UNIK4250 Security in Distributed Systems
University of Oslo
Spring 2012

Part 5

Transport Layer Security

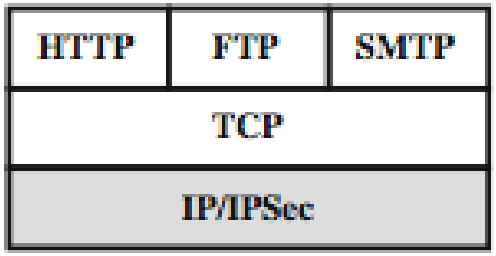
Outline

- SSL/TLS transport layer security protocols
- HTTPS
- Secure Shell (SSH)

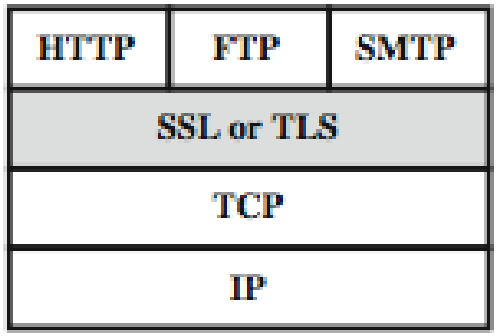
Web Security

- Web now widely used by business, government, individuals
- but Internet & Web are vulnerable
- have a variety of threats
  - integrity
  - confidentiality
  - availability
  - authentication
- need added security mechanisms

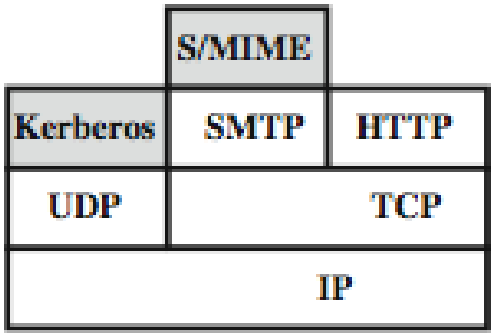
Web Traffic Security Approaches



(a) Network Level



(b) Transport Level

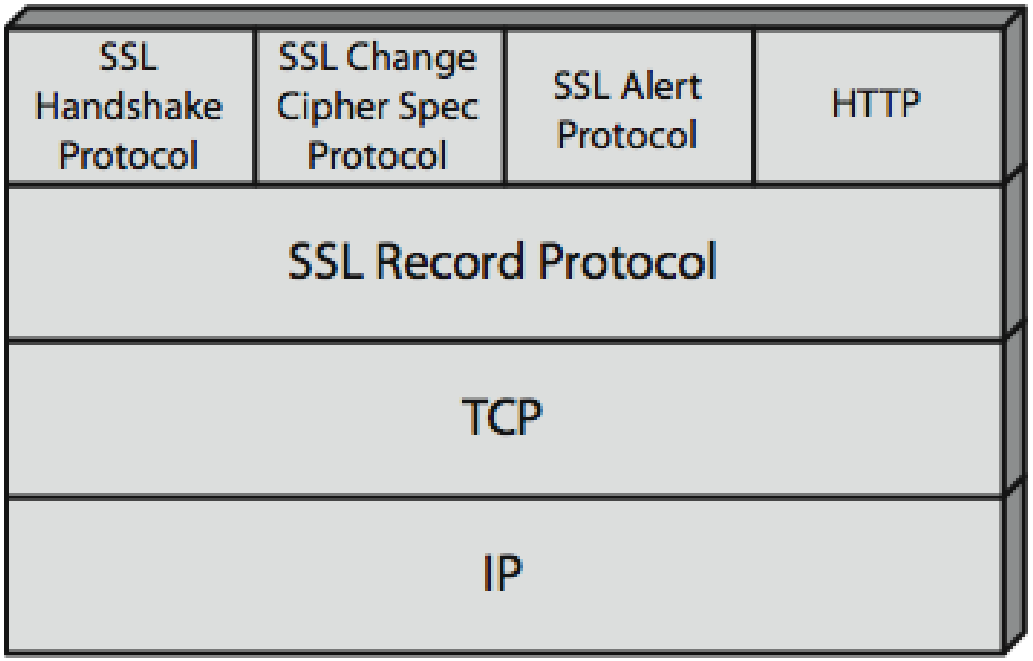


(c) Application Level

SSL (Secure Socket Layer)

- Transport layer security service
- Originally developed by Netscape
- Version 3 designed with public input
- 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



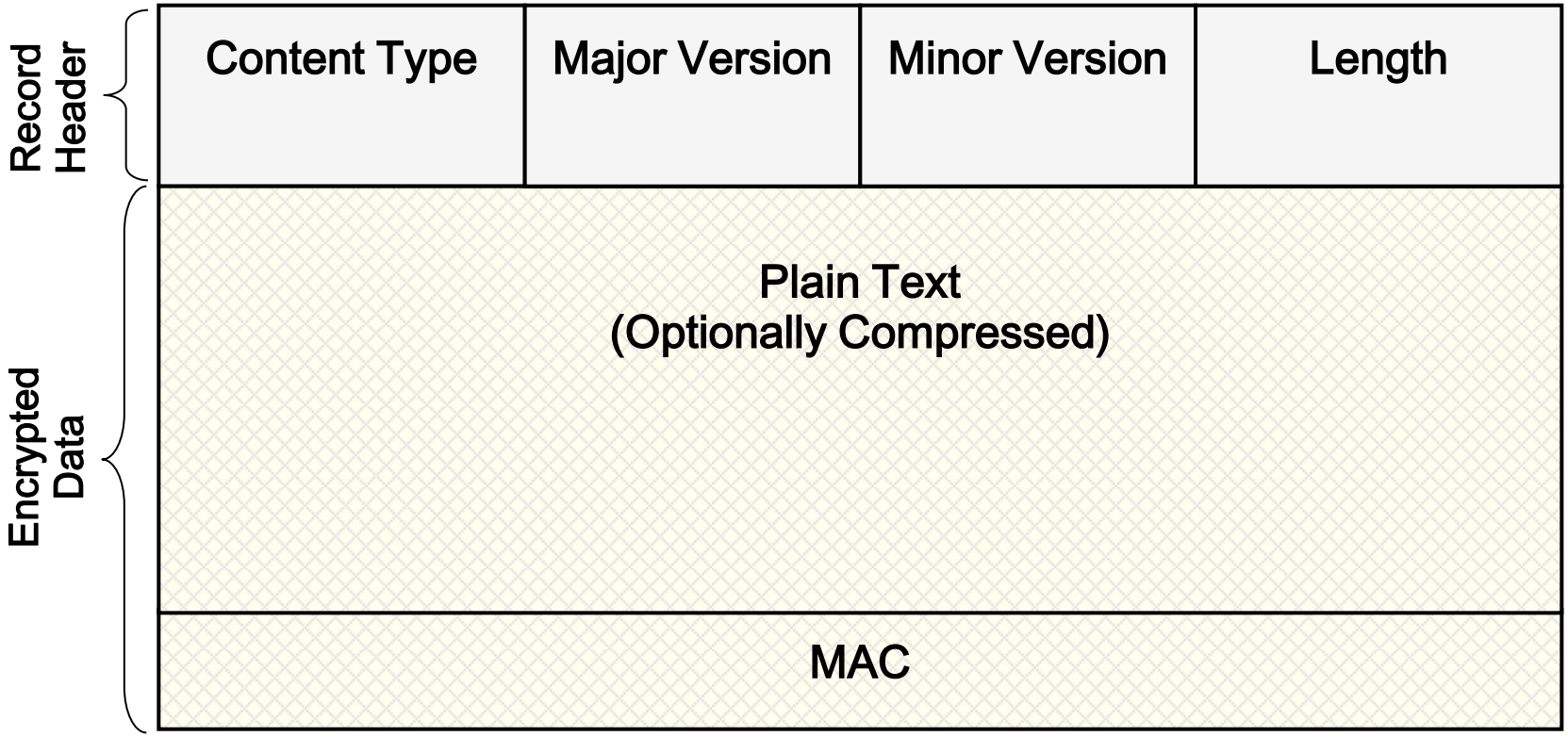
SSL Architecture

- **SSL connection**
  - a transient, peer-to-peer, communications link
  - associated with 1 SSL session
- **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

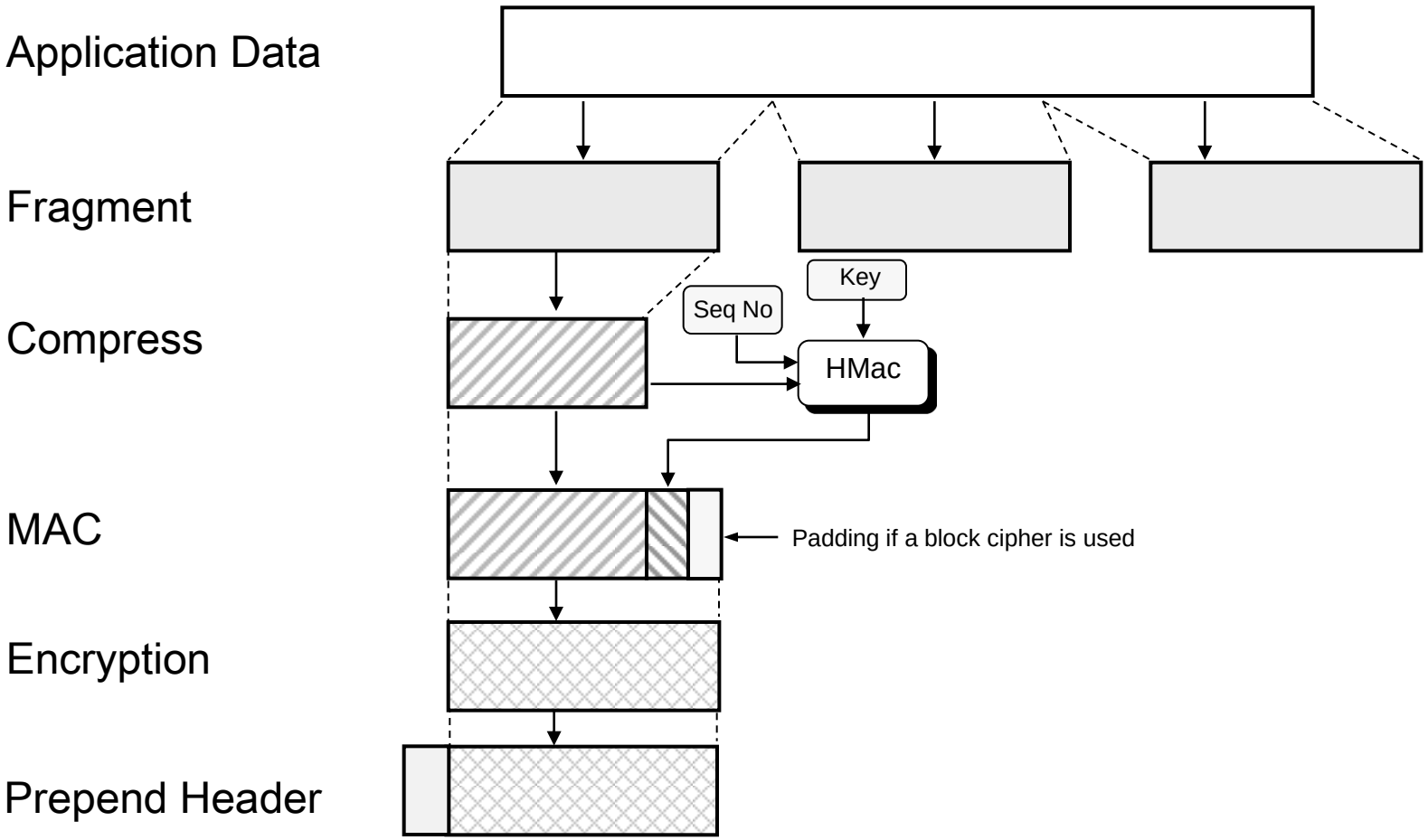
SSL Record Protocol Services

- **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

TLS:
Record Format



TLS:
Record Protocol Operation



TLS:
Record Protocol Operation

- Fragmentation:
  - Each application layer message is fragmented into blocks of 214 bytes or less.
- Compression:
  - Optionally applied.
  - SSL v3 & TLS – default compression algorithm is null
- Add MAC:
  - Calculate a MAC over the compressed data using a MAC secret from the connection state.
  - The algorithm used is based on the HMAC as defined in RFC 2104.

TLS:
Record Protocol Operation

- Encrypt:
  - The compressed data plus MAC are encrypted using a symmetric cipher.
  - Permitted ciphers include AES, IDEA,DES, 3DES, RC4
  - For block ciphers, padding is applied after the MAC to make a multiple of the cipher's block size.

TLS:
Record Protocol Operation

- Prepend TLS Record Header containing:
  - Content Type
  - Protocol Version:

Major Version	Minor Version	Version Type
3	0	SSLv3
3	1	TLS 1.0
3	2	TLS 1.1
3	3	TLS 1.2

  - Length: length in octets of the data
  - Defined content types are:
    - change_cipher_spec
    - alert
    - handshake
    - application_data

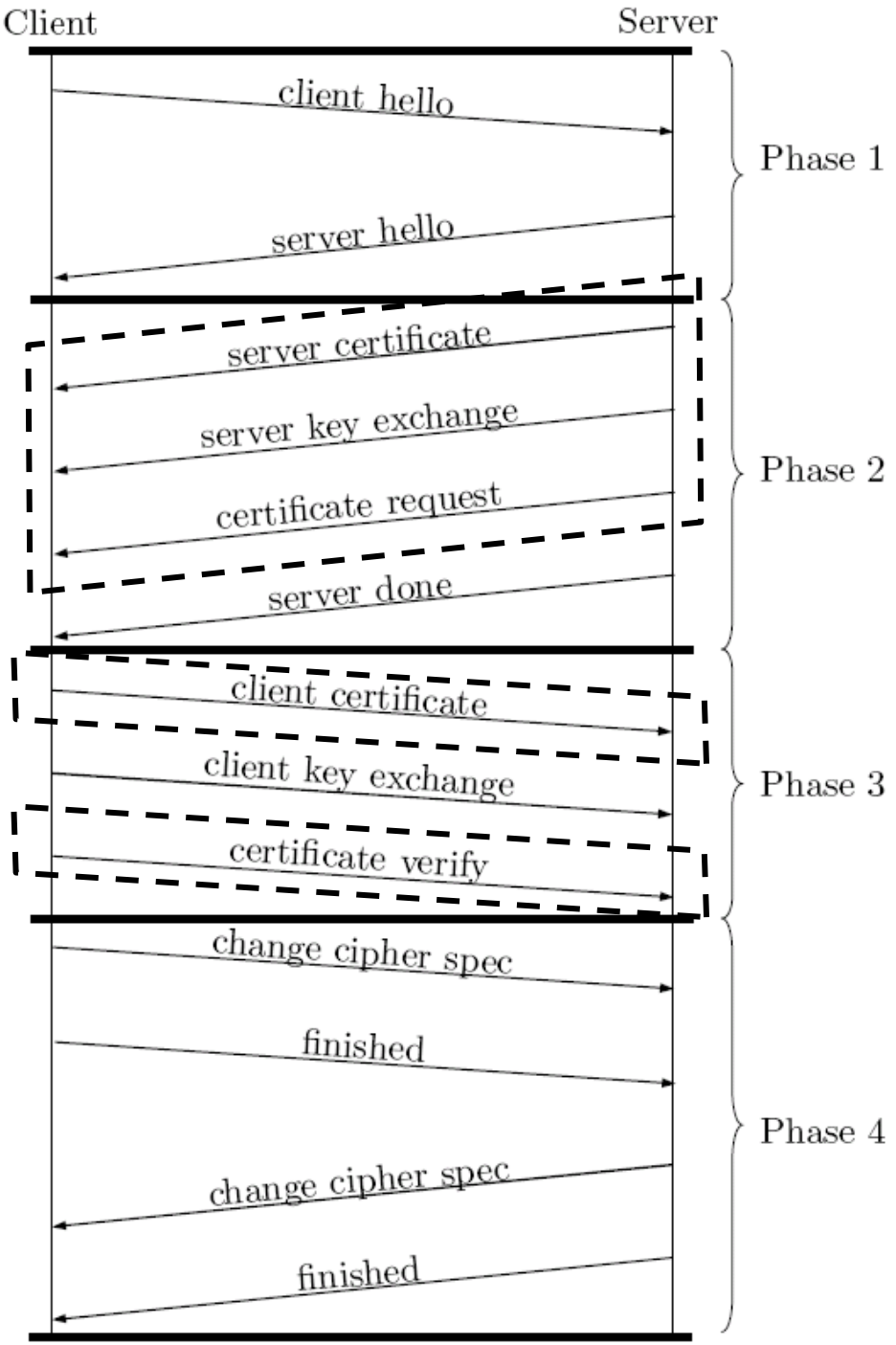
TLS: Handshake Protocol

- The handshake protocol
  - Negotiates the encryption to be used
  - Establishes a shared session key
  - Server authentication and key exchange
  - Client authentication and key exchange
  - Completes the session establishment
- After the handshake application data is transmitted securely
- Several variations of the handshake exist
  - RSA variants
  - Diffie-Hellman variants

TLS: Handshake Four phases

- Phase 1: Initiates the logical connection and establishes its security capabilities
- Phases 2 and 3: Performs key exchange. The messages and message content used in this phase depends on the handshake variant negotiated in phase 1.
- Phase 4: Completes the setting up of a secure connection.

Optional

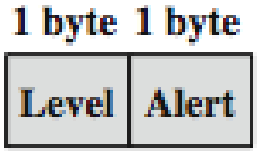


SSL ChangeCipherSpecProtocol

- A single message, sent by both client and server
- Notifies to other party that the just negotiated cipher suite (pending state) becomes current
  - triggers the encryption
- Sent after Handshake protocol
- Cipher suite can be renegotiated,
  - change to new cipher suite is triggered by ChangeCipherSpecProtocol

(a) Change Cipher Spec Protocol

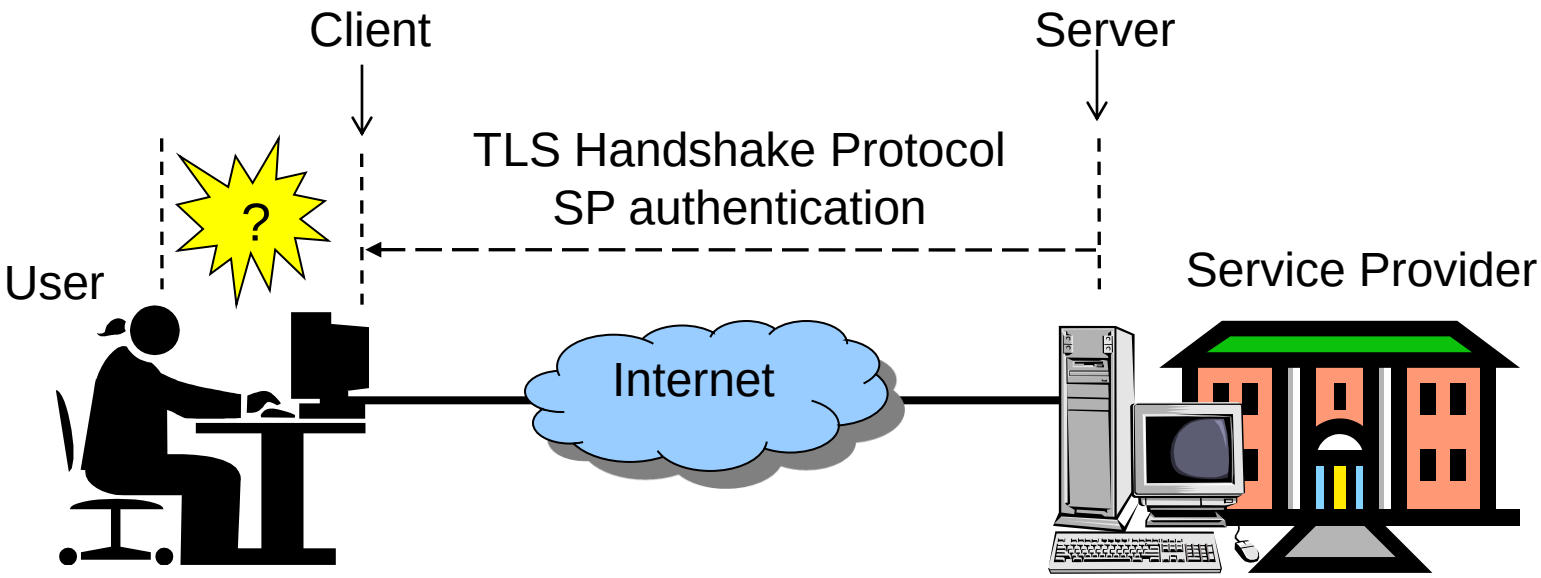
SSL Alert Protocol



(b) Alert Protocol

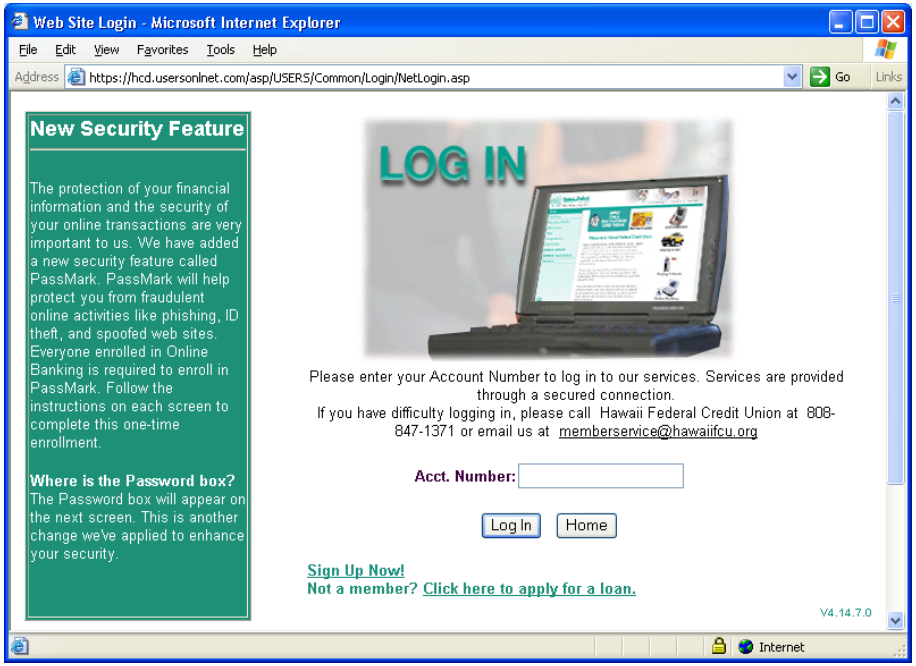
- Conveys SSL-related alerts to peer entity
- Level
  - Warning or Fatal
- Alerts:
  - fatal: unexpected message, bad record mac, decompression failure, handshake failure, illegal parameter
  - warning: close notify, no certificate, bad certificate, unsupported certificate, certificate revoked, certificate expired, certificate unknown
- Compressed & encrypted like all SSL data

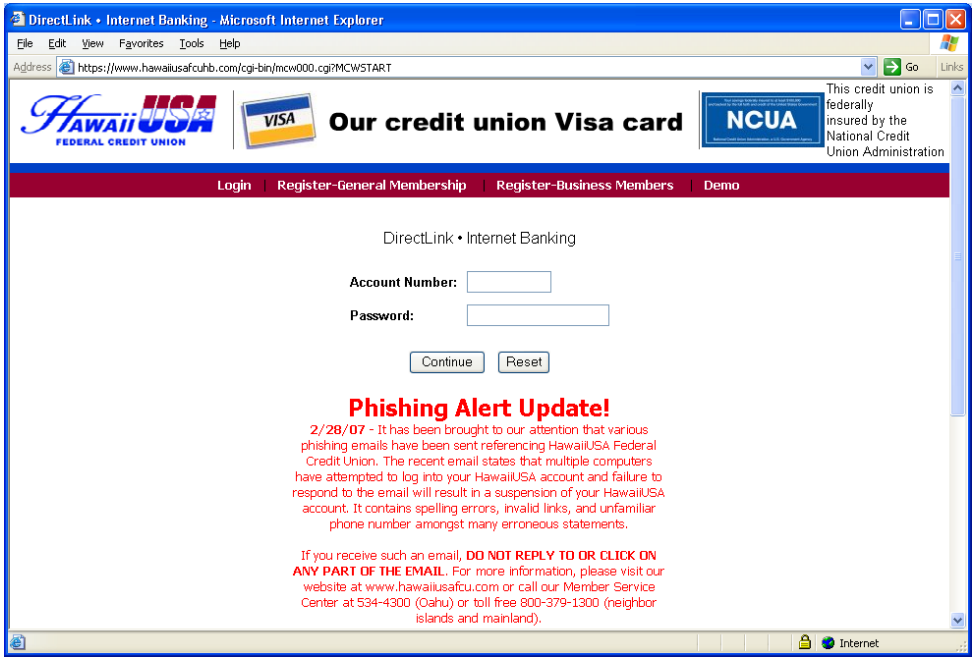
Meaningless server authentication



- Client assumes that the URL fed into browser is the server name that the user intends to contact.
- Client validates certificate name against URL without verifying that it is the intended name.
- The URL is often not the intended name

A phishing example
Hawaii Federal Credit Union





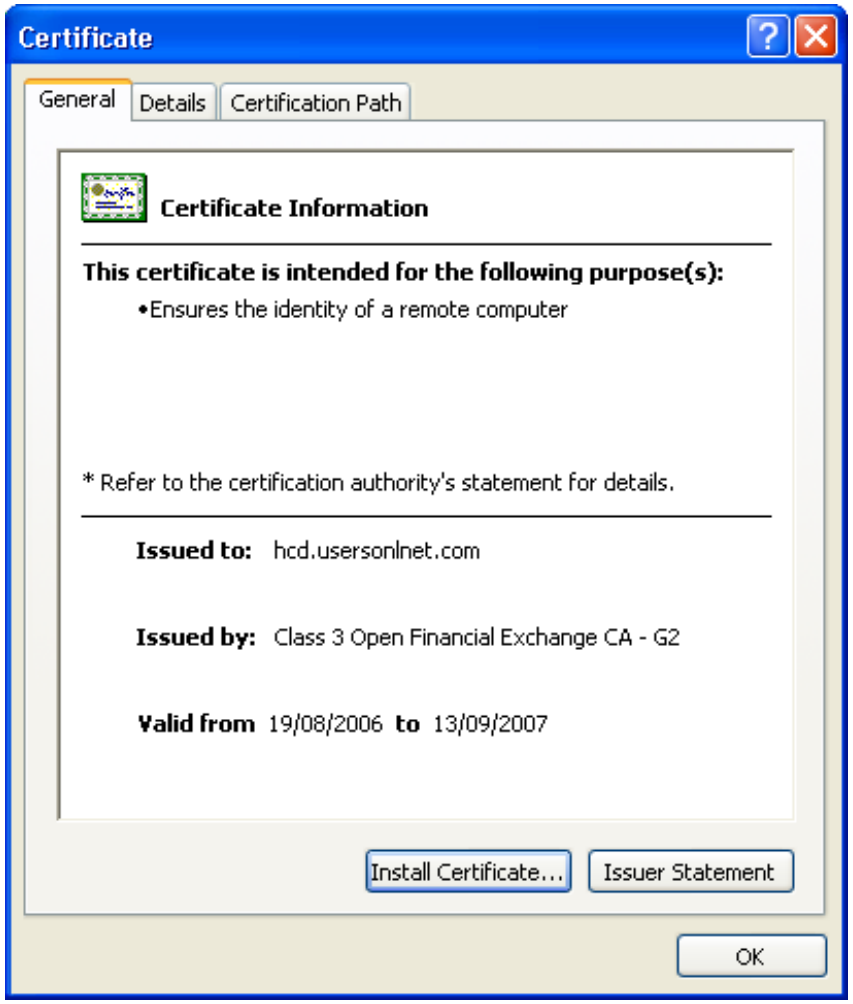
Genuine bank login

https://hcd.usersonlnet.com/asp/USERS/Common/Login/NettLogin.asp

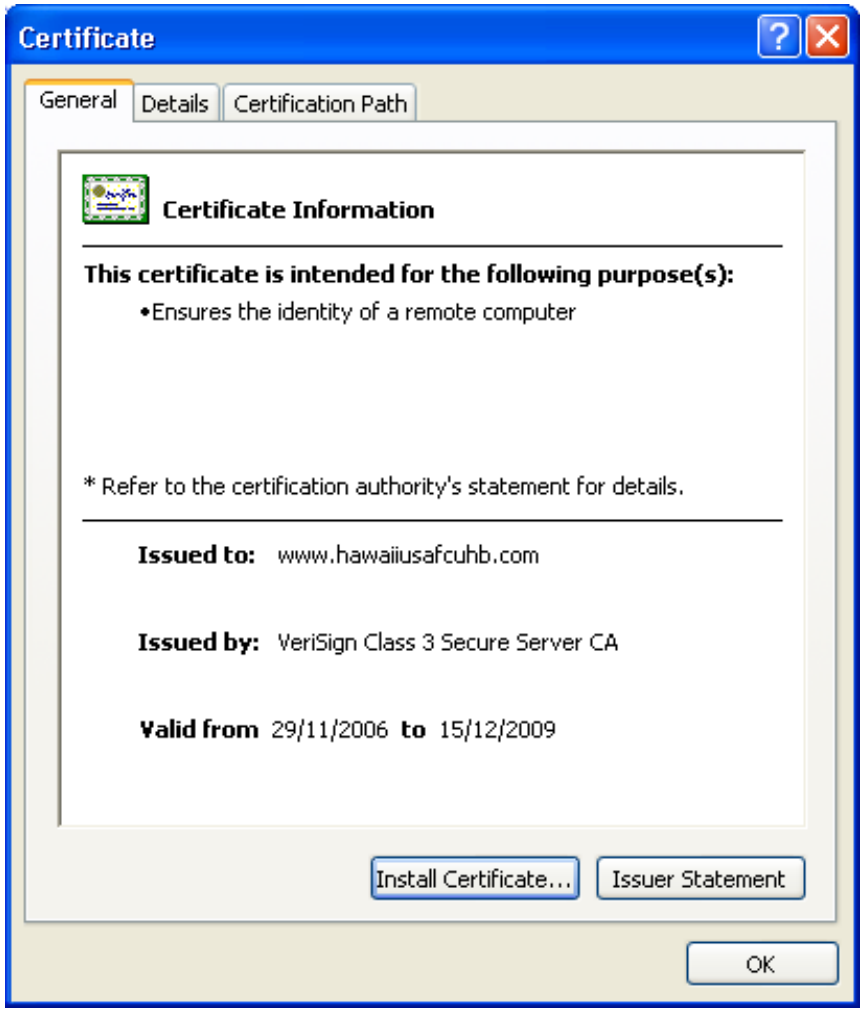
Fake bank login

https://hawaiiusafcuhb.com/cgi-bin/mcw00.cgi?MCWSTART

Certificate comparison 1

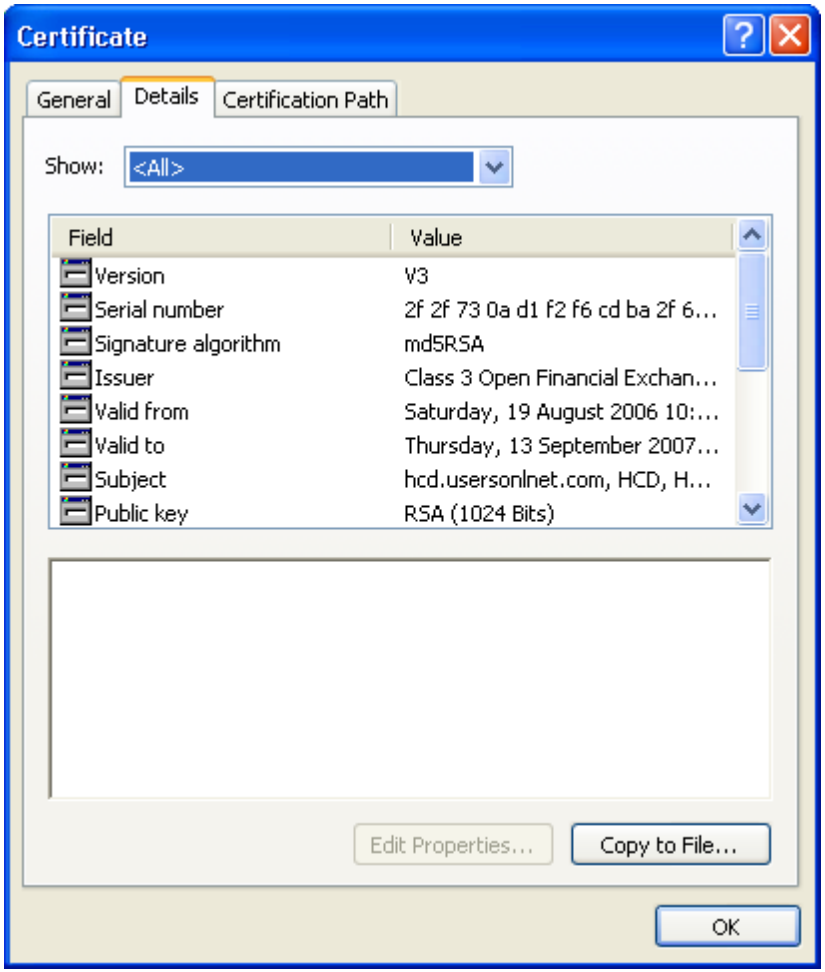


Genuine certificate

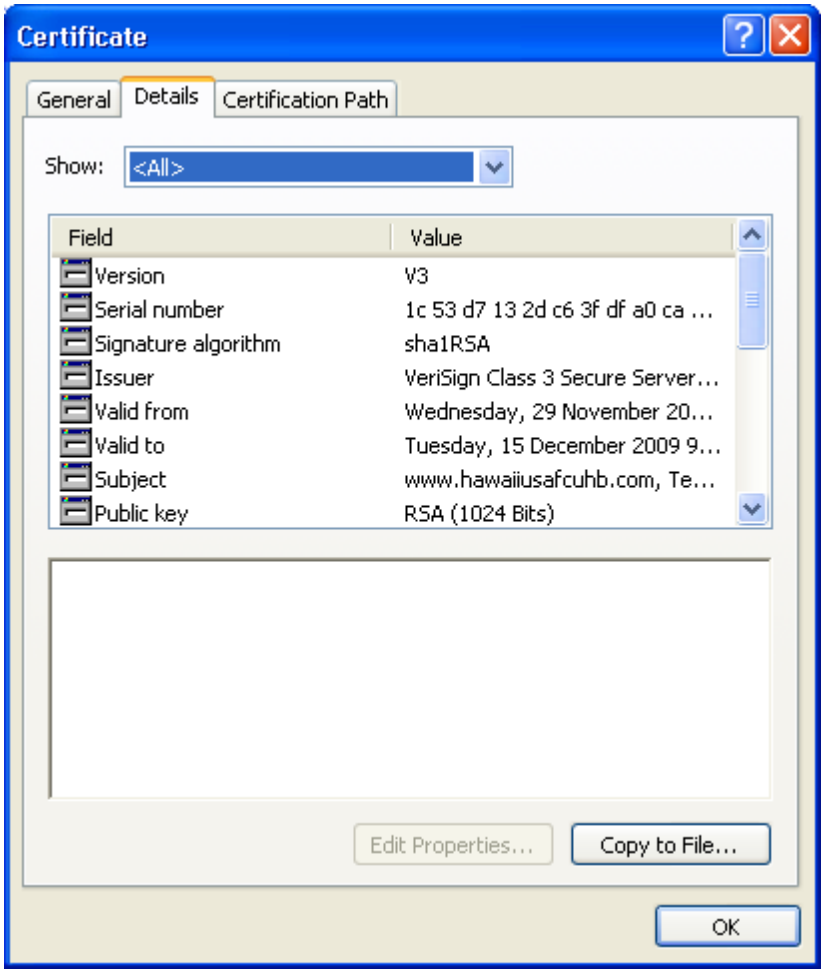


Fake certificate

Certificate comparison 2

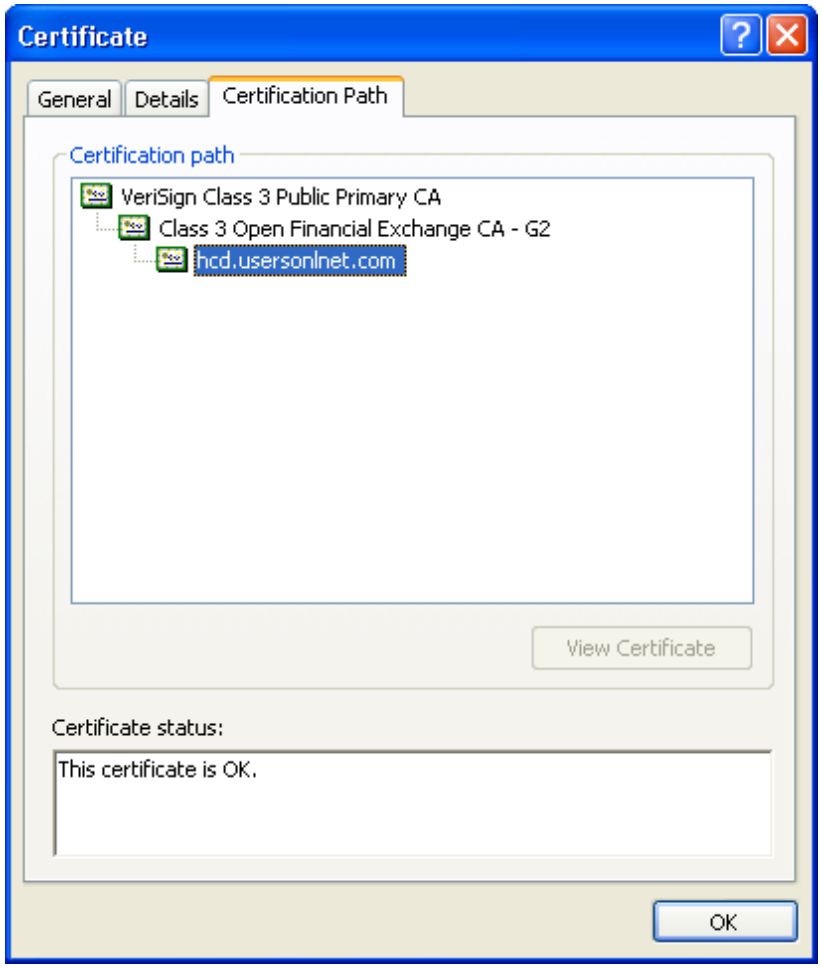


Genuine certificate

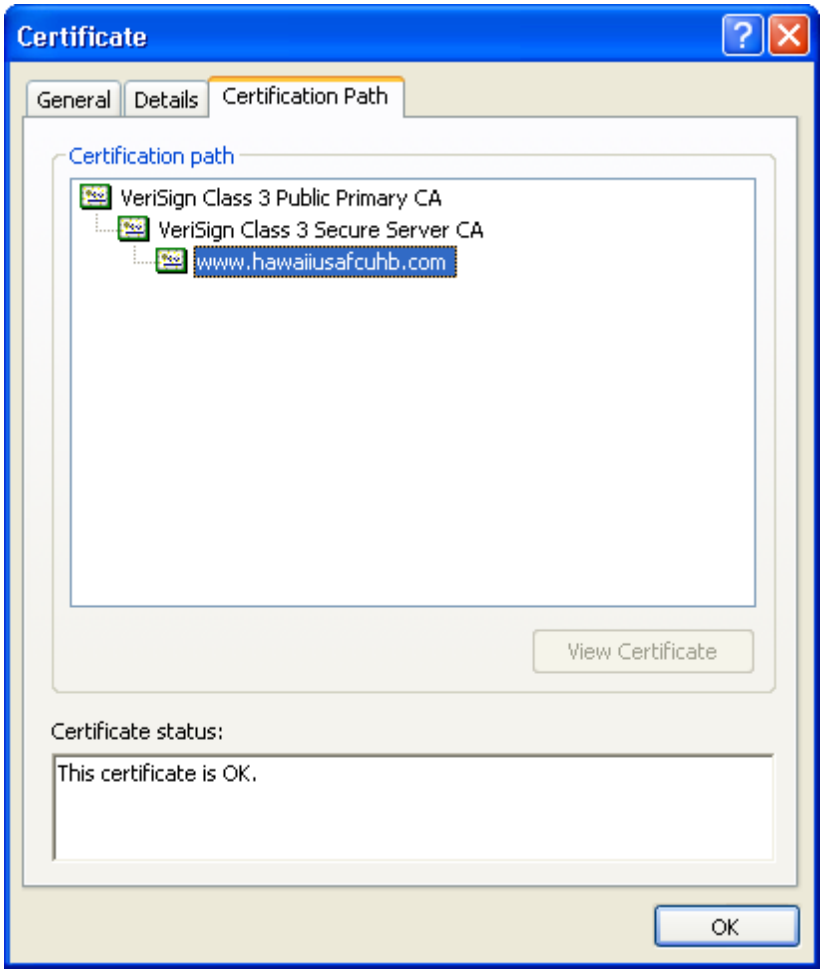


Fake certificate

Certificate comparison 3

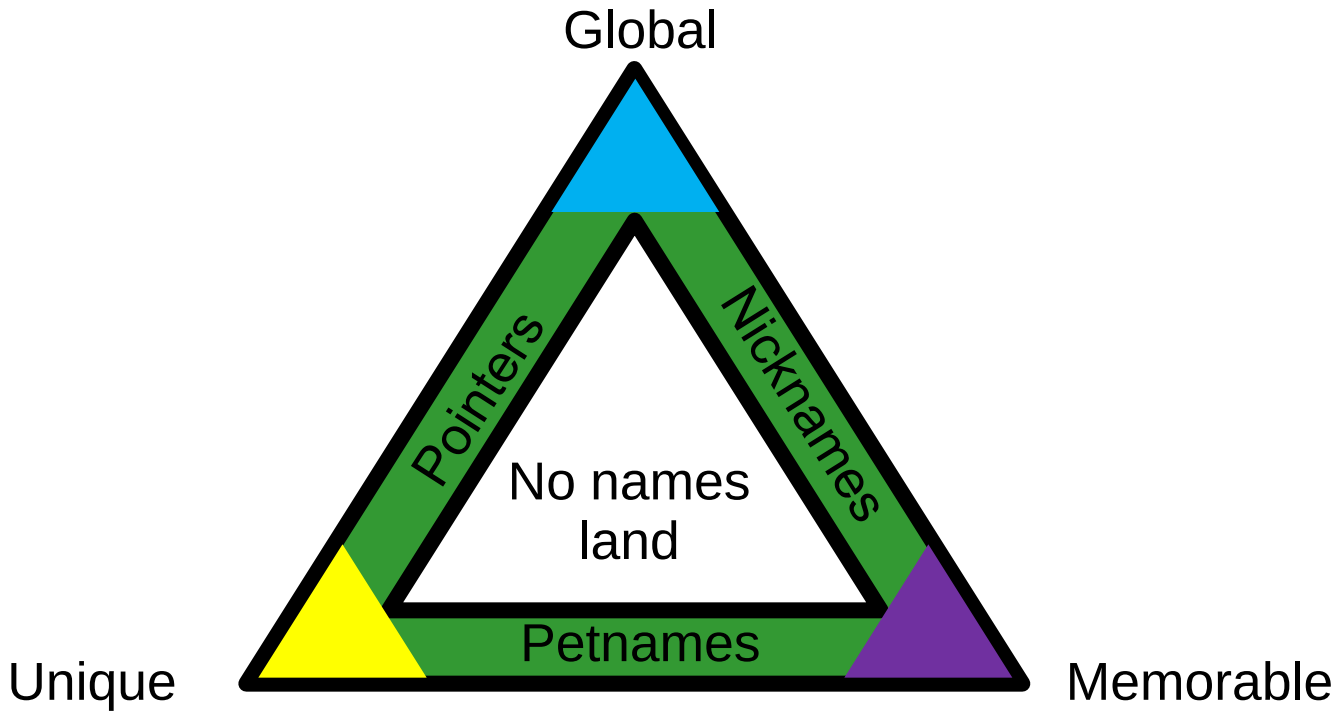


Genuine certificate



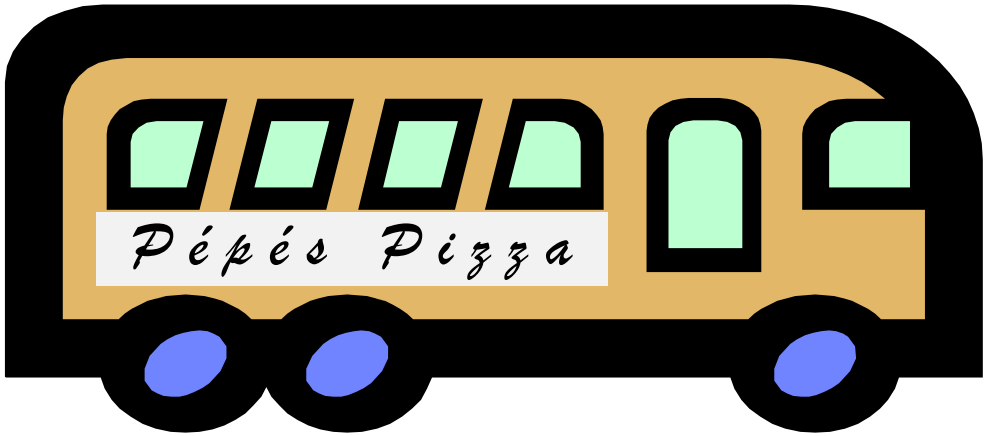
Fake certificate

Zooko's Triangle of Id Properties



- No name can be at the same time global, unique and memorable
- Names can only have 2 of the 3 properties at the same time

Passing bus test for memorability



- If you see a name written on a passing bus, and you can remember the name after 5 minutes, then the name is memorable
- Petnames, which are unique & memorable, must be mapped to global & unique names to make TLS server authentication meaningful.

How lazy can we get?

Communication with n entities normally requires **secure distribution** of keys.

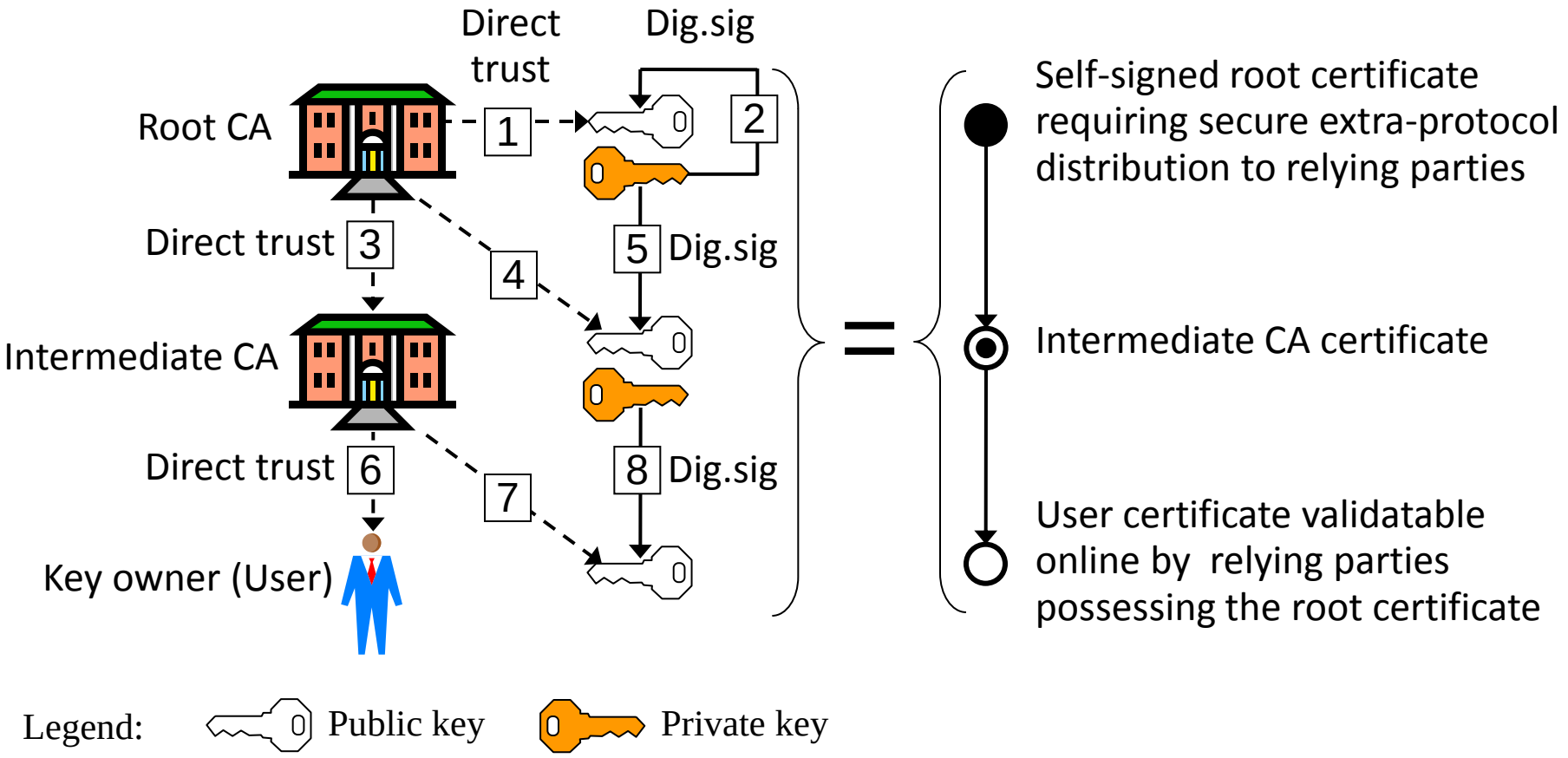
- $n(n-1)/2$ symmetric keys
  - Every pair must exchange secret key
  - Secure, but too hard !
- n asymmetric root keys of PKI
  - Every entity must receive root public key
  - Secure, but also too hard !
- **0** keys: Send root keys insecurely online
  - Insecure distribution of root certificates is easy, so that's what we do, ...
  - but assurance is weak !

Self-signed root keys: Why?

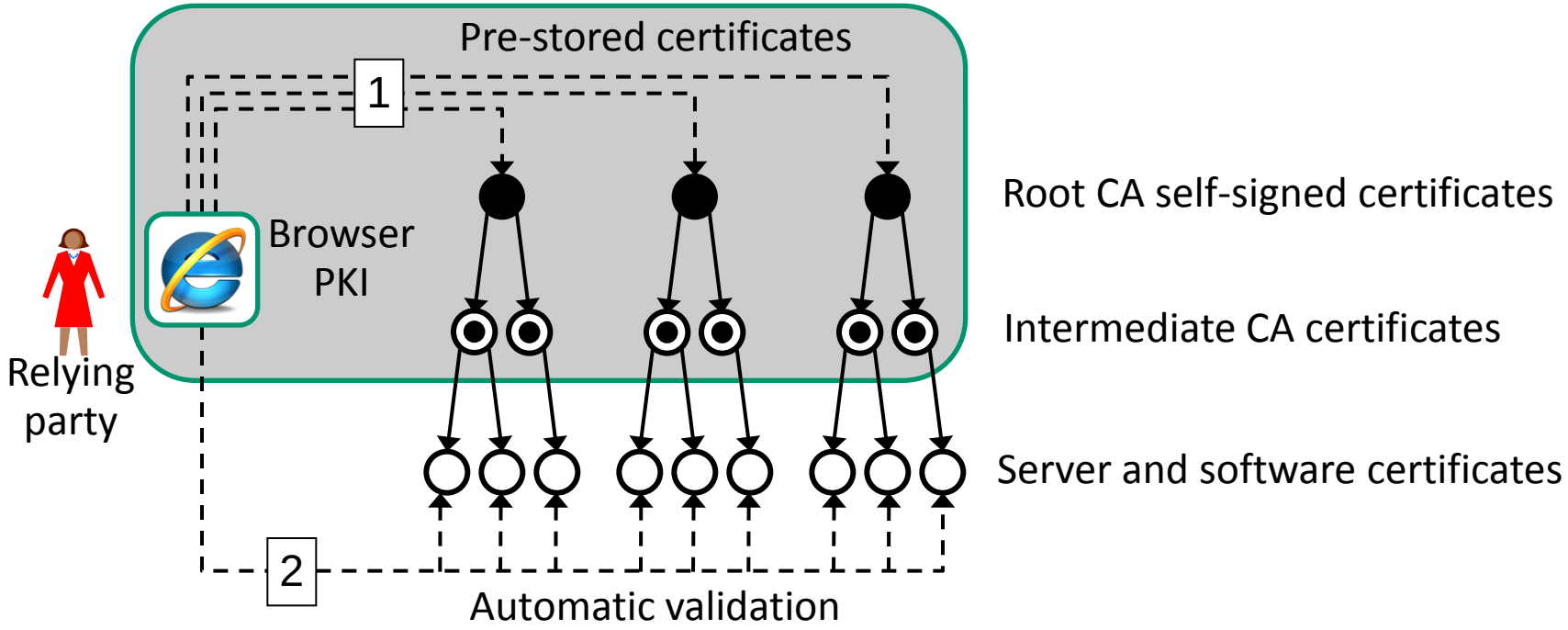
- Most people think a root public key is authentic just because it is self-signed
- Not a coincidence
  - Gives impression of assurance
  - Disguises insecure practice

- Self-signing has absolutely no useful purpose
  - indicates that somebody holds private key, but so what?

Certificate signature chain



Certificate validation in Browser PKI



Assurance(B-PKI) = min [Assurance(PKI_k)]
Assurance(PKI_k) = min [Assurance(CA_i)]

Each CA is a possible weakest link

Public-key certificate meaning

- Public-key certificates are only about identity, not about honesty & reliability normally associate with trust
- Stuxnet worm was considered advanced because it was signed under a valid software certificate
- Why were people surprised?
- Anybody can buy software certificates and sign whatever they want, even the Mafia !!!

Anonymous Diffie-Hellmann
TLS option that provides confidentiality
(provides no authentication)

Alice picks random integer a

$g^a \bmod p$

$g^b \bmod p$

Bob picks random integer b

Alice computes the shared secret

$g^{ab} \bmod p = (g^b)^a \bmod p$

Bob computes the same shared secret

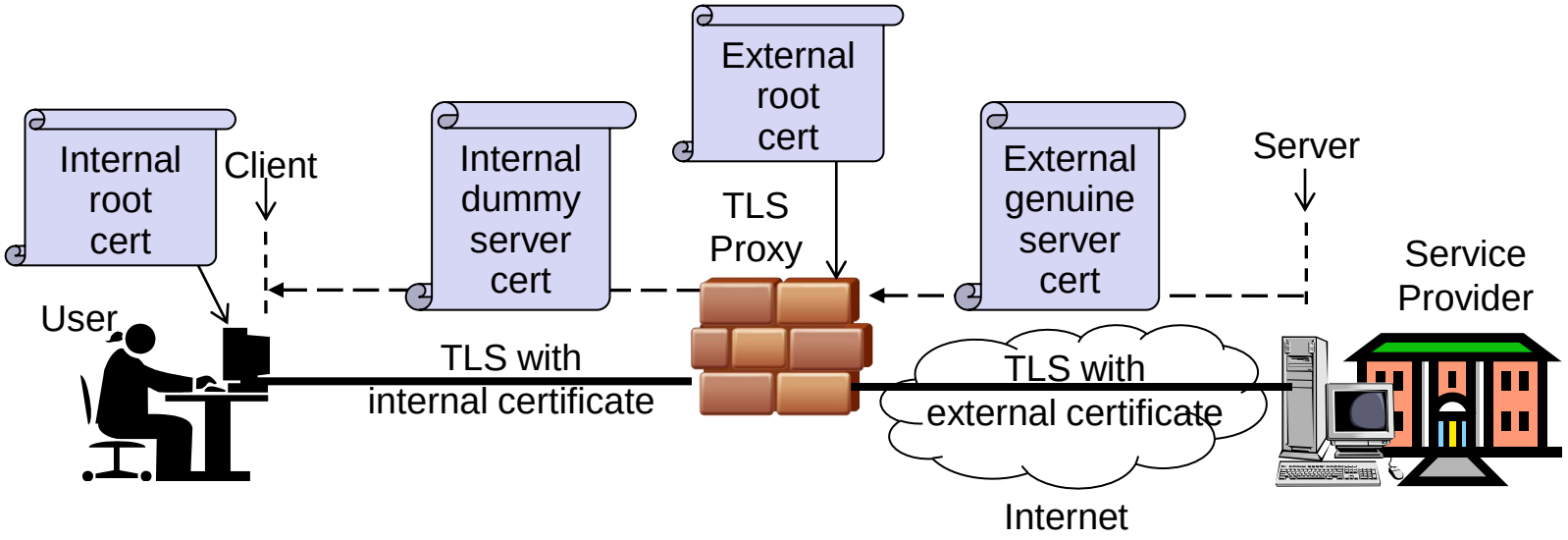
$g^{ab} \bmod p = (g^a)^b \bmod p$.

TLS doesn't need certificates

- TLS encryption possible by using ADH (Anonymous Diffie-Hellman) profile
- No certificate needed
- Why is nobody using TLS ADH profile ???
- TLS-ADH described as vulnerable to MitM
  - What can go wrong?
  - Very difficult to spoof IP addresses
  - Network based MitM attack would be difficult
- TLS-RSA meaningless as long as domain names are not reliably recognized
  - Vulnerable to client based MitB attack

TLS proxy

- Organisations that require inspection of TLS traffic must split the TLS connection in two.
- The internal TLS session uses a ”dummy” certificate that looks like the genuine external certificate.
- External and internal certificates have identical names
- Internal ”dummy” certificate is signed by internal root



TLS Cryptographic Computations

- Master secret creation
  - a one-time 48-byte value
  - generated using secure key exchange (RSA / Diffie-Hellman) and then hashing info
- 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

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

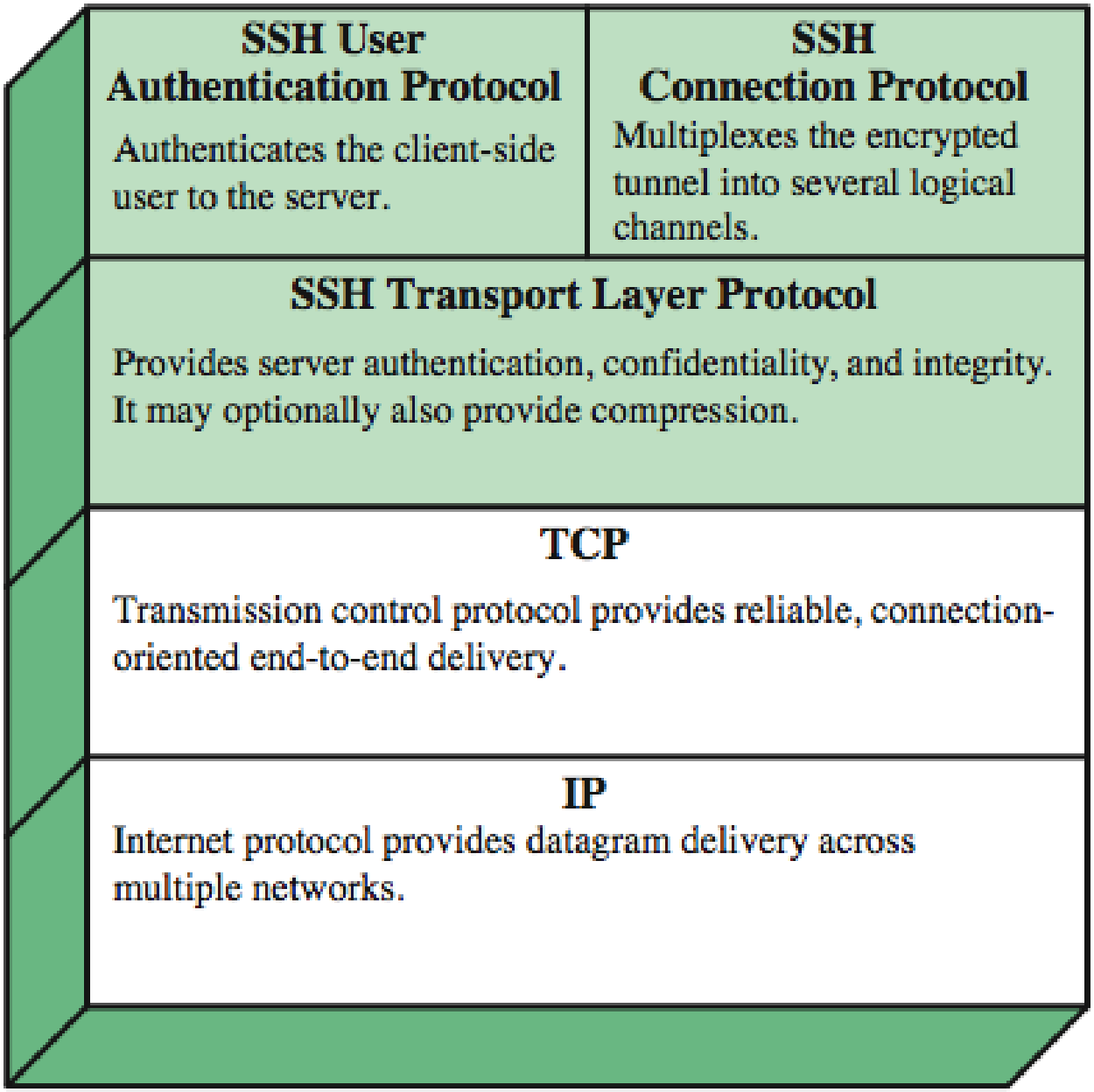
HTTPS Use

- 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 TCP close before alert exchange sent or completed

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
- Documented in RFCs 4250 through 4254
- SSH clients & servers are widely available
- Method of choice for remote login/ X tunnels

SSH Protocol Stack



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
  - warning if no key stored, then the key (certificate) can be imported on user approval (insecure channel)
- Packet exchange
  - establish TCP connection
  - can then exchange data
    - identification string exchange, algorithm negotiation, key exchange, end of key exchange, service request
  - using specified packet format

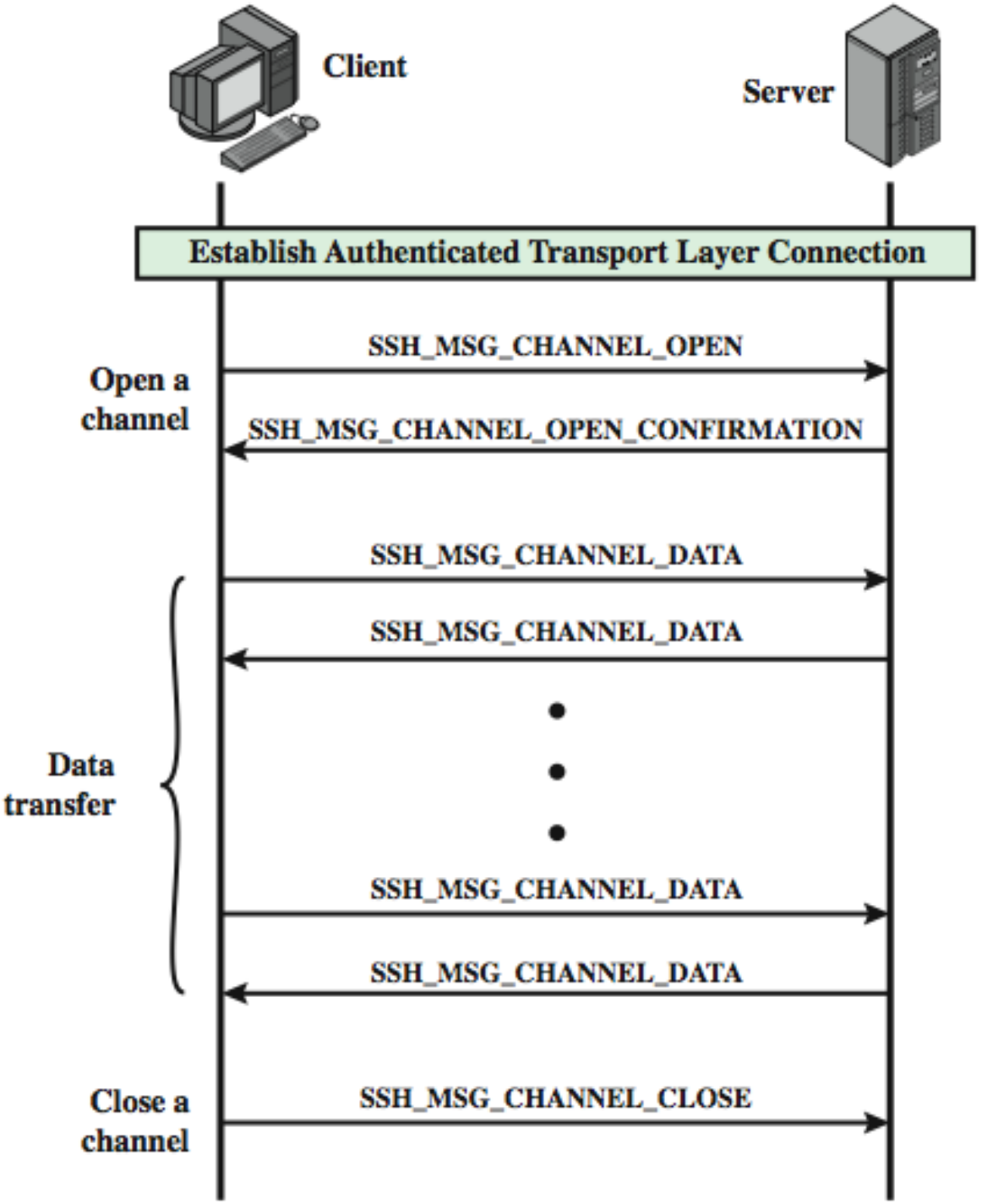
SSH User Authentication Protocol

- 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

SSH Connection Protocol

- 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, direct-tcpip.

SSH Connection Protocol Exchange



Port Forwarding

- 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
  - remote forwarding – client acts for server

Summary

- Topics studied in this lecture:
  - The need for web security
  - SSL/TLS transport layer security protocols
  - Semantic limitation of TLS authentication
  - HTTPS
  - SSH (Secure Shell)