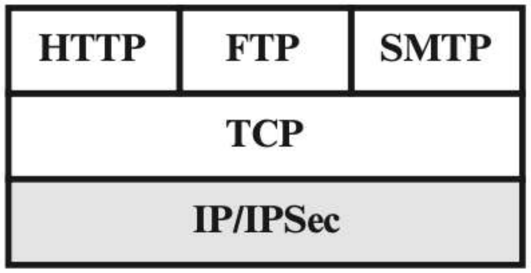
2.1.1 Web security threats



	Threats	Consequences	Countermeasures
Integrity	• Modification of user data • Trojan horse browser • Modification of memory • Modification of message traffic in transit	• Loss of information • Compromise of machine • Vulnerability to all other threats	Cryptographic checksums
Confidentiality	• Eavesdropping on the net • Theft of info from server • Theft of data from client • Info about network configuration • Info about which client talks to server	• Loss of information • Loss of privacy	Encryption, Web proxies
Denial of Service	• Killing of user threads • Flooding machine with bogus requests • Filling up disk or memory • Isolating machine by DNS attacks	• Disruptive • Annoying • Prevent user from getting work done	Difficult to prevent
Authentication	• Impersonation of legitimate users • Data forgery	• Misrepresentation of user • Belief that false information is valid	Cryptographic techniques

2.1 Web Security Considerations

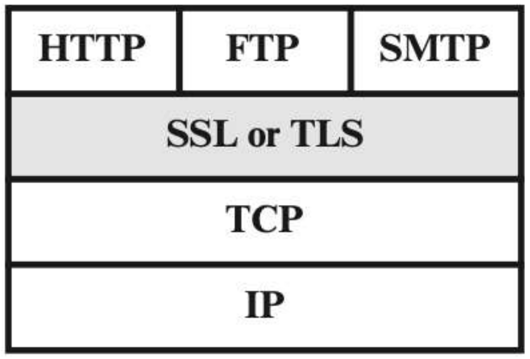
2.1.2 Web traffic security approaches



(a) Network level

2.1 Web Security Considerations

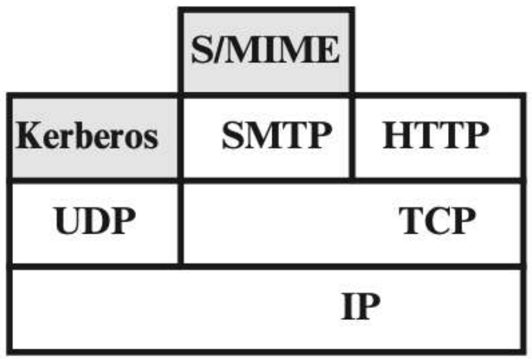
2.1.2 Web traffic security approaches



(b) Transport level

2.1 Web Security Considerations

2.1.2 Web traffic security approaches



(c) Application level

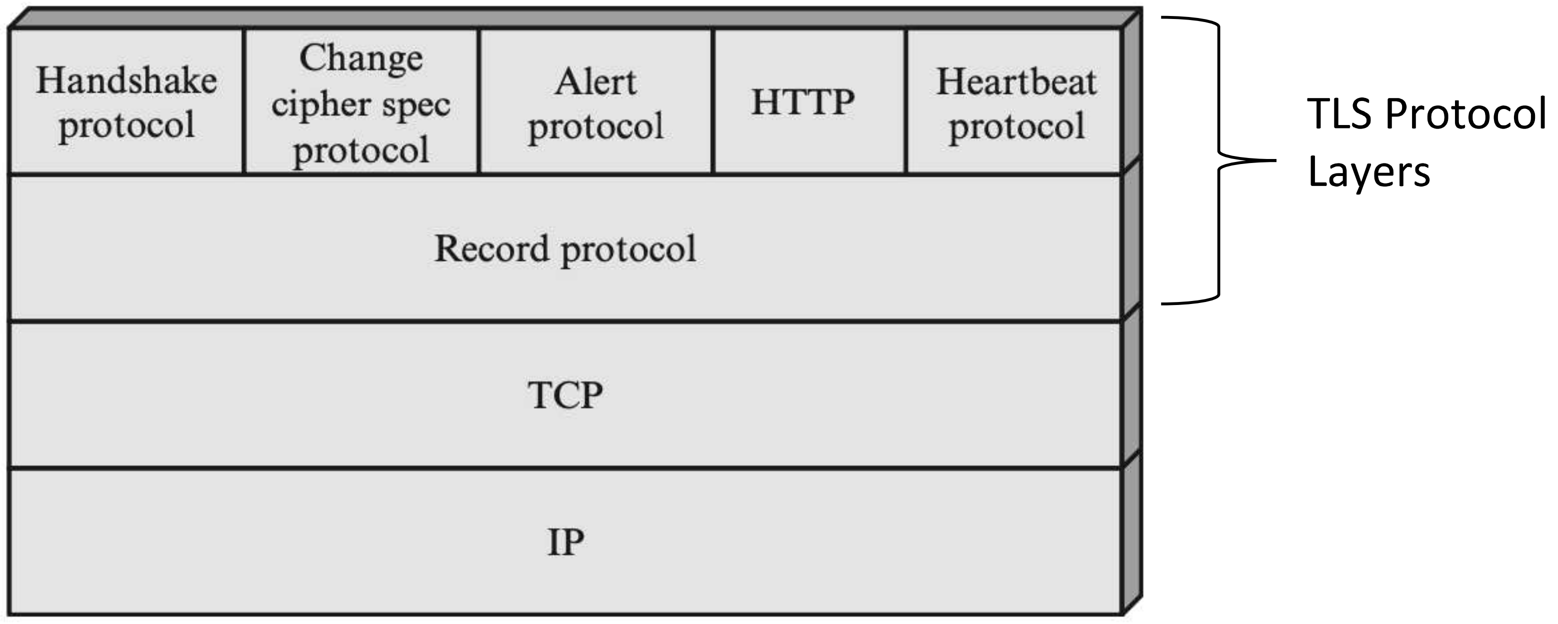
2.2 Transport Layer Security

- Transport Layer Security (TLS)
- the current version is Version 1.2, defined in RFC 5246
- TLS is a general purpose service implemented as a set of protocols that rely on TCP
- TLS could be provided as part of the underlying protocol suite and therefore be transparent to applications
- Most browsers come equipped with TLS, and most Web servers have implemented the protocol.

2.2 Transport Layer Security

2.2.1 TLS Architecture

- TLS is not a single protocol but rather two layers of protocols above TCP



2.2 Transport Layer Security

<u>2.2.1 TLS Architecture</u>

- Two important TLS concepts are the TLS session and the TLS connection
- Between any pair of parties (applications such as HTTP on client and server) there may be multiple secure connections. In theory, there may also be multiple simultaneous sessions between parties, but this feature is not used in practice.
- **Connection:** peer-to- peer relationships, the connections are transient, and every connection is associated with one session.

2.2 Transport Layer Security

2.2.1 TLS Architecture

- **Session:**
  - A TLS 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.

2.2 Transport Layer Security

2.2.1 TLS Architecture

- A session state is defined by the following parameters:
    - Session identifier
    - Peer certificate
    - Compression method
    - Cipher spec
    - Master secret
    - Is resumable
    - Server and client random
    - Server write MAC secret
    - Client write MAC secret
    - Server write key
    - Client write key
    - Initialization vectors
    - Sequence numbers

2.2 Transport Layer Security

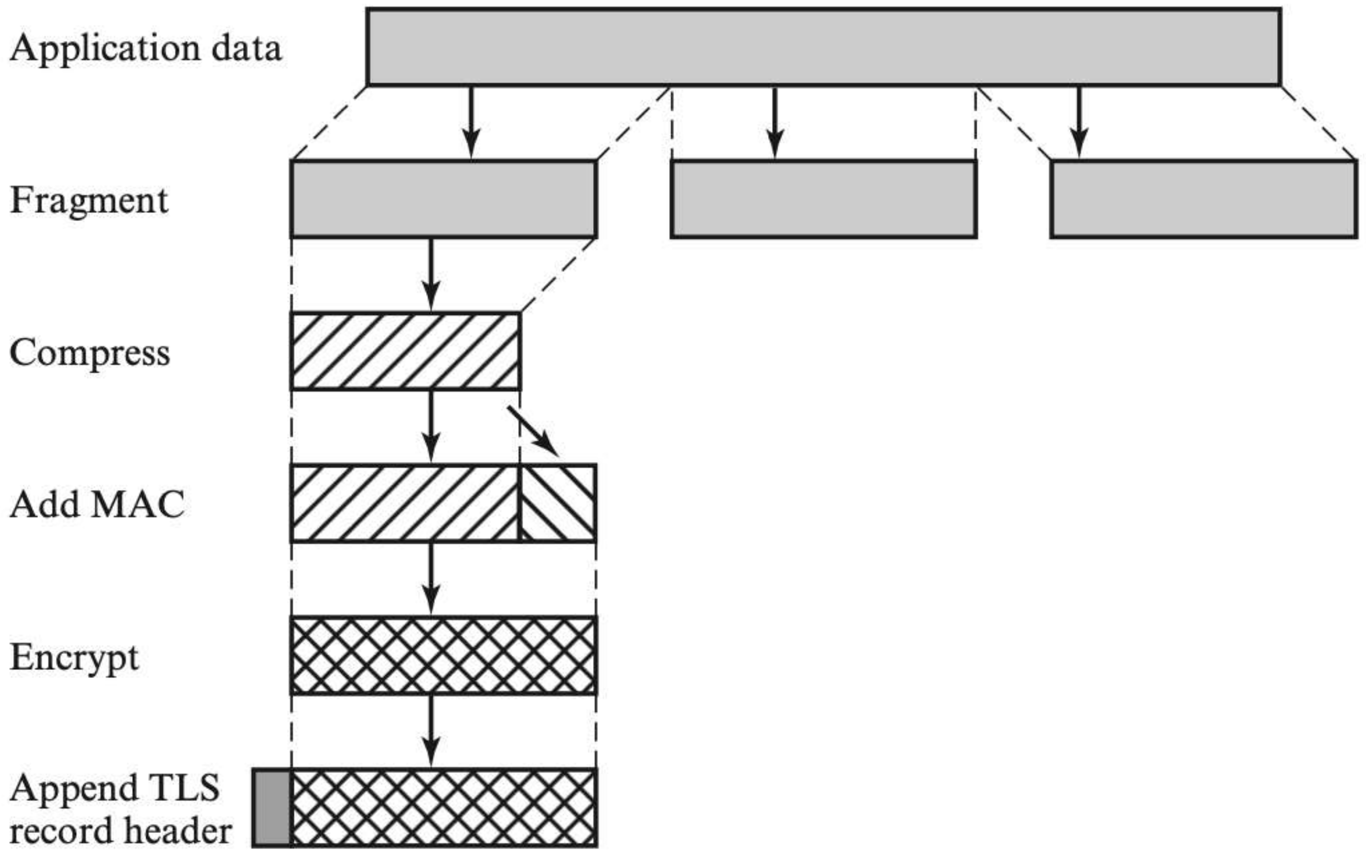
2.2.2 TLS Record Protocol

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

2.2 Transport Layer Security

2.2.2 TLS Record Protocol

- The TLS Record Protocol operation:



2.2 Transport Layer Security

2.2.2 Change Cipher Spec Protocol

- The Change Cipher Spec Protocol is one of the four TLS-specific protocols that use the TLS Record Protocol
- This protocol consists of a single message
- 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

2.2 Transport Layer Security

2.2.2 Alert Protocol

- The Alert Protocol is used to convey TLS-related alerts to the peer entity
- 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
    - The second byte contains a code that indicates the specific alert.
- There are number of alerts that are fatal.

2.2 Transport Layer Security

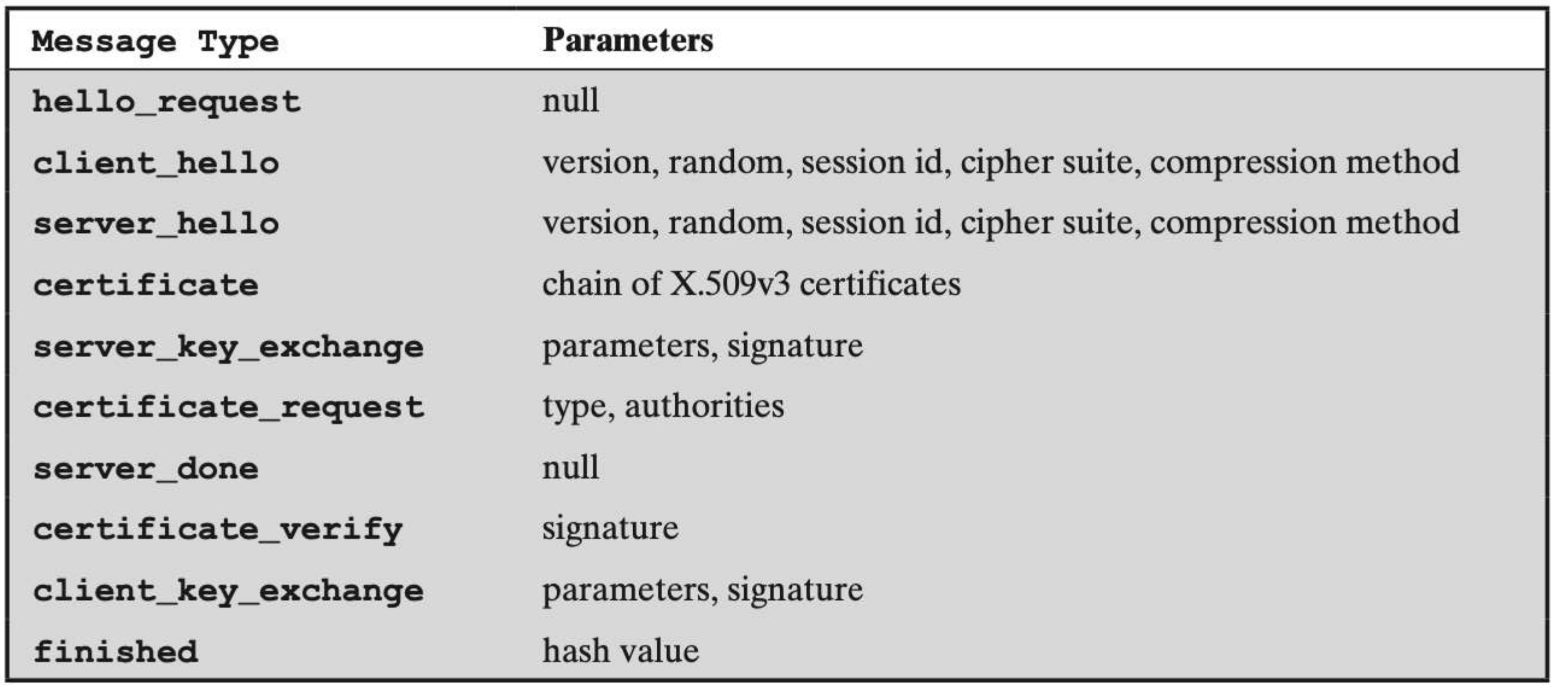
2.2.3 HandShake Protocol

- This protocol allows the server and client to authenticate each other and to negotiate an encryption and MAC algorithm and cryptographic keys to be used to protect data sent in a TLS record.
- The Handshake Protocol is used before any application data is transmitted
- The Handshake Protocol consists of a series of messages exchanged by client and server.

2.2 Transport Layer Security

2.2.3 HandShake Protocol

- Handshake protocol message types



Message Type	Parameters
hello_request	null
client_hello	version, random, session id, cipher suite, compression method
server_hello	version, random, session id, cipher suite, compression method
certificate	chain of X.509v3 certificates
server_key_exchange	parameters, signature
certificate_request	type, authorities
server_done	null
certificate_verify	signature
client_key_exchange	parameters, signature
finished	hash value

2.2 Transport Layer Security

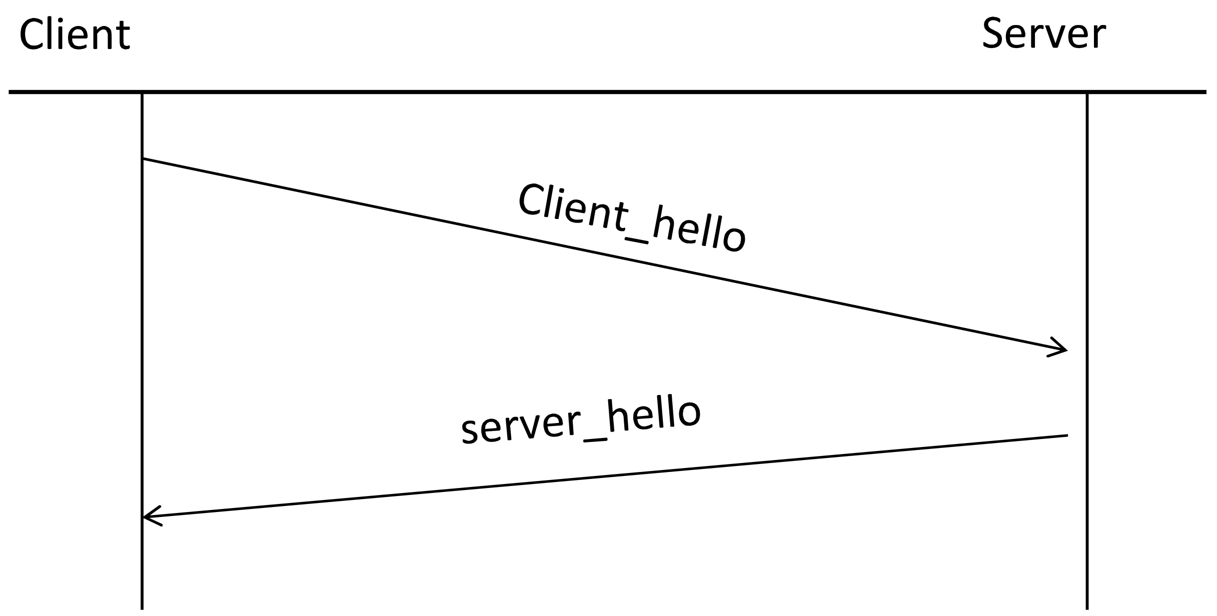
2.2.3 HandShake Protocol

- The exchange can be viewed as having four phases
  - Phase I : Establish security capabilities
  - Phase II : Server authentication and key exchange
  - Phase III : Client authentication and key exchange
  - Phase IV : Finish

2.2 Transport Layer Security

2.2.3 HandShake Protocol - Phase I : Establish security capabilities

- Establish security capabilities, including protocol version, session ID, cipher suite, compression method, and initial random numbers.



2.2 Transport Layer Security

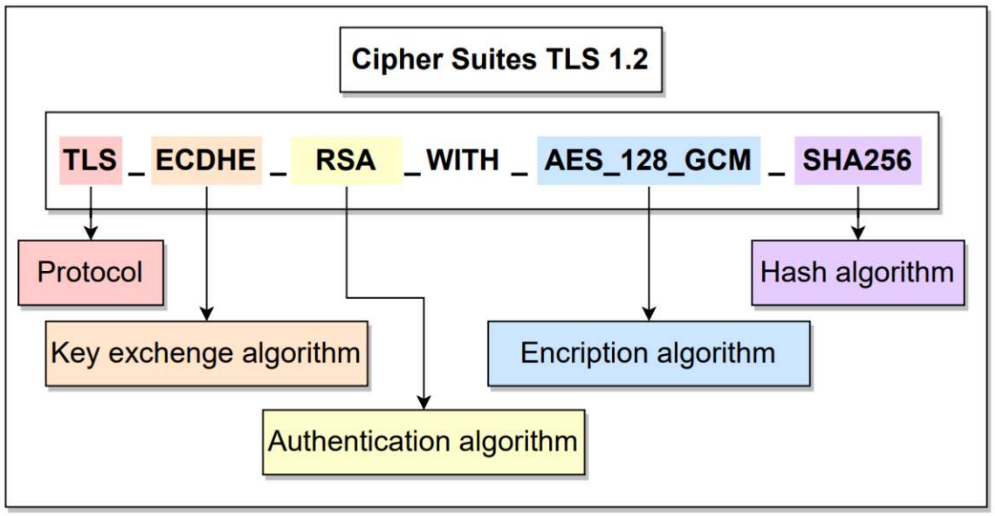
2.2.3 HandShake Protocol - Phase I : Establish security capabilities

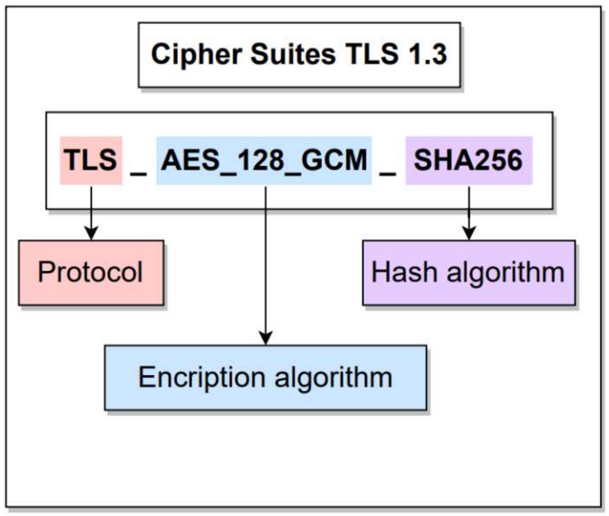
- Parameters of client_hello message
    - Version
    - Random
    - Session ID
    - CipherSuite
    - Compression method

2.2 Transport Layer Security

2.2.3 HandShake Protocol - Phase I : Establish security capabilities

- A CipherSuite





Ref: Wikimedia

2.2 Transport Layer Security

2.2.3 HandShake Protocol - Phase I : Establish security capabilities

- The first element of the Ciphersuite parameter is the key exchange method
- The following key exchange methods are supported
  - RSA
  - Fixed Diffie–Hellman
  - Ephemeral Diffie-Hellman
  - Anonymous Diffie–Hellman

2.2 Transport Layer Security

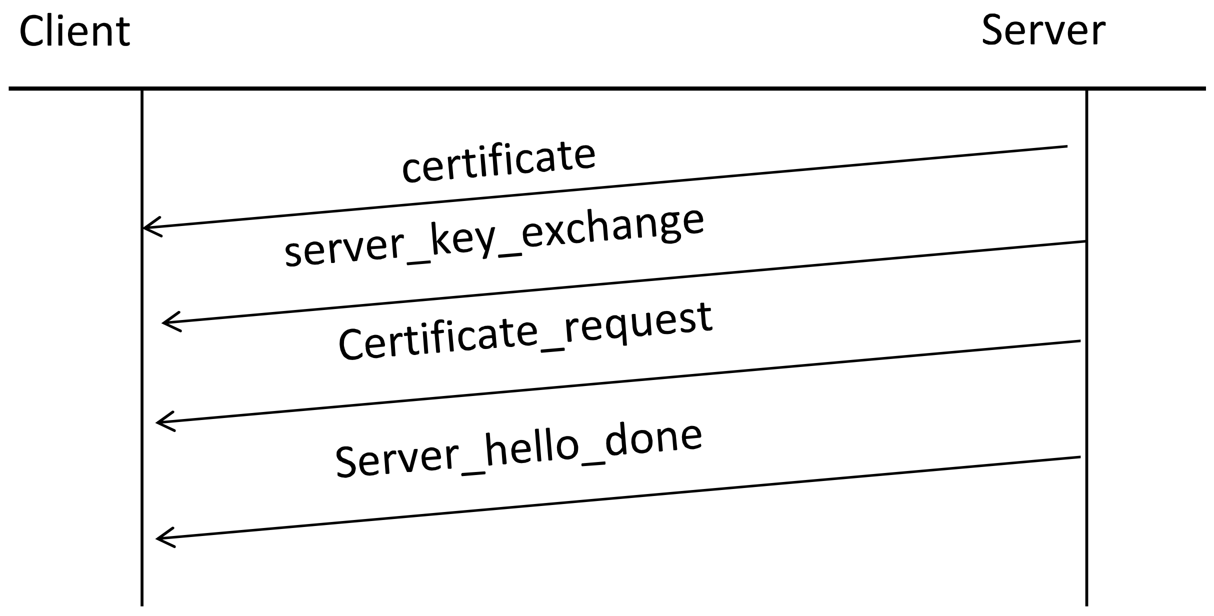
<u>2.2.3 HandShake Protocol - Phase I : Establish security capabilities</u>

- Following the definition of a key exchange method is the CipherSpec, which includes the following fields
  - CipherAlgorithm
  - MACAlgorithm
  - CipherType
  - IsExportable
  - HashSize
  - Key Material
  - IV Size

2.2 Transport Layer Security

2.2.3 HandShake Protocol - Phase II : Server authentication and key exchange

- Server may send certificate, key exchange, and request certificate. Server signals end of hello message phase.



2.2 Transport Layer Security

2.2.3 HandShake Protocol - Phase II : Server authentication and key exchange

- The **certificate** message is required for any agreed-on key exchange method except anonymous Diffie–Hellman. Note
- **server_key_exchange** message may be sent if it is required. It is not required in two instances:
    - The server has sent a certificate with fixed Diffie–Hellman parameters; or
    - RSA key exchange is to be used

2.2 Transport Layer Security

2.2.3 HandShake Protocol - Phase II : Server authentication and key exchange

- The **server_key_exchange** message is needed for the following:
  - Anonymous Diffie–Hellman
  - Ephemeral Diffie–Hellman
  - RSA key exchange (in which the server is using RSA but has a signature-only RSA key)

2.2 Transport Layer Security

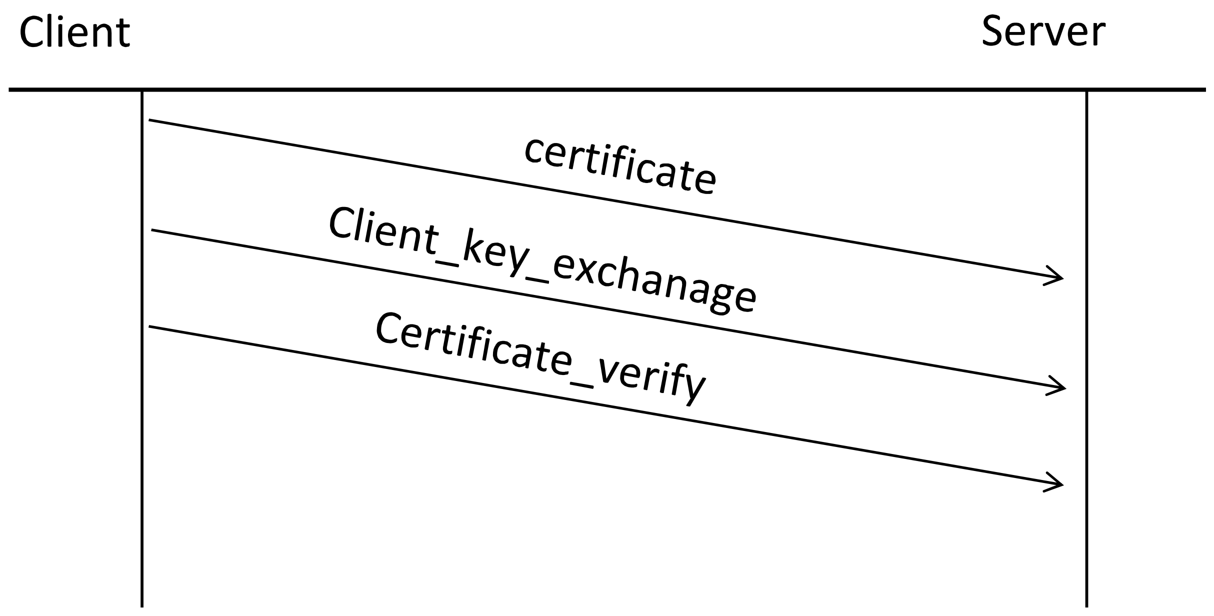
<u>2.2.3 HandShake Protocol - Phase II : Server authentication and key exchange</u>

- Next, a nonanonymous server (server not using anonymous Diffie–Hellman) can request a certificate from the client
  - The **certificate_request** message includes two parameters: certificate_type and certificate_authorities
  - The certificate type indicates the public-key algorithm and its use
  - The certificate_authorities in the **certificate_request** message is a list of the distinguished names of acceptable certificate authorities
  - The final message in phase 2, and one that is always required, is the **server_ done** message

2.2 Transport Layer Security

2.2.3 HandShake Protocol - Phase III : Client authentication and key exchange

- Client sends certificate if requested. Client sends key exchange. Client may send certificate verification.



2.2 Transport Layer Security

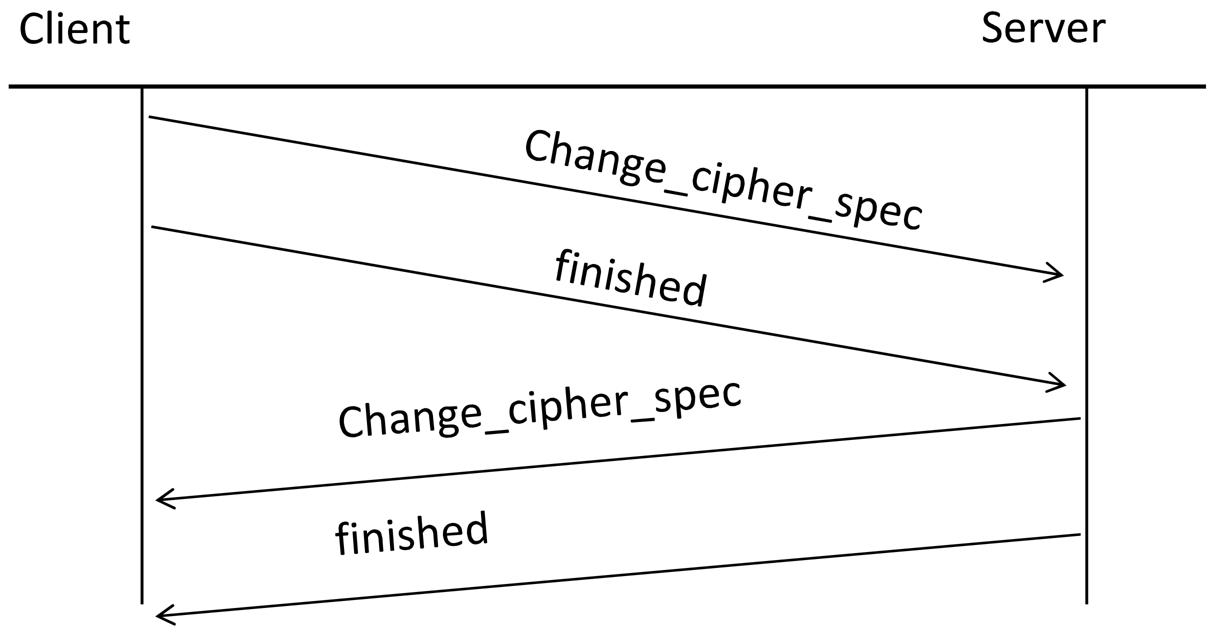
2.2.3 HandShake Protocol - Phase III : Client authentication and key exchange

- If the server has requested a certificate, the client begins this phase by sending a **certificate** message
- Next is the **client_key_exchange** message, which must be sent in this phase. The content of the message depends on the type of key exchange.
- Finally, in this phase, the client may send a **certificate_verify** message to provide explicit verification of a client certificate

2.2 Transport Layer Security

2.2.3 HandShake Protocol - Phase IV : Finish

- Change cipher suite and finish handshake protocol.



2.2 Transport Layer Security

2.2.3 HandShake Protocol - Phase IV : Finish

- The client sends a **change_cipher_spec** message and copies the pending CipherSpec into the current CipherSpec. This message is not considered part of the Handshake Protocol but is sent using the Change Cipher Spec Protocol
- The client then immediately sends the **finished** message under the new algorithms, keys, and secrets

2.2 Transport Layer Security

2.2.3 HandShake Protocol

- Cryptographic computations
    - Creation of a shared master secret by means of the key exchange
        - RSA
        - Diffie–Hellman
    - Generation of cryptographic parameters from the master secret
- Psudo Random Function

2.2 Transport Layer Security

2.2.3 Heartbeet Protocol

- In the context of computer networks, a heartbeat is a periodic signal generated by hardware or software to indicate normal operation or to synchronize other parts of a system
- The Heartbeat protocol runs on top of the TLS Record Protocol and consists of two message types
  - heartbeat_response
  - heartbeat_response
- The heartbeat serves two purposes
  - it assures the sender that the recipient is still alive
  - avoids closure of idle connection by a firewall that does not tolerate idle connections

2.2 Transport Layer Security

2.2.4 SSL/TLS Attacks

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

2.2 Transport Layer Security

<u>2.2.5 TLSv1.3</u>

- TLSv1.3 removes support for a number of options and functions
  - Compression
  - Ciphers that do not offer authenticated encryption
  - Static RSA and DH key exchange
  - 32-bit timestamp as part of the Random parameter in the client_hello message
  - Renegotiation
  - Change Cipher Spec Protocol
  - RC4
  - Use of MD5 and SHA-224 hashes with signatures

2.2 Transport Layer Security

2.2.5 TLSv1.3

- TLSv1.3 uses Diffie–Hellman or Elliptic Curve Diffie–Hellman for key exchange and does not permit RSA
- TLSv1.3 allows for a “1 round trip time” handshake by changing the order of message sent with establishing a secure connection

2.3 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
- 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

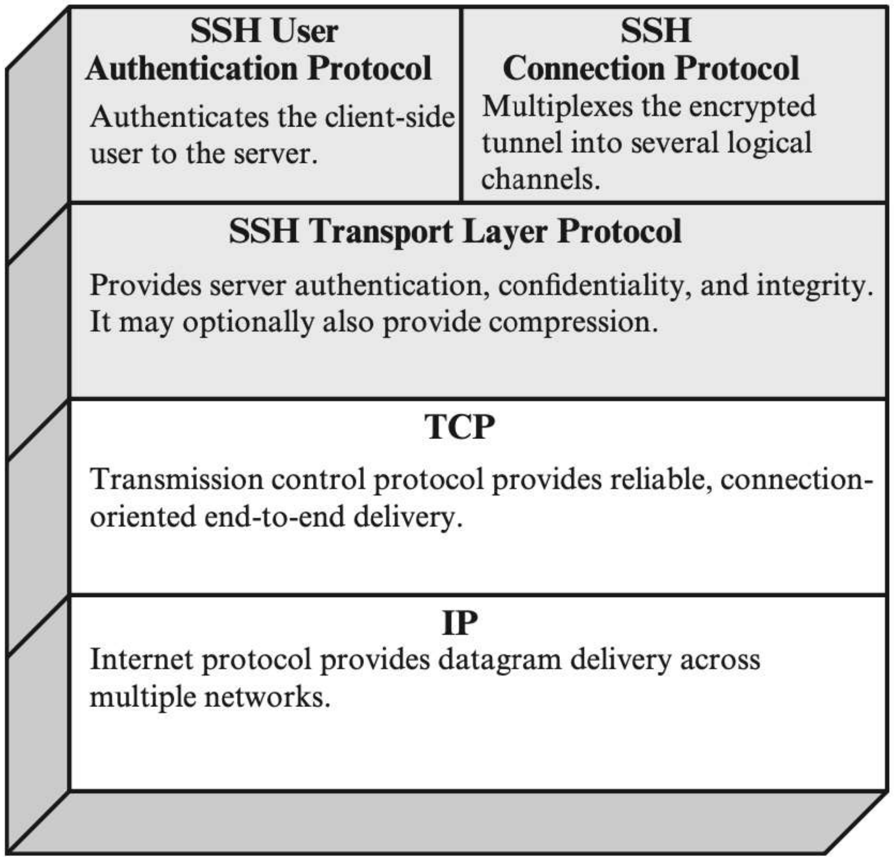
Study how HTTPS works using the knowledge of TLS protocol.

2.4 Secure Shell

- Secure Shell (SSH) is a protocol for secure network communications designed to be relatively simple and inexpensive to implement
- SSH is organized as three protocols that typically run on top of TCP
    - Transport Layer Protocol
    - User Authentication Protocol
    - Connection Protocol

2.4 Secure Shell

- SSH protocol stack


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.

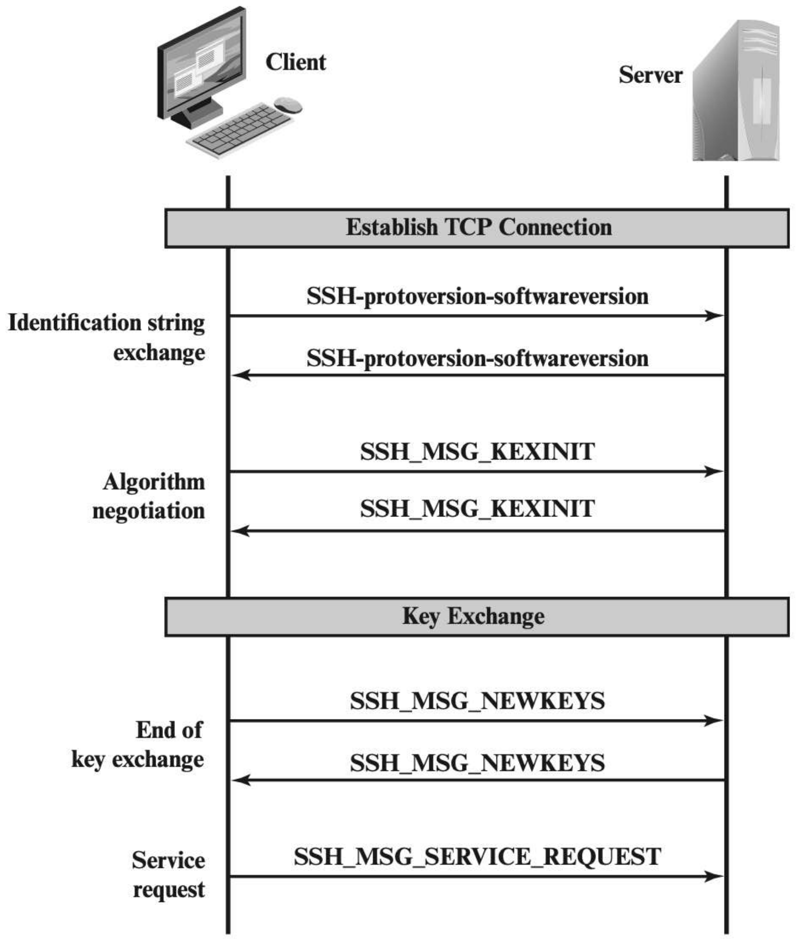
2.4 Secure Shell

2.4.1. Transport Layer Protocol

- Server authentication occurs at the transport layer, based on the server possessing a public/private key pair
- Two alternative trust models that can be used:
  - The client has a local database that associates each host name (as typed by the user) with the corresponding public host key
  - The host name-to-key association is certified by a trusted certification authority (CA)

2.4 Secure Shell

2.4.1. Transport Layer Protocol



2.4 Secure Shell

2.4.2. User Authentication Protocol

- The User Authentication Protocol provides the means by which the client is authenticated to the server
- The message exchange involves the following steps
    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. If the user name is valid, the server proceeds to step 3
    3. The server returns SSH_MSG_USERAUTH_FAILURE with a list of one or more authentication methods to be used.

2.4 Secure Shell

2.4.2. User Authentication Protocol

- The message exchange cont.

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

2.4 Secure Shell

2.4.2. User Authentication Protocol

- Authentication methods
    - Public key
    - password
    - Host based

2.4 Secure Shell

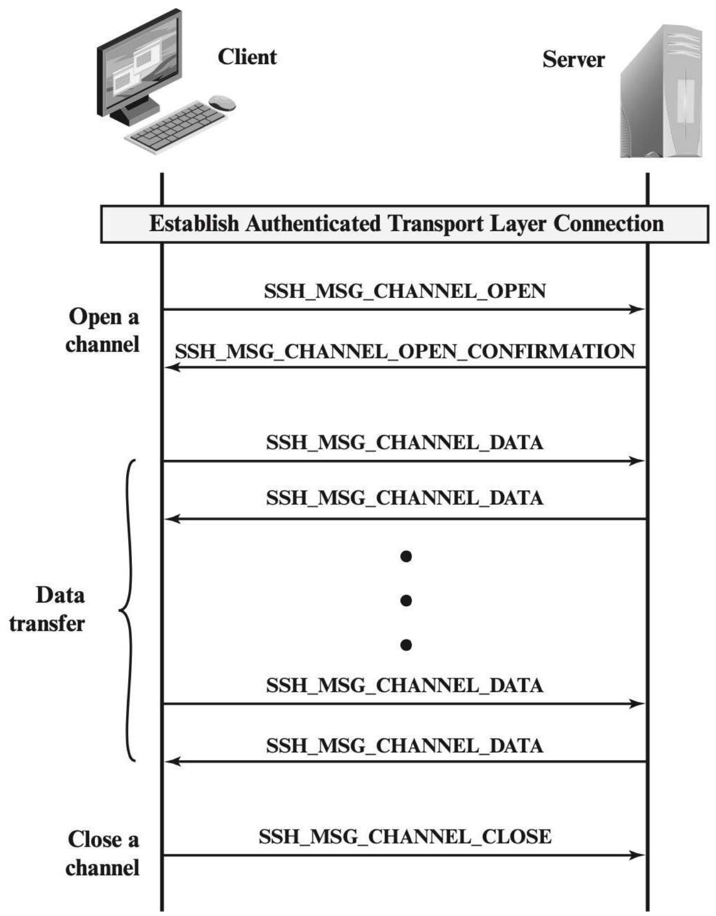
2.4.3. Connection Protocol

- The SSH Connection Protocol runs on top of the SSH Transport Layer Protocol and assumes that a secure authentication connection is in use
- That secure authentication connection, referred to as a **tunnel**, is used by the Connection Protocol to multiplex a number of logical channels
- Four channel types are recognized in the SSH Connection Protocol specification
    - Session
    - X11
    - forwarded-tcpip
    - direct-tcpip

2.4 Secure Shell

2.4.3. Connection Protocol

- Connection Protocol message exchange



2.4 Secure Shell

<u>2.4.3. Connection Protocol</u>

- SSH port forwarding
  - port forwarding provides the ability to convert any insecure TCP connection into a secure SSH connection. This is also referred to as SSH **tunneling**
  - SSH supports two types of port forwarding:
    - Local forwarding
    - Remote forwarding

References

Ref 1. Cryptography and Network Security, Principles and Practice, 7th Edition, William Stallings