Chapter 16

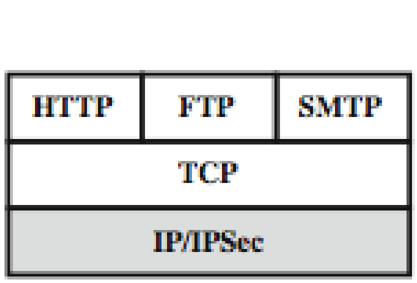
Transport-Level Security

Web Security

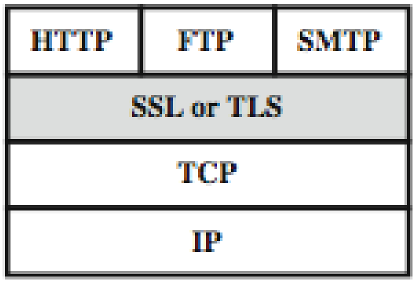
- The World Wide Web is fundamentally a client/server application running over the Internet and widely used by businesses, government agencies, and many individuals. But the Internet and the Web are extremely vulnerable .
- Number of security threats faced when using the Web in terms of passive and active attacks.
  - **Passive attacks** including eavesdropping on network traffic between browser and server and gaining access to information on a Web site that is supposed to be restricted;
  - **Active attacks** including impersonating another user, altering messages in transit between client and server, and altering information on a Web site.
- **These types can be classified according to:**
  - Integrity
  - Confidentiality
  - Denial of Service
  - Authentication
- The web needs added security mechanisms to address these 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 • Vulnerabilty to all other threats	Cryptographic checksums
Confidentiality	• Eavesdropping on the net • Theft of info from server • Theft of data from client • Info about network configuration • Info about which client talks to server	• Loss of information • Loss of privacy	Encryption, Web proxies
Denial of Service	• Killing of user threads • Flooding machine with bogus requests • Filling up disk or memory • Isolating machine by DNS attacks	• Disruptive • Annoying • Prevent user from getting work done	Difficult to prevent
Authentication	• Impersonation of legitimate users • Data forgery	• Misrepresentation of user • Belief that false information is valid	Cryptographic techniques

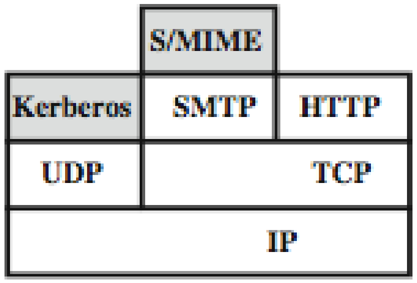
Web Traffic Security Approaches



(a) Network Level

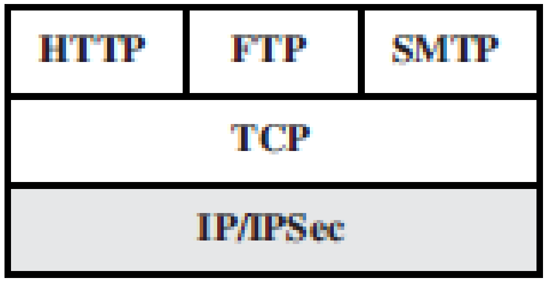


(b) Transport Level



(c) Application Level

(a) IP security (IPsec)

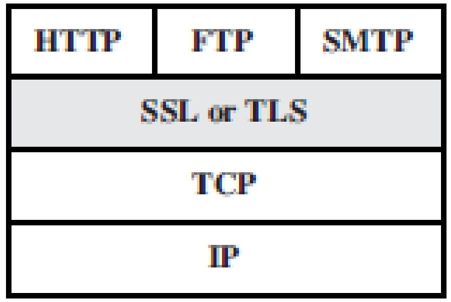


(a) Network level

The advantages:

- It is transparent to end users and applications and provides a general-purpose solution.
- It includes a filtering capability so that selected traffic need incur the overhead of IPsec processing.

(b) Secure Sockets Layer/Transport Layer Security (SSL/TLS)

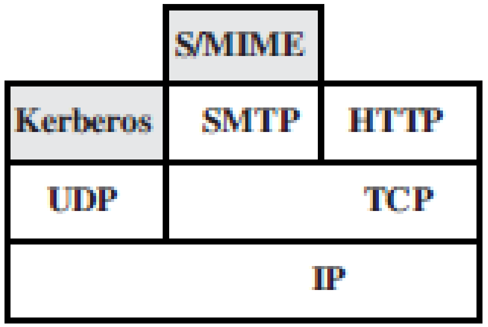


(b) Transport level

- Part of the underlying protocol and therefore be transparent to applications.
- Embedded into packages (e.g. Browsers)

  Explorer browsers come with SSL & Most Web servers have implemented it.

(c) Application-specific security services:



(c) Application level

- Embedded within particular application.

The advantage:

- – The service can be adapted to the specific needs of a given application.

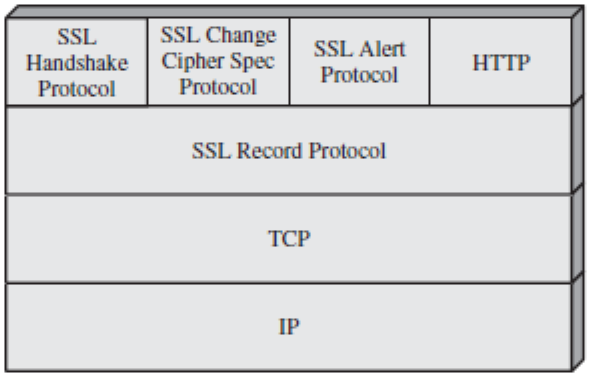
SSL (Secure Socket Layer) and TLS (Transport Layer Security)

- Most widely used Web security mechanism [implemented at the Transport layer].

- 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 protocol.

SSL Architecture



- **SSL Record Protocol** provides basic security services to various higher-layer protocols.
- **Hypertext Transfer Protocol (HTTP)**, provides transfer service for Web client/server interaction, can operate on top of SSL.

- **Three higher-layer protocols are defined as part of SSL and used in the management of SSL exchanges :**
  1. Handshake Protocol,
  2. Change Cipher Spec Protocol
  3. Alert Protocol.

SSL Concepts

- **SSL connection**
  - A connection is a transport that provides a suitable type of service, such connections are transient, peer-to-peer relationships, associated with one SSL session.
- **SSL session**
  - An association between client & server
  - 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.

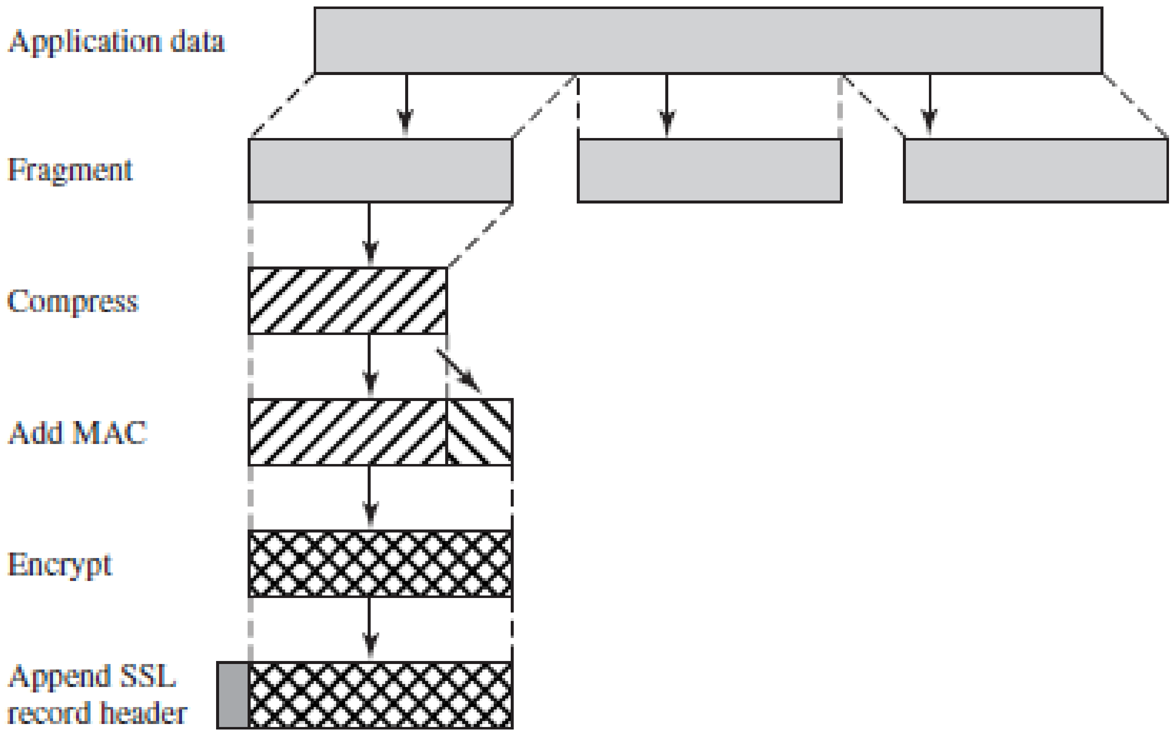
SSL Architecture

- Between any pair of parties, with multiple secure connections [applications such as HTTP on client and server].
- There are a number of states associated with each session.
- Once a session is established, there is a current operating state for both read and write (i.e., receive and send).
- In addition, during the Handshake Protocol, pending read and write states are created.
- Upon successful conclusion of the Handshake Protocol, the pending states become the current states.
- A session state and a connection state are defined by sets of parameters.

SSL Record Protocol

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

SSL Record Protocol Operation

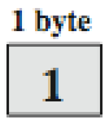


- **Fragmentation:** Each upper-layer message is fragmented into blocks of 2^{14} bytes (16384 bytes) or less.
- **Compression** (optionally)**:** Compression must be lossless and may not increase the content length by more than 1024 bytes.
- **Message Authentication Code (MAC):** over the compressed data. For this purpose, a shared secret key is used.

- The Record Protocol takes an application message to be transmitted, fragments the data into blocks
- Optionally compresses the data,
- Apply a **Message Authentication Code (**MAC**)**,
- Encrypts (symmetric algorithms),
- Append SSL record header.
- Transmits the resulting unit in a TCP segment.
- Received data are decrypted, verified, decompressed, and reassembled and then delivered to higher-layer applications.

SSL Change Cipher Spec Protocol

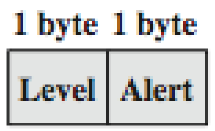
- The simplest one of the three SSL-specific protocols that use the SSL Record Protocol.
- A single message [Single byte with the value 1].
- The purpose of this message is to cause the pending state to be copied into the current state, which updates the cipher suite to be used on this connection.



(a) Change Cipher Spec Protocol

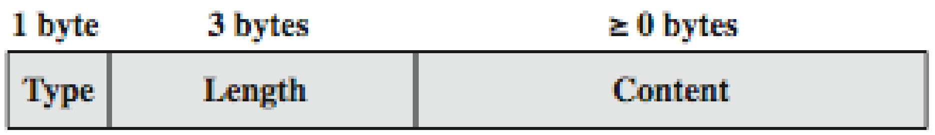
SSL Alert Protocol

- Convey SSL-related alerts to the peer entity.
- Alert messages are compressed and encrypted.
- Each message consists of two bytes.
  - The first byte takes the value warning (1) or fatal (2) to convey the severity of the message.
    - If the level is fatal,
      - SSL immediately terminates the connection.
      - Other connections in same session continue.
      - No new connections allowed.
  - The second byte contains a code that indicates the specific alert.



(b) Alert Protocol

Handshake Protocol

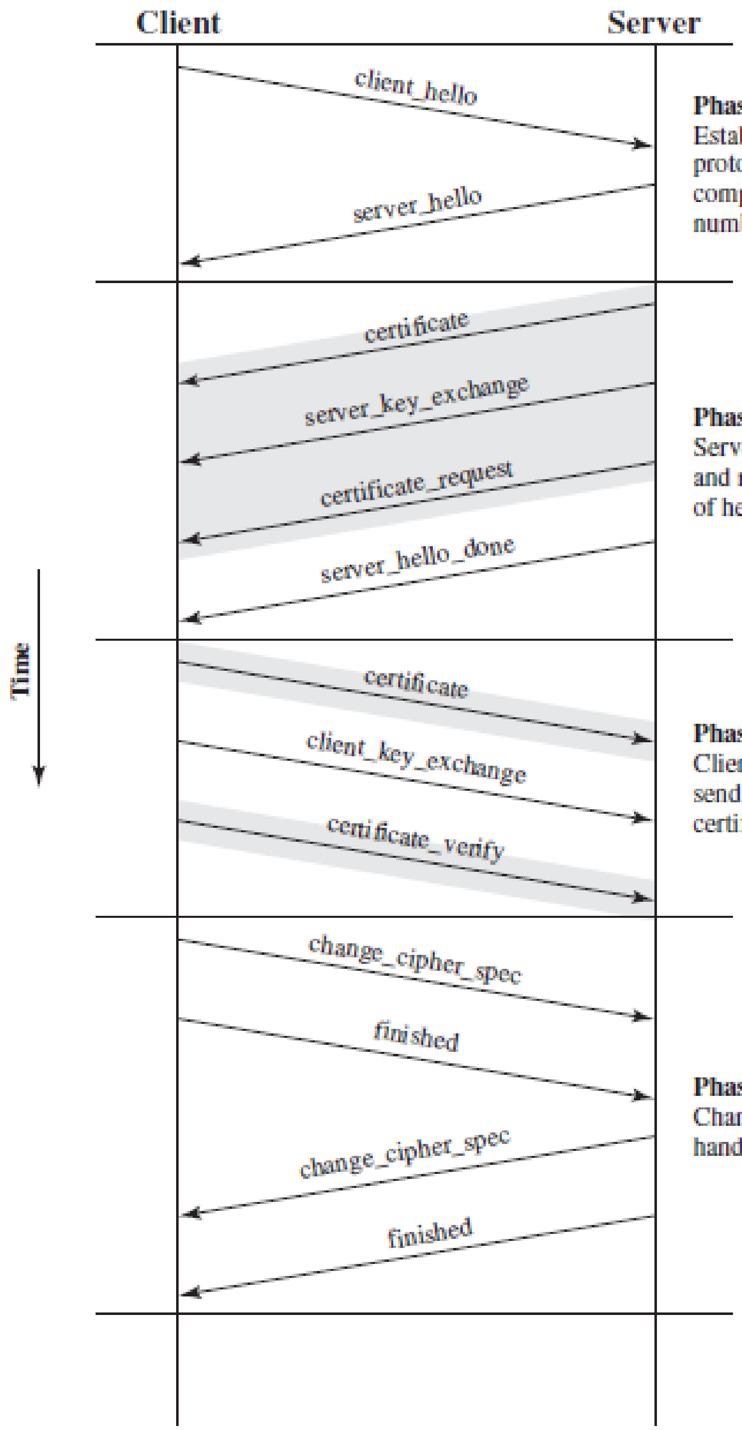


(c) Handshake Protocol

- Most complex part of SSL
- Allows server and client to
  - Authenticate each other
  - negotiate an encryption, MAC algorithm and cryptographic keys to protect data sent in an SSL record.

Consists of 4 phases:

- ***PHASE 1. ESTABLISH SECURITY CAPABILITIES***
  - Used by the client to initiate a logical connection and to establish the security capabilities that will be associated with it.
- ***PHASE 2. SERVER AUTHENTICATION AND KEY EXCHANGE***
  - The server send its certificate if it needs to be authenticated.
- ***PHASE 3. CLIENT AUTHENTICATION AND KEY EXCHANGE***
  - The client should verify that the server provided a valid certificate and check that the **server_hello** parameters are acceptable
- ***PHASE 4. FINISH***
  - 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 finished message verifies that the key exchange and authentication processes were successful.



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.

Cryptographic Computations

(1) Creation of a shared master secret by means of the key exchange.

- The shared master secret is a one-time 48-byte value (384 bits) for this session
- The creation is in two stages:
  - First, a **pre_master_secret** is exchanged. Using either RSA or Diffie-Hellman
  - Second, the **master_secret** is calculated by both parties. By hashing the relevant information

(2) Generation of cryptographic parameters from the master secret.

- Client write MAC secret, a server write MAC secret, a client write key, a server write key, a client write IV, and a server write IV which are generated from the master secret in that order
- These parameters are generated by hashing master secret into a sequence of secure bytes of sufficient length for all needed parameters.

TLS (Transport Layer Security)

- TLS is IETF standard RFC 5246 similar to SSL V.3
- **Minor differences**
  - Record format version number
  - Using HMAC for MAC
  - A pseudo-random function expands secrets
    - based on HMAC using SHA-1 or MD5
  - Additional alert codes
  - Some changes in supported ciphers
  - Changes in certificate types & negotiations
  - Changes in crypto computations & padding

HTTPS

- **HTTPS (HTTP over SSL)**
  - Combination of HTTP & SSL/TLS to secure communications between browser & server
    - documented in RFC2818
    - no fundamental change using either SSL or TLS
  - The HTTPS capability is built into all modern Web browsers
- **Using https:// URL rather than http://**
  - Port 443 rather than 80
- **Encrypts**
  - URL, document contents, form data, cookies, HTTP headers

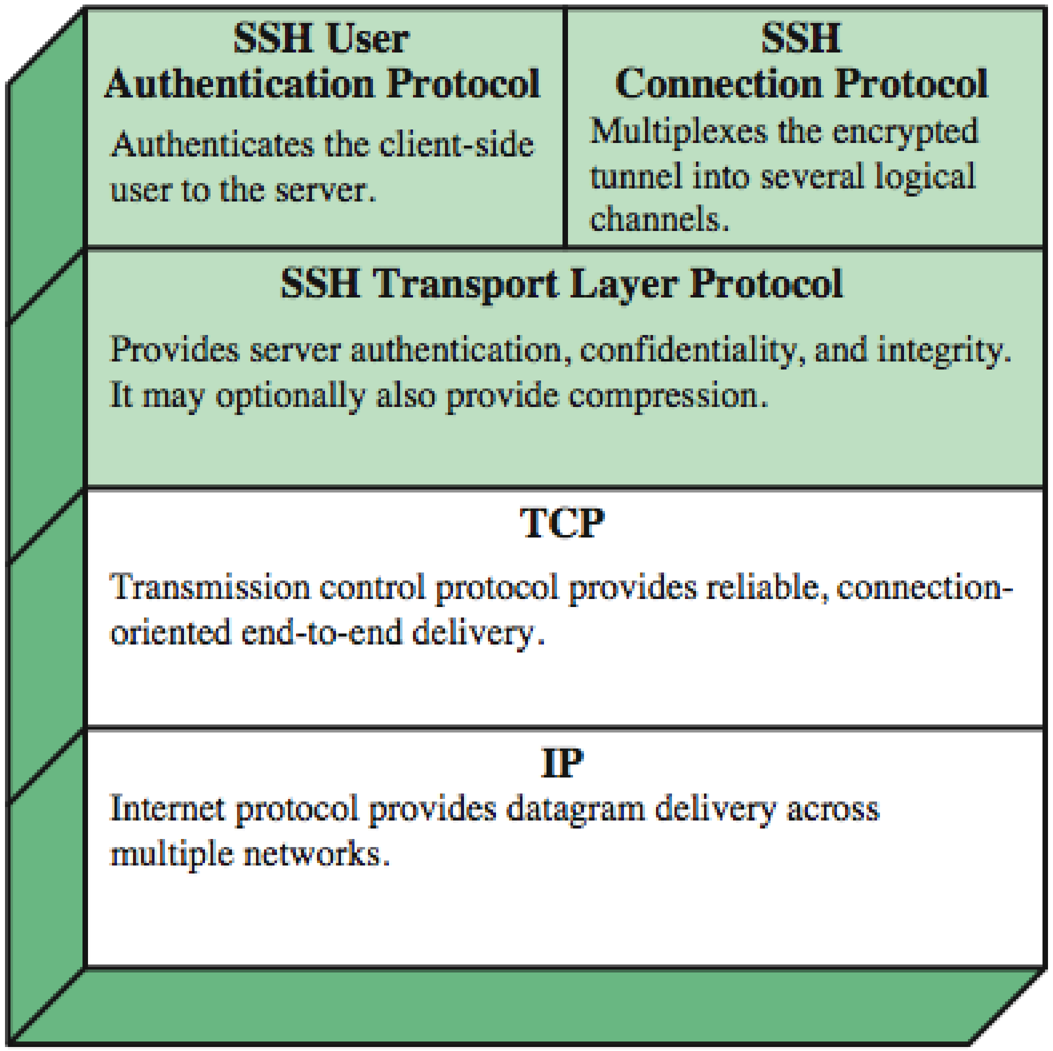
HTTPS Use

- Connection initiation (The client initiates a connection to the server on the appropriate port)
  - TLS handshake then HTTP request(s)
  - All HTTP data is to be sent as TLS application data
- Connection closure
  - An HTTP client or server can indicate the closing of a connection by including 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.
  - Can then close TCP connection

Secure Shell (SSH)

- Protocol for secure network communications
  - Designed to be simple & inexpensive
- SSH1 provided secure remote logon facility
  - Replace TELNET & other insecure schemes
  - Also has more general client/server capability
- SSH2 fixes a number of security flaws
- SSH clients & servers are widely available

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.

SSH Transport Layer Protocol

- Server Authentication occurs at transport layer, based on server/host key pair(s)
  - Server authentication requires clients to know host keys in advance

- Packet Exchange
  - Establish TCP connection
  - Exchange data
    - identification string exchange, algorithm negotiation, key exchange, end of key exchange, service request
  - Using specified packet format

The SSH Transport Layer packet exchange steps:

1. **The client sending a packet with an identification string.**

2. **Algorithm negotiation:** Each side sends an **SSH_MSG_KEXINIT** containing lists of supported algorithms, one list for each type of cryptographic algorithm, in the order of preference to the sender. For each category, the algorithm chosen is the first algorithm on the client's list that is also supported by the server.

3. **Key exchange:** Only two versions of Diffie-Hellman key exchange are specified. As a result of these steps, the two sides now share a master key K. In addition, the server has been authenticated to the client.

4. **End of key exchange:** By the exchange of **SSH_MSG_NEWKEYS** packets.

5. **Service request:**
   - The client sends an **SSH_MSG_SERVICE_REQUEST** packet to request either the User Authentication or the Connection Protocol.
   - All data is exchanged as the payload of an SSH Transport Layer packet, protected by encryption and MAC.

SSH User Authentication Protocol

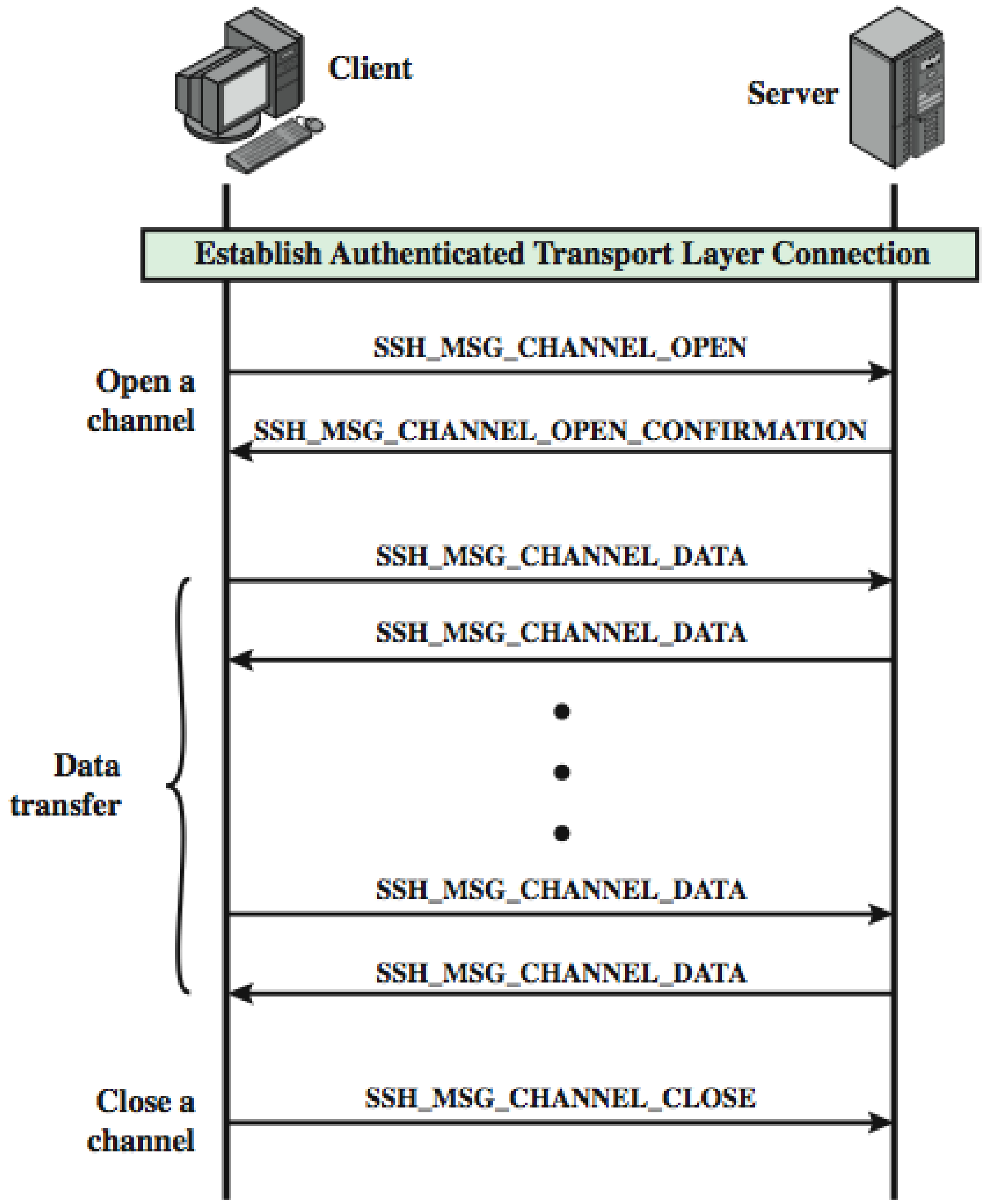
- **Authenticates client to server**
- **Three message types:**
  - **SSH_MSG_USERAUTH_REQUEST:** Authentication requests from the client
  - If the server rejects the authentication request, or accepts the request but requires one or more additional authentication methods, the server sends a **SSH_MSG_USERAUTH_FAILURE**
  - If the server accepts authentication then it sends a single byte message, **SSH_MSG_USERAUTH_SUCCESS**.

- **The server may require one or more of the following authentication methods:**
  - **Public-key:** client sends a message to the server that contains the client's public key, with the message signed by the client's private key
  - **Password:** client sends a message containing a plaintext password, protected by TLS encryption
  - **Host-based:** authentication is performed on the client's host, rather than the client itself

SSH Connection Protocol

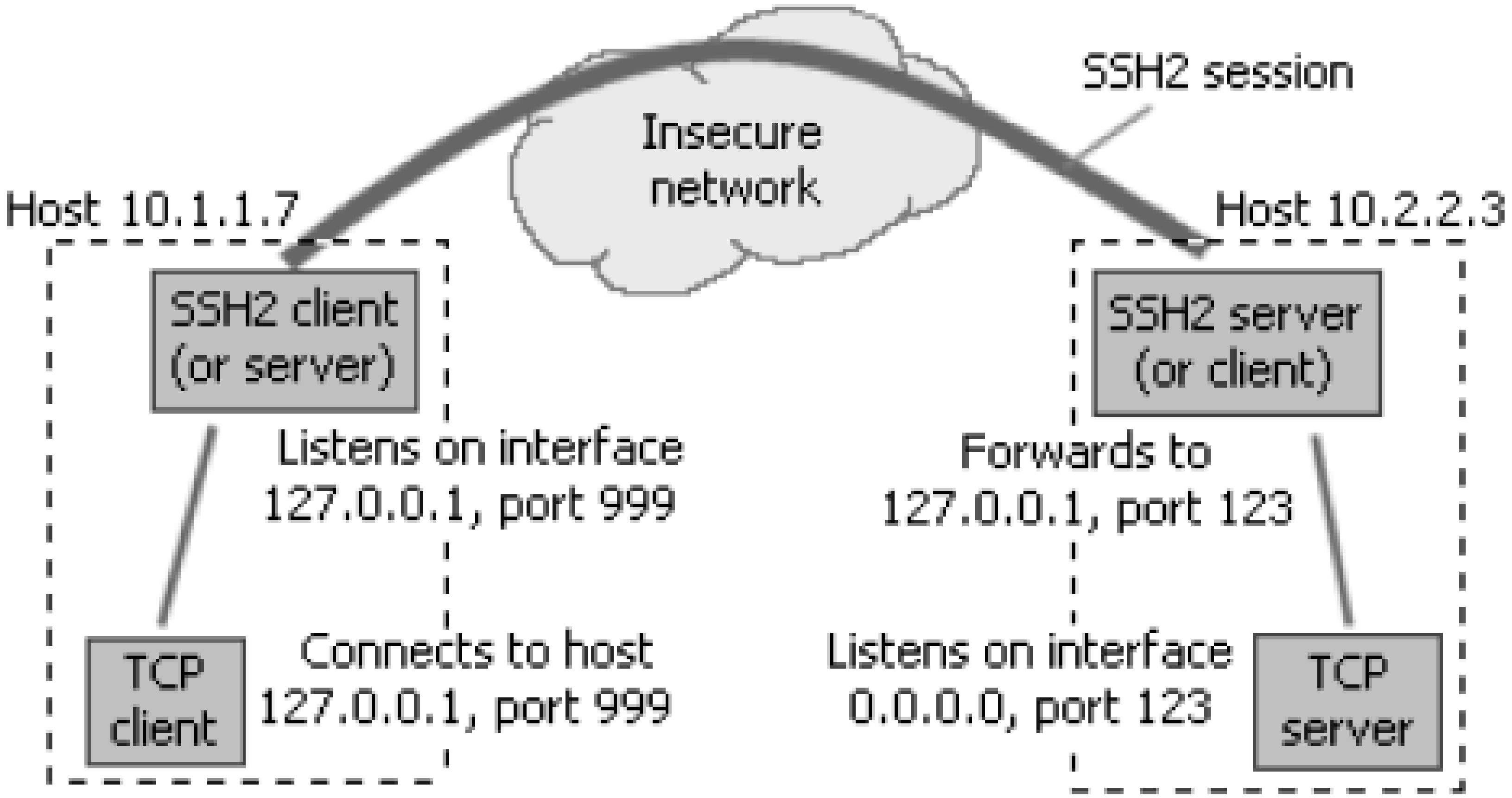
- Runs on top of the SSH Transport Layer Protocol
- Assumes secure authentication connection referred to as a **tunnel**
- Used for multiple logical channels (Either side may open a channel. For each channel, each side associates a unique channel number)
  - SSH communications use separate channels
  - either side can open with unique id number
  - flow controlled
  - The life of a channel progresses through three stages:
    - opening a channel, data transfer, closing a channel
  - Four channel types :
    - Session: remote execution of a program
    - X11: X Window System display traffic
    - forwarded-tcpip: remote port forwarding (**Remote TCP to local TCP**)
    - direct-tcpip: local port forwarding (**Local TCP to remote TCP**)

SSH Connection Protocol Exchange



Port Forwarding

- One of the most useful features of SSH is port forwarding / tunneling.
- It convert insecure TCP connection into a secure SSH connection
  - SSH Transport Layer Protocol establishes a TCP connection between SSH client & server
  - client traffic redirected to local SSH, travels via tunnel, then remote SSH delivers to server
- supports two types of port forwarding
  - local forwarding – hijacks selected traffic
    - 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. This could be used to secure the traffic from an email client on your desktop that gets email from the mail server via POP, the Post Office Protocol
  - remote forwarding – client acts for server
    - 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. This could be used to securely access a server at work from a home computer.


SSH2 session
Insecure network
Host 10.1.1.7
Host 10.2.2.3
SSH2 client (or server)
SSH2 server (or client)
Listens on interface 127.0.0.1, port 999
Forwards to 127.0.0.1, port 123
TCP client
Connects to host 127.0.0.1, port 999
Listens on interface 0.0.0.0, port 123
TCP server

Port Forwarding

- On each of the client workstations in Building #1 (workstation 10.1.1.7), you install an SSH client. On the machine in Building #2 that runs the server for your application, you install an SSH server.

- You configure SSH client with the following client-to-server port forwarding rule: for each connection that comes on interface 127.0.0.1 and port 999, forward that connection to SSH server, and request SSH server to forward that connection to host 127.0.0.1 (relative to the server), port 123.

- Now, your application client doesn't connect to the server directly. Rather, it connects to the SSH client, which encrypts all data before transmission.

- SSH client forwards the encrypted data to SSH server, which decrypts it and forwards it to your application server. Data sent by the server application is similarly encrypted by the SSH server and forwarded back to the client.