Transport Level Security

Raj Jain
Washington University in Saint Louis
Saint Louis, MO 63130
Jain@cse.wustl.edu

Audio/Video recordings of this lecture are available at:

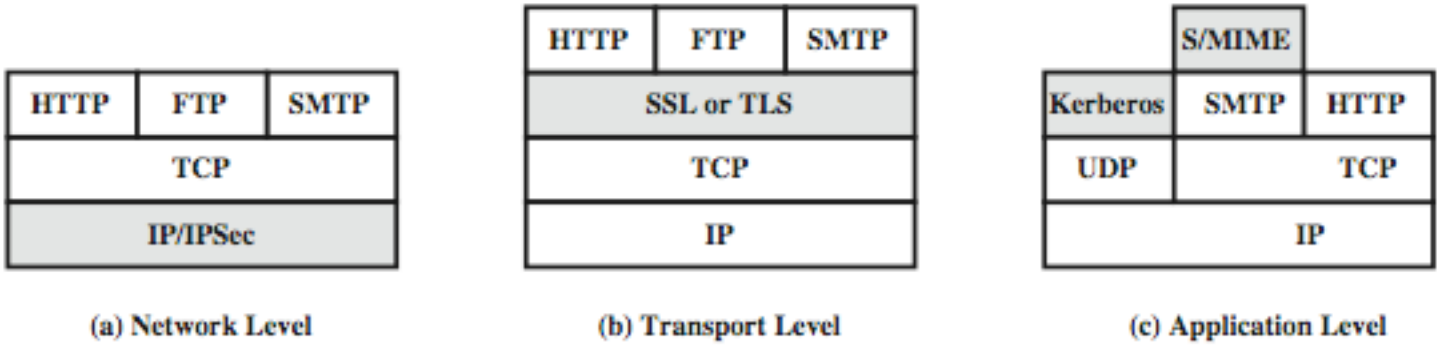
http://www.cse.wustl.edu/~jain/cse571-14/

Overview

1. Secure Sockets Layer (SSL)
2. Transport Layer Security (TLS)
3. HTTPS
4. Secure Shell (SSH)

These slides are based partly on Lawrie Brown's slides supplied with William Stallings's book "Cryptography and Network Security: Principles and Practice," 6th Ed, 2013.

Web Traffic Security Approaches



- SSL/TLS provides the following services **over TCP** layer :
  1. **Crypto negotiation**: Negotiate encryption and hash methods
  2. **Key Exchange**: Secret key exchange using public key certificates
  3. **Privacy**: Encryption using secret key
  4. **Integrity**: Message authentication using a keyed hash

History

- SSL was developed by Netscape. V1 was never deployed. V2 had major issues.
- SSL v3 is most commonly deployed protocol
- IETF standardized SSL V3 with some upgrades as Transport Layer Security (TLS) V1 in RFC 2246 1999
  TLS is encoded as SSL V3.1
  The differences are small but the protocols do not interoperate.
- TLS v1.1 (SSL V3.2) added protection against CBC attacks [RFC 4346 2006]

Ref: http://en.wikipedia.org/wiki/Transport_Layer_Security

History (Cont)

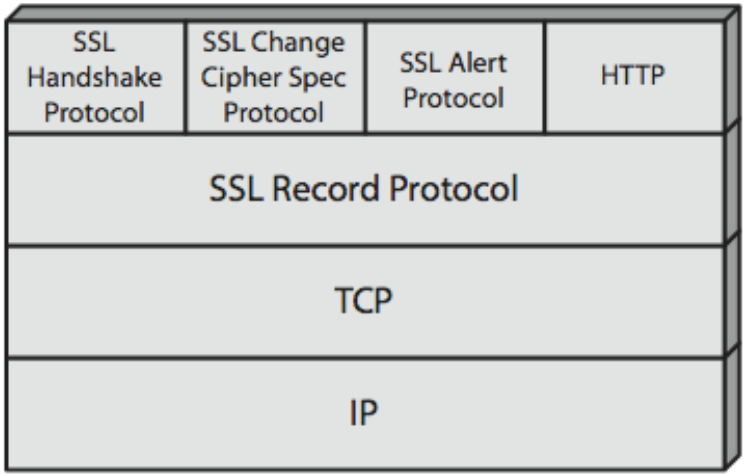
- TLS v1.2 (SSL V3.3) in RFC 5246 August 2008 added:
  - MD5-SHA-1 pseudorandom function (PRF) replaced with SHA-256
  - MD5-SHA-1 Finished message hash replaced with SHA-256
  - MD5-SHA-1 in digitally-signed element replaced with a single hash negotiated during handshake, default=SHA-1.
  - Enhanced Client's and server's specification for hash and signature algorithms
  - Expansion of support for authenticated encryption ciphers
  - TLS Extensions definition and Advanced Encryption Standard Cipher Suites
- RFC 6176 updated TLS v1.2 by requiring that SSL V2 is never accepted.

Ref: http://en.wikipedia.org/wiki/Transport_Layer_Security#TLS_1.3_.28draft.29 ←Must Read

SSL Architecture

- SSL has 4 components in two layers

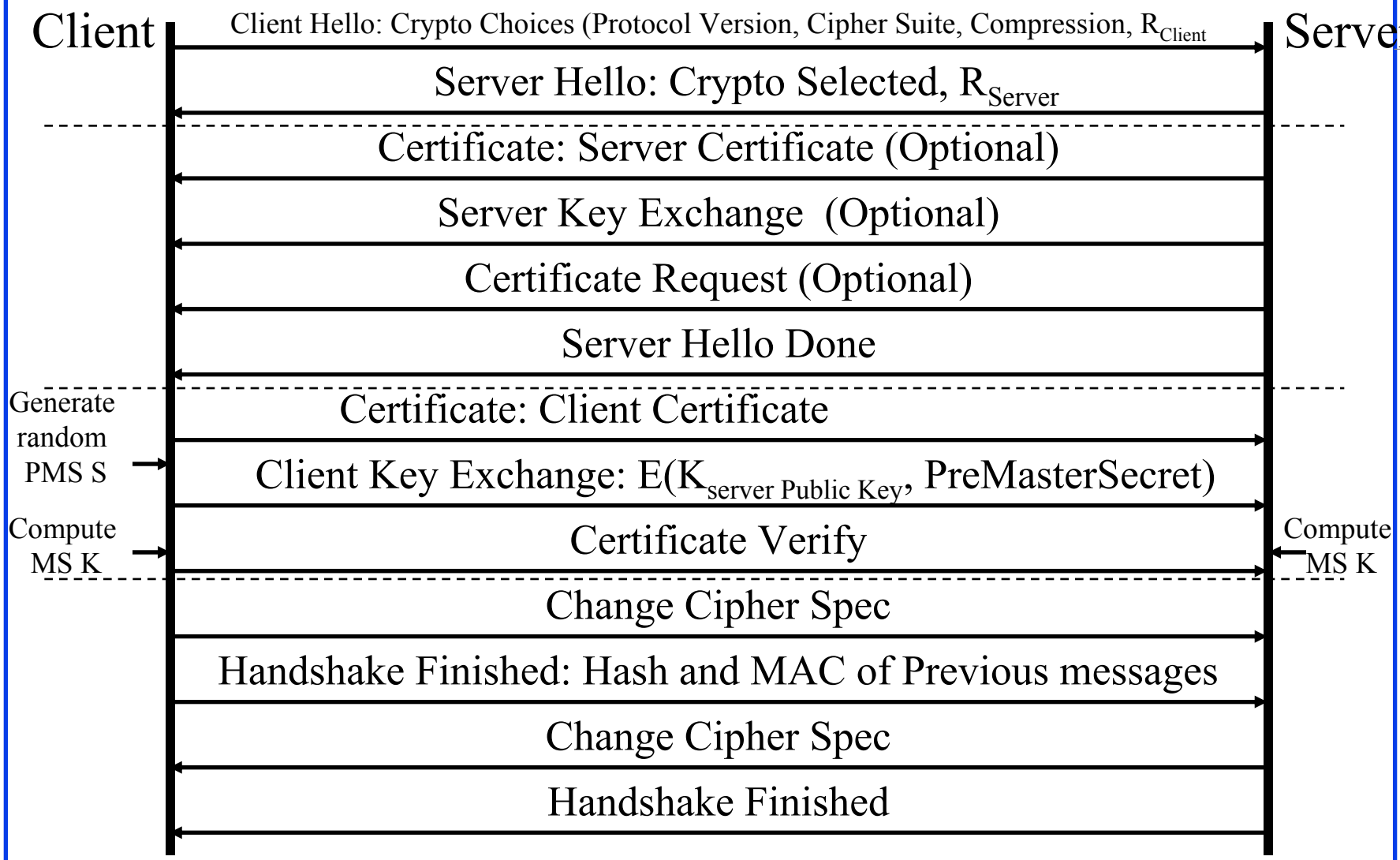
1. **Handshake protocol**: Negotiates crypto parameters for a "SSL session" that can be used for many "SSL/TCP connections"
2. **Record Protocol**: Provides encryption and MAC
3. **Alert protocol**: To convey problems
4. **Change Cipher Spec Protocol**: Implement negotiated crypto parameters



SSL Handshake Protocol

- Allows server and client to:
  - Authenticate each other
  - To negotiate encryption & MAC algorithms
  - To negotiate cryptographic keys to be used
- Comprises a series of messages in phases
  1. Establish Security Capabilities
  2. Server Authentication and Key Exchange
  3. Client Authentication and Key Exchange
  4. Finish

SSL Handshake Protocol Actions



Handshake Messages

All messages are Type-Length-Value (TLV) encoded.

Types

1 = Client Hello: Highest Version Supported, R_{Client}, Session ID, Cipher Suites, Compressions

2 = Server Hello: Version Accepted, R_{Server}, Session ID, Chosen Cipher, Chosen Compression

14 = Server Hello Done

16 = Client Key Exchange: Encrypted pre-master key

12 = Server Key Exchange: Modulus p, Exponent g, Signature (export only)

13 = Certificate Request: CA Names (requested by server)

11 = Certificate: sent by server

15 = Certificate Verify: Signature of Hash of messages

20 = Handshake Finished: MD5 and SHA Digest of message halves

Security Capability Negotiation

- Key-Exchange Methods:
  - RSA
  - Fixed D-H: Shared secret generated using fixed public keys
  - Ephemeral D-H: Ephemeral = Temporary, one-time secret key is generated after certificate exchange and authentication
  - Anonymous D-H: No authentication. Only public key exchange. Subject to MITM attack
  - Fortezza: Using PC-Cards (http://en.wikipedia.org/wiki/Fortezza)
- CipherSpec:
  - Cipher Algorithm: RC4, RC2, DES, 3DES, DES40, IDEA, or Fortezza
  - MAC Algorithm: MD5 or SHA-1
  - CipherType: Stream or Block
  - IsExportable: True or False
  - HashSize: 0, 16 (for MD5), or 20 (for SHA-1) bytes
  - Key Material: info used to generate keys
  - IV Size: Size of IV for CBC

Ref: http://en.wikipedia.org/wiki/Cipher_suite

Cryptographic Computations

- Master secret creation
  - A one-time 48-byte value based on nonces
  - A 48-byte pre-master secret is exchanged/generated using secure key exchange (RSA / Diffie-Hellman) and then hashing:
  - *Master_Secret = MD5(Pre_master_Secret || SHA('A') || pre_master_secret || clientHello.random || ServerHello.random)) ||....*
- Generation of cryptographic parameters
  - 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
  - Generated by hashing master secret

SSL Change Cipher Spec Protocol

- A single 1-byte message
- Causes negotiated parameters to become current
- Hence updating the cipher suite in use

(a) Change Cipher Spec Protocol

SSL Alert Protocol

Conveys SSL-related alerts to peer entity

Two byte message: Level-Alert, level = warning or fatal,
fatal ⇒ Immediate termination

0 Close notify (warning or fatal)

10 Unexpected message (fatal)

20 Bad record MAC (fatal)

21 Decryption failed (fatal, TLS only)

22 Record overflow (fatal, TLS only)

41 No certificate (SSL v3 only) (warning or fatal)

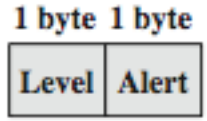
42 Bad certificate (warning or fatal)

43 Unsupported certificate (warning or fatal)

44 Certificate revoked (warning or fatal)

45 Certificate expired (warning or fatal)

….

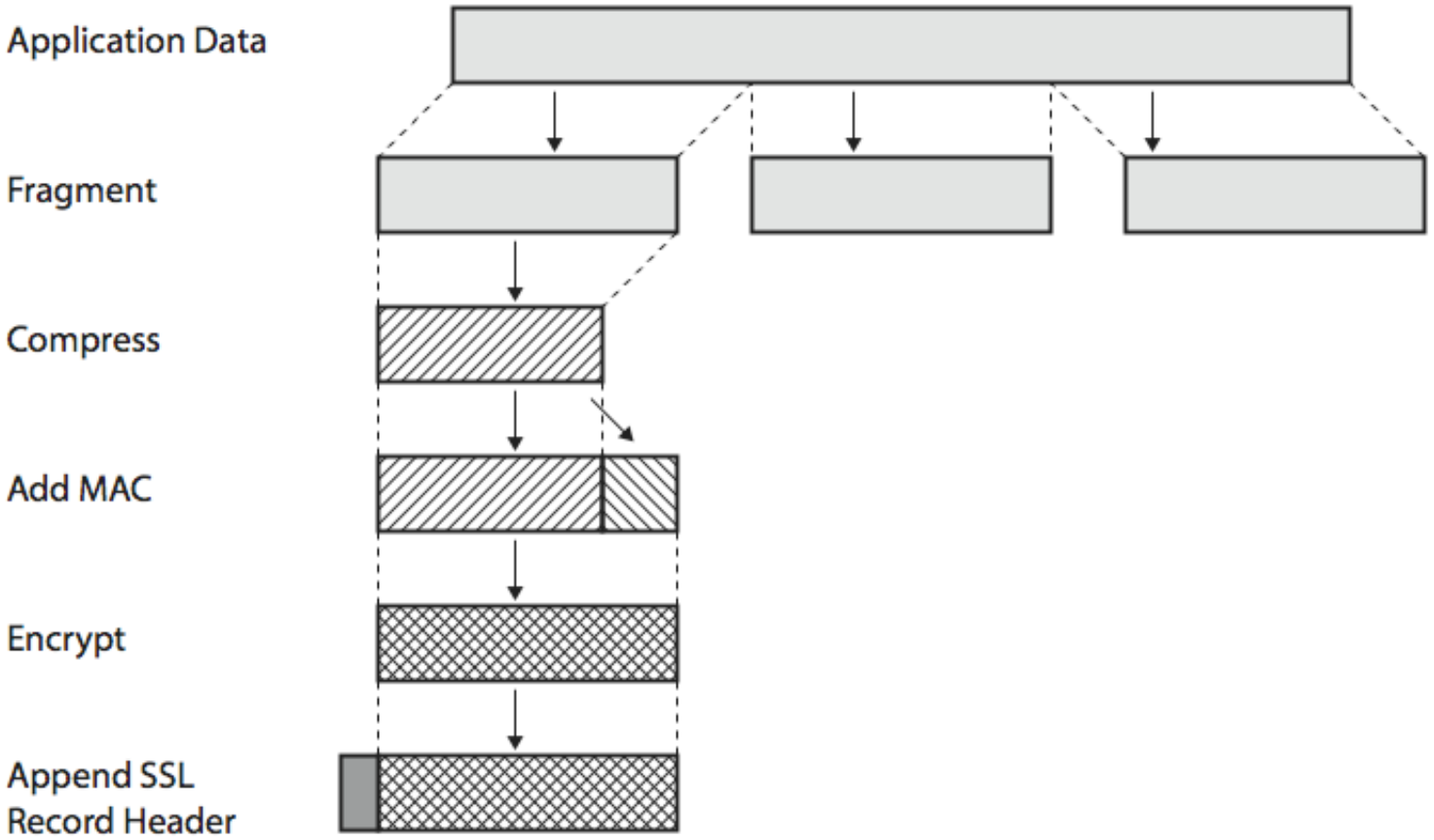


(b) Alert Protocol

SSL Record Protocol Services

- **Confidentiality**
  - Using symmetric encryption with a shared secret key defined by Handshake Protocol
  - AES, IDEA, RC2-40, DES-40, DES, 3DES, Fortezza, RC4-40, RC4-128
  - Message is compressed before encryption
- **Message integrity**
  - Using a MAC with shared secret key
  - Similar to HMAC but with different padding

SSL Record Protocol Operation



Encoding

- All exchanges are in records up to 2^{14}B or 2^{16-1}B.
- Standard allows multiple messages in one record or multiple records.
- Most implementations use one message per record.
- Four Record Types:
  - 20 = Change Cipher Spec
  - 21 = Alerts (1 = Warning, 2 = Fatal)
  - 22 = Handshake
  - 23 = Application Data
- Record header:

Record Type	Version #	Rec Length
1B	2B	2B

- Each message starts with a 1B message-type and 3B message length.

Msg Type	Msg Len	Msg

TLS (Transport Layer Security)

- IETF standard RFC 2246 similar to SSLv3
- With minor differences
  - In record format version number
  - Uses HMAC for MAC
  - A pseudo-random function expands secrets
    - Based on HMAC using SHA-1 or MD5
  - Has 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
- Use https:// URL rather than http://
  - And port 443 rather than 80
- Encrypts URL, document contents, form data, cookies, HTTP headers

Ref: http://en.wikipedia.org/wiki/HTTP_Secure

HTTPS Use

- Connection initiation
  - TLS handshake then HTTP request(s)
- Connection closure
  - Have "Connection: close" in HTTP record
  - TLS level exchange close_notify alerts
  - Can then close TCP connection
  - Must handle abnormal TCP close before alert exchange sent or completed

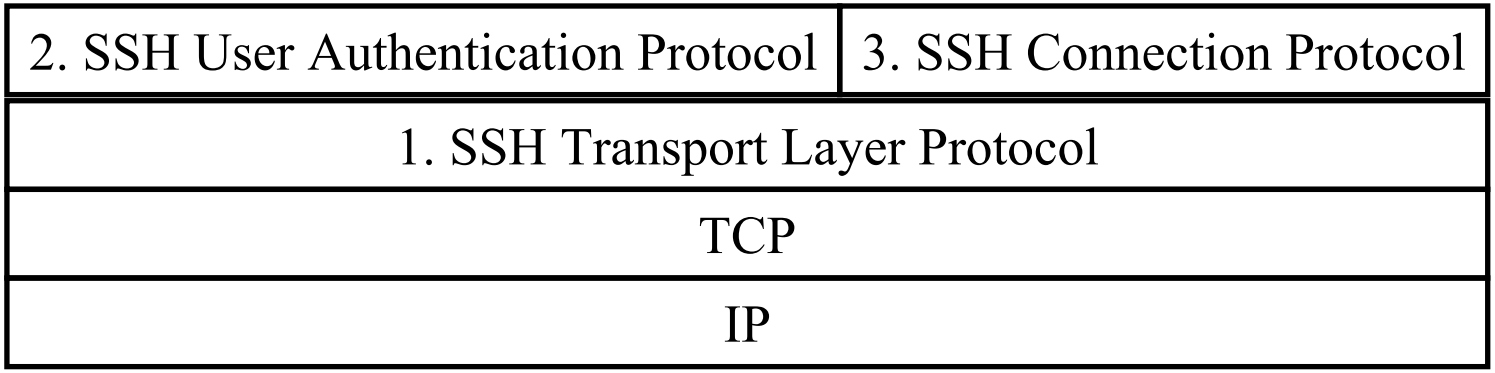
Secure Shell (SSH)

- Secure remote login
- 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
- Documented in RFCs 4250 through 4254
- SSH clients & servers are widely available

Ref: http://en.wikipedia.org/wiki/Secure_Shell

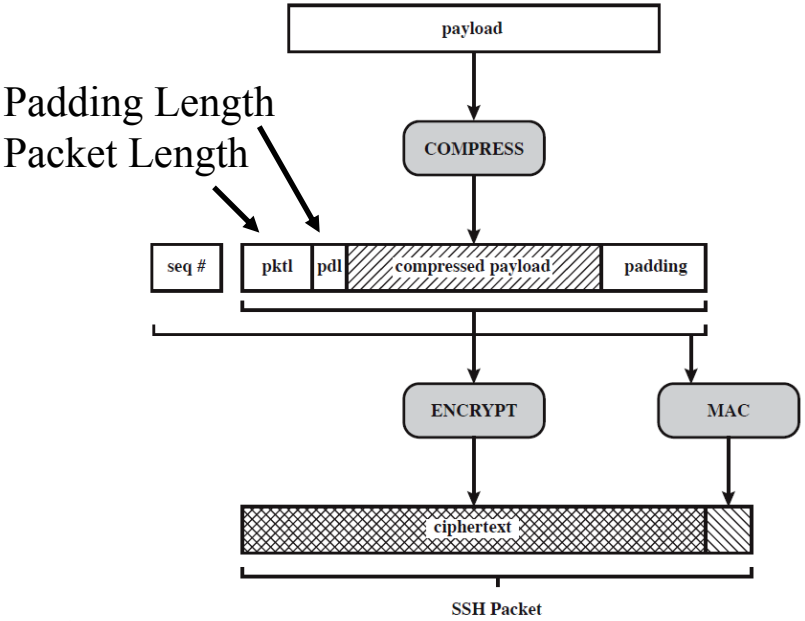
SSH Protocol Layers

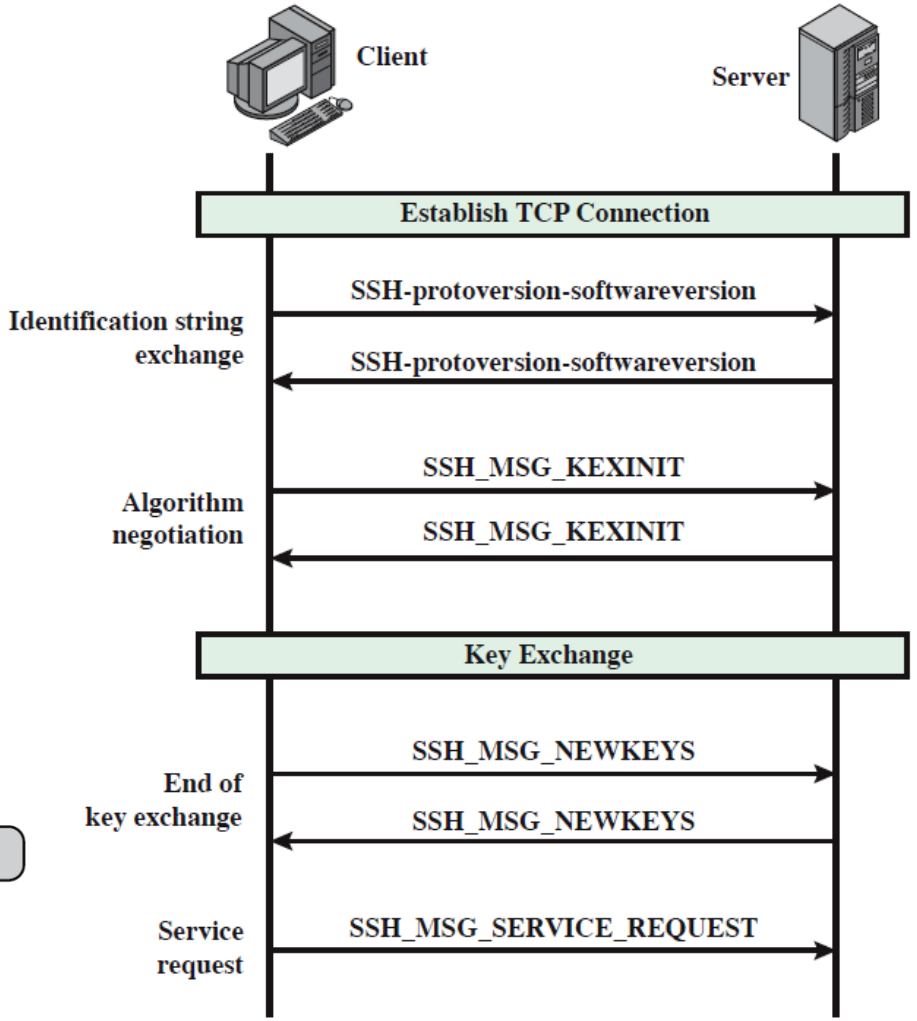
- **IP**: Routes messages to destination
- **TCP**: end-to-end reliable delivery
- **SSH Transport Layer Protocol**:
  - Server authentication, confidentiality, integrity
  - May optionally provide compression
- **SSH User Authentication Protocol**: Authenticates client
- **SSH Connection Protocol**: Provided multiple logical channels



SSH Transport Layer

- Server Authentication, Privacy and Integrity
- Client must know the servers public key in advance

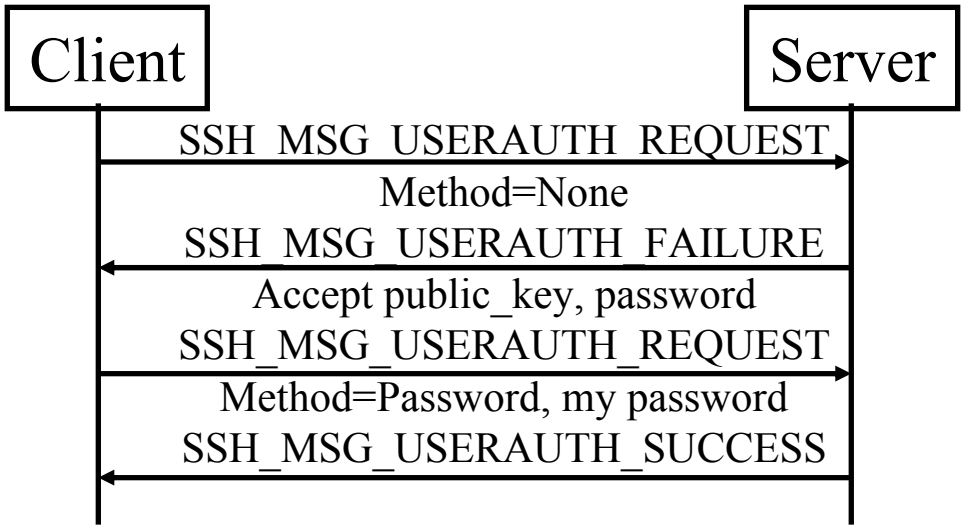




Ref: RFC 4253

SSH User Authentication Layer

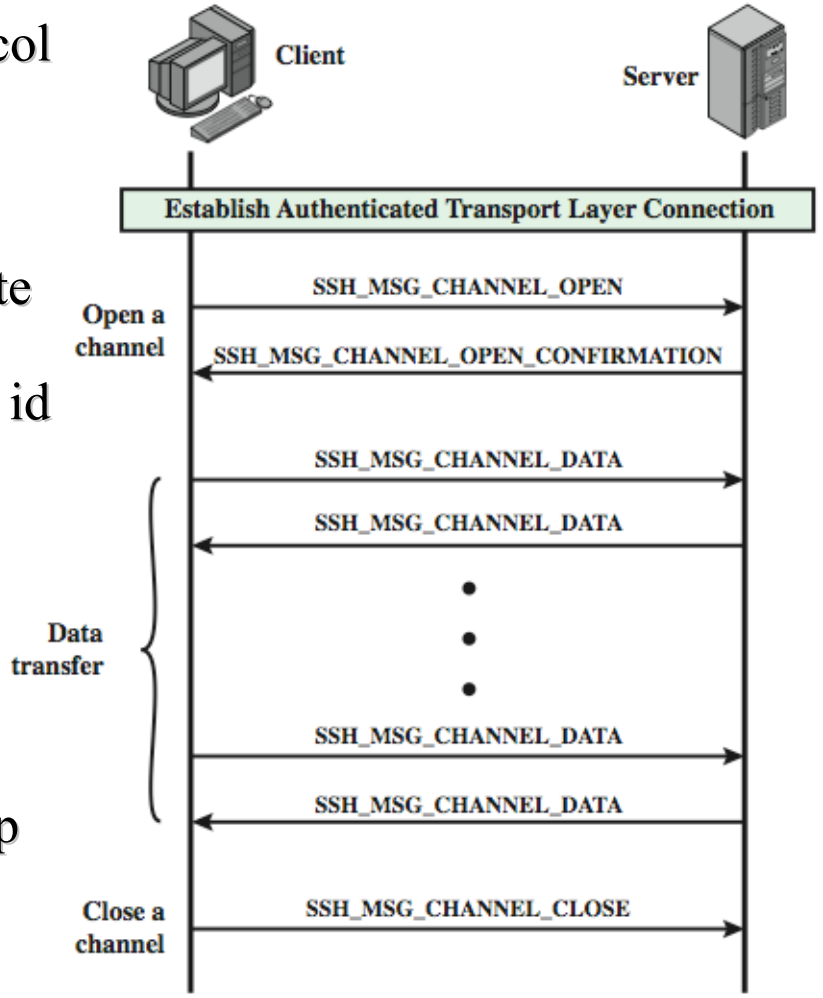
- Authenticates client to server
- Three message types:
  - SSH_MSG_USERAUTH_REQUEST
  - SSH_MSG_USERAUTH_FAILURE
  - SSH_MSG_USERAUTH_SUCCESS
- Authentication methods used: Public-key, password, host-based



Ref: RFC 4252

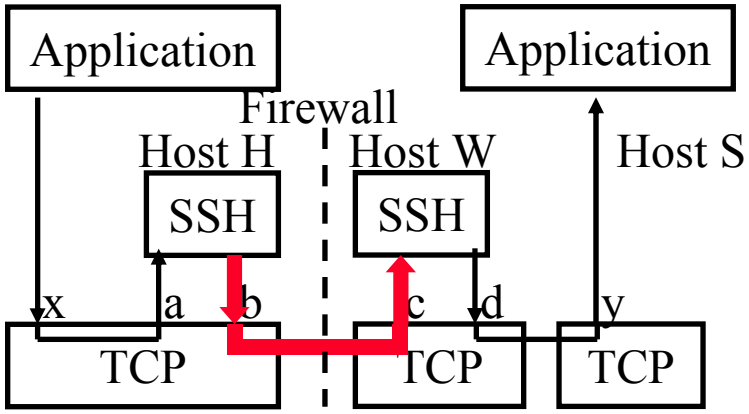
SSH Connection Layer

- Runs on SSH Transport Layer Protocol
- Assumes secure authentication connection
- Used for multiple logical channels
  - SSH communications use separate channels
  - Either side can open with unique id number
  - Flow controlled
  - Have three stages:
    - Opening a channel, data transfer, closing a channel
  - Four types:
    - Session, x11, forwarded-tcpip (remote port forwarding), direct-tcpip (local port forwarding).

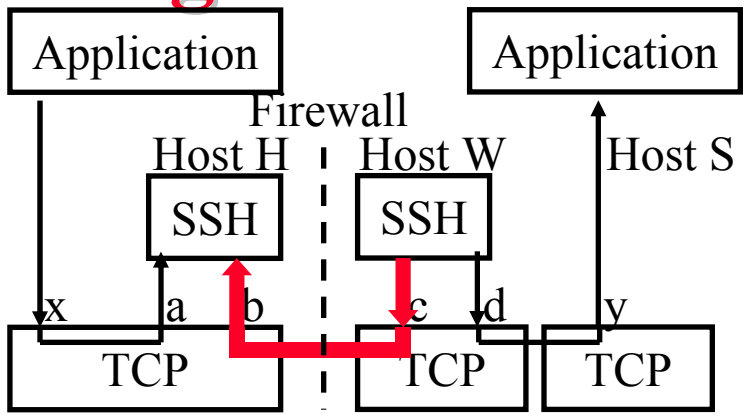


Ref: RFC 4254

Port Forwarding



(a) Local Forwarding
ssh –La:S:y W

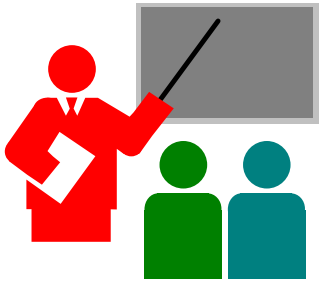


(a) Remote Forwarding
ssh –Ra:S:y H

- Port forwarding or tunneling allows insecure applications to run over secure SSH. SSH tells location application to connect to H:a rather than S:y. SSH listens to H:a, encrypts the traffic and sends to other side where SSH sends to S:y.
- Note: All TCP connections are bidirectional. Arrows show the TCP connect message direction. If application server is on W, "localhost" is used in place of S.
- Local forwarding: Client SSH (Host H) starts the tunnel, informs the server SSH (Host W): "Please forward the traffic on this *channel* to S:y"
- Remote Forwarding: Client SSH (Host W) starts the tunnel, informs the server SSH (Host H): "I will forward the traffic on this *channel* to S:y"

Ref: http://docstore.mik.ua/orelly/networking_2ndEd/ssh/ch09_02.htm

Summary



1. SSL provides security at transport layer. TLS is a standardization of SSL V3.
2. SSL consists of 4 protocols: Handshake (Crypto Negotiation), Change Cipher, Alert, and Record (Encryption and MAC)
3. HTTPS is simply http over SSL.
4. SSH provides secure remote login and consists of 3 protocols: User authentication, Connection (Channels), Transport layer (Encryption, MAC, Server authentication)
5. SSH port forwarding (tunneling) allows insecure applications to run in a secure mode.

Homework 17

Consider the following threats to Web security and describe how each is encountered by a particular feature of SSL.

A. **Brute-Force Cryptanalytic Attack**: An exhaustive search of the key space for a conventional encryption algorithm

B. **Know Plaintext Dictionary Attack**: Many messages will contain predictable plain text, such as the HTTP GET command. An attacker constructs a dictionary containing every possible encryption of the known-plaintext message. When an encrypted message is intercepted, the attacker takes the portion containing the encrypted known plaintext and looks up the ciphertext in the dictionary. The ciphertext should match against an entry that was encrypted wit the same secret key. If there are several matches, each of these can be tried against the full ciphertext to determine the right one. This attack is especially effective against small key sizes (e.g., 40-bite keys).

C. **Replay Attack**: Earlier SSL handshake messages are replayed.

D. **Man in the middle Attack**: AN attacker interposes during key exchange, active as the client to the server and as the server to the client.

E. **Password Sniffing**: Passwords in HTT or other application traffic are eaves dropped.

F. **IP Spoofing**: Uses forced IP addresses to fool a host into accepting bogus data.

G. **IP Hijacking**: An active, authenticated connection between two hosts is disrupted and the attacker takes the place of one of the hosts.

H. **SYN Flooding**: An attacker sends TCP SYN messages to request a connection but does not respond to the final message to establish the connection fully. The attacked TCP module typically leaves the "half-open connection" around for a few minutes. Repeated SYN messages can clog the TCP module.