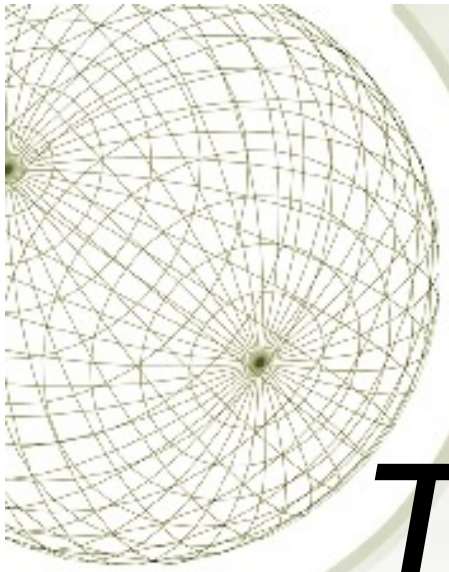
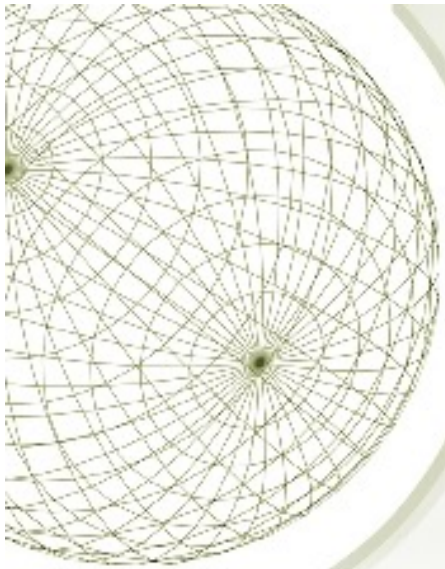


Transport Level Security

Ola Flygt
Linnaeus University, Sweden
http://homepage.lnu.se/staff/oflmsi/
Ola.Flygt@lnu.se



Outline

- Web Security Considerations
- Secure Socket Layer (SSL) and Transport Layer Security (TLS)
- HTTPS and HSTS
- HTTP/3
- SSH

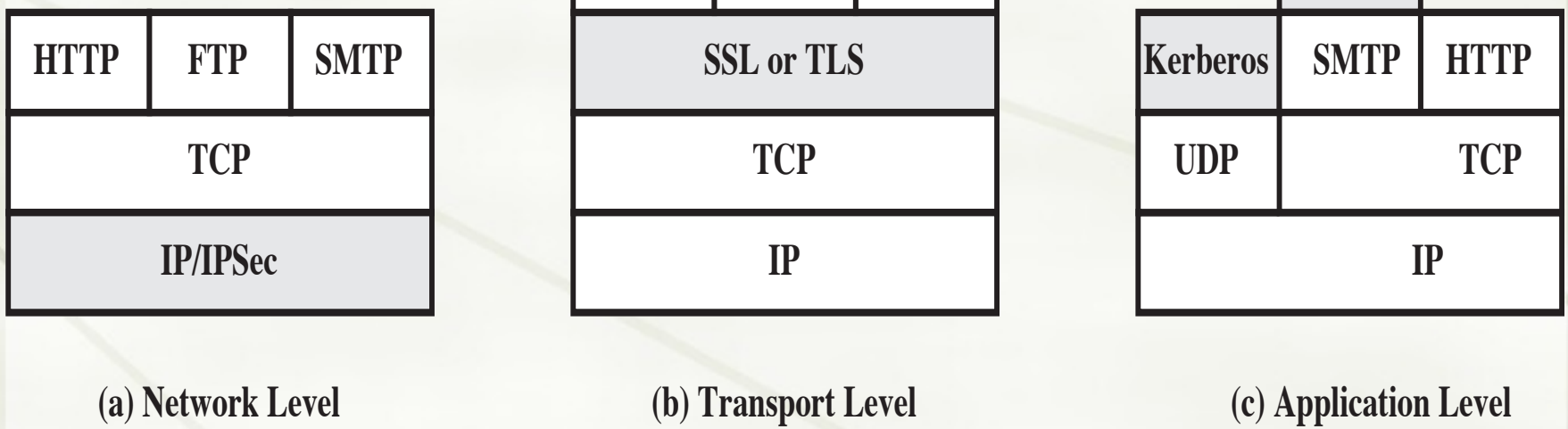
Web Security Considerations

- The WEB is very visible
- Complex software hide many security flaws
- Web servers are easy to setup, but not in a secure way
- Users are not aware of the risks

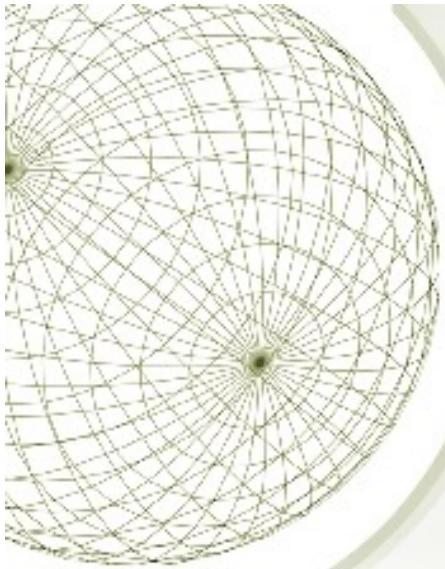
Threats on the Web

	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

Security facilities in the TCP/IP protocol stack



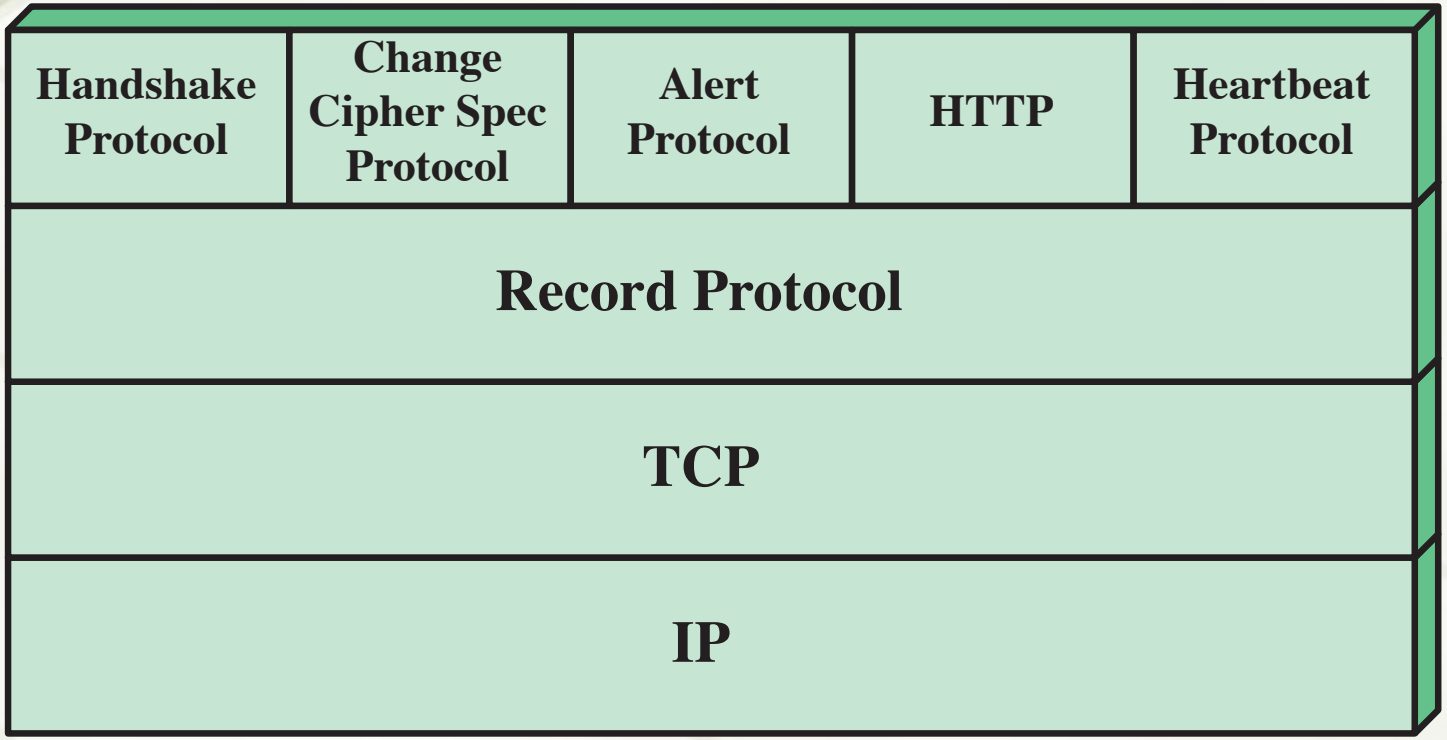
(a) Network Level (b) Transport Level (c) Application Level

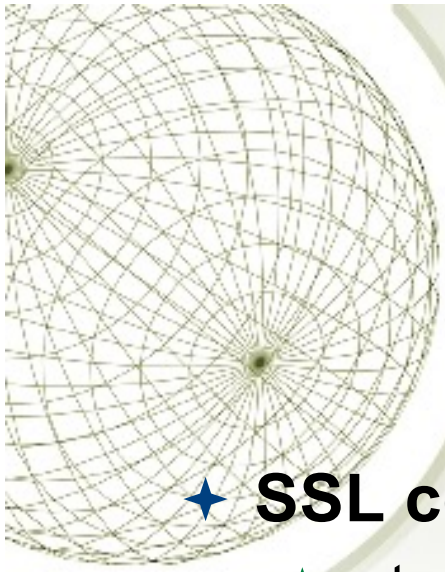


SSL and TLS

- SSL was originally created by Netscape, specifically to secure web traffic
- TLS working group was formed within IETF
- First version of TLS can be viewed as an SSLv3.1

SSL/TLS Protocol Stack





SSL Architecture

- **SSL connection**
  - a transient, peer-to-peer, communications link
  - associated with one 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
  - Reduces the cost of negotiating security parameters for each connection

A session state is defined by the following parameters:

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

Server write MAC secret

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

Client write MAC secret

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

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

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

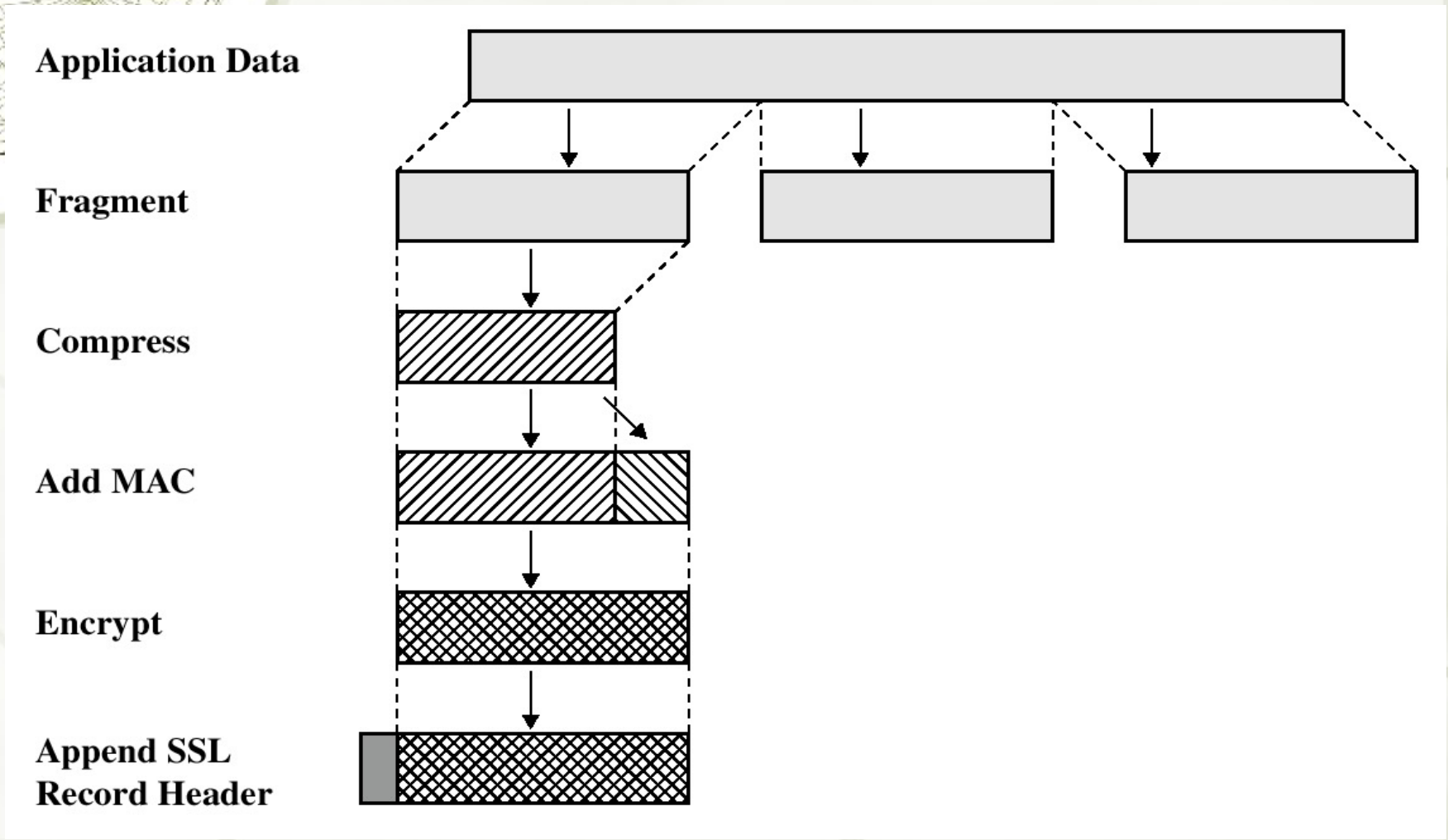
Sequence numbers

- 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 $2^{64} - 1$

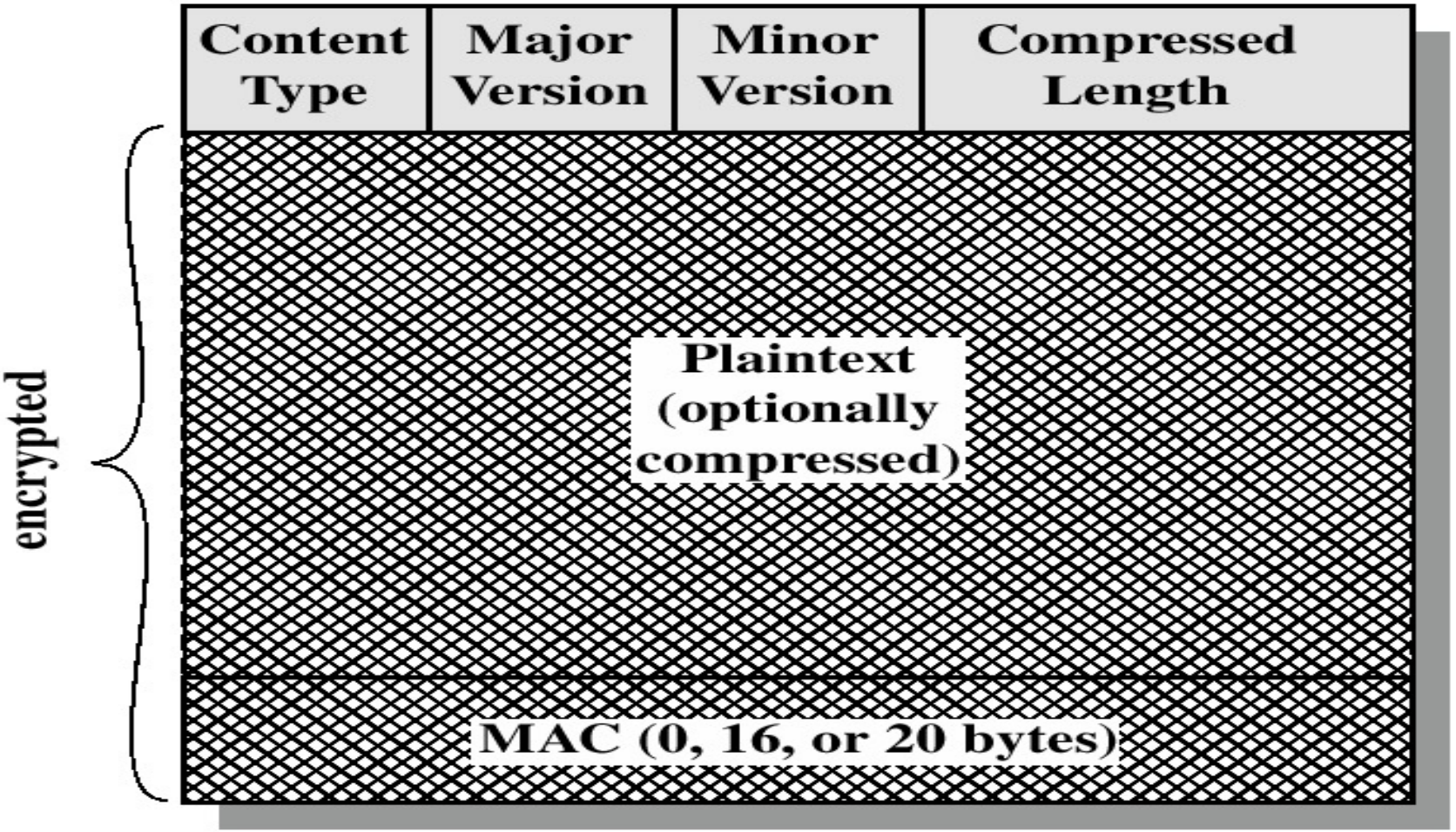
SSL Record Protocol Services

- **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

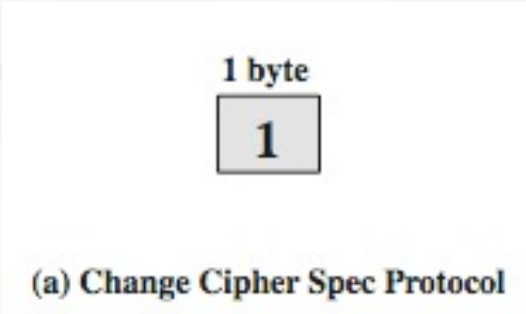


SSL Record Format



SSL Change Cipher Spec Protocol

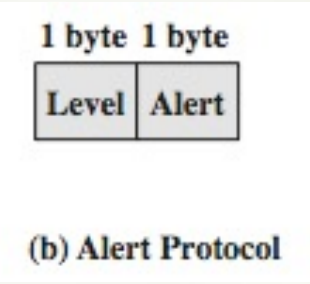
- one of 3 SSL specific protocols which uses the SSL Record protocol
- a single message
- causes pending state to become current
- hence updating the cipher suite in use



(a) Change Cipher Spec Protocol

SSL Alert Protocol

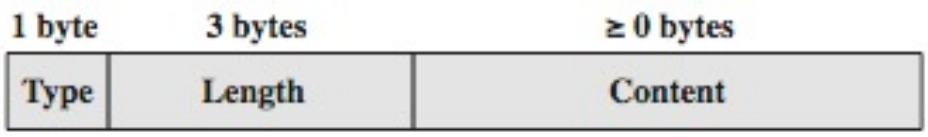
- conveys SSL-related alerts to peer entity
- severity
  - warning or fatal
- specific alert
  - fatal: unexpected message, bad record mac, decompression failure, handshake failure, illegal parameter
  - warning: close notify, no certificate, bad certificate, unsupported certificate, certificate revoked, certificate expired, certificate unknown
- compressed & encrypted like all SSL data

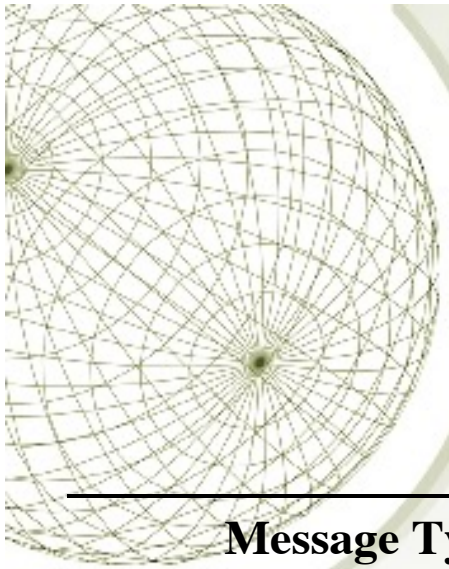


(b) Alert Protocol

SSL Handshake Protocol

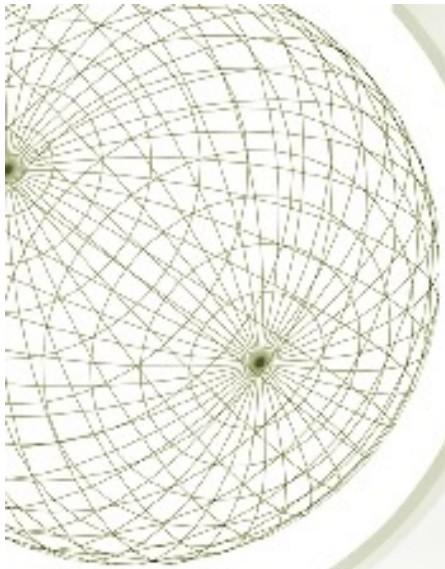
- allows server & client to:
  - authenticate each other
  - to negotiate encryption & MAC algorithms
  - to negotiate cryptographic keys to be used
- comprises a series of messages in phases
  1. Establish Security Capabilities
  2. Server Authentication and Key Exchange
  3. Client Authentication and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



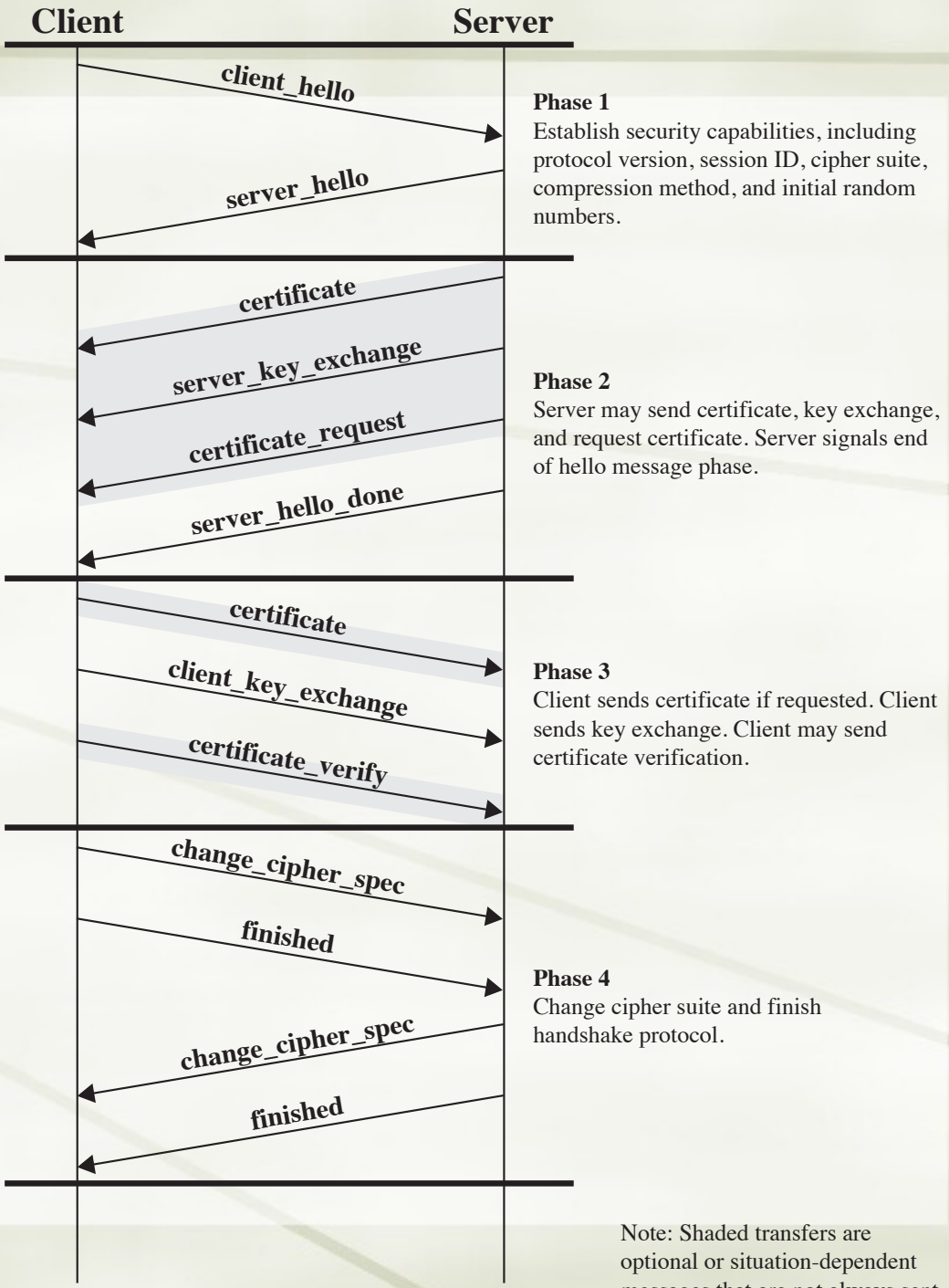


SSL Handshake Protocol Message Types

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



SSL Handshake Protocol



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.

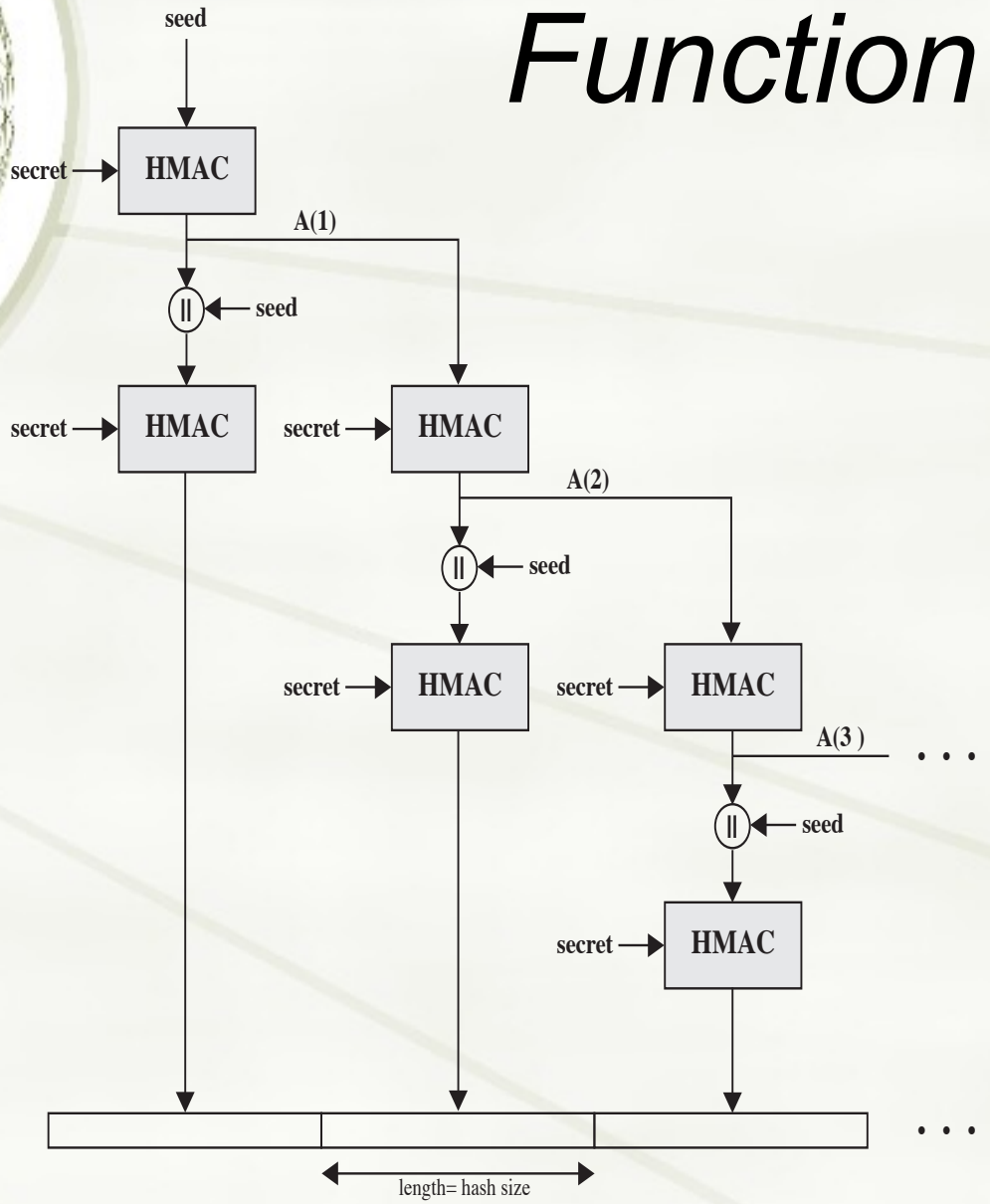
Phase 4
Change cipher suite and finish handshake protocol.

Note: Shaded transfers are optional or situation-dependent messages that are not always sent.

Cryptographic Computations

- 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 the master secret

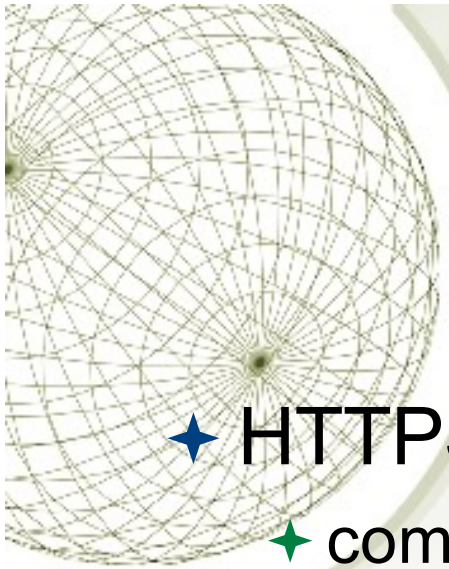
Function P_hash



The output of the function give the client write MAC secret, the server write MAC secret etc.

Transport Layer Security (RFC 2246)

- The same record format as the SSL record format and very similar to SSLv3.
- Differences in the protocols:
  - message authentication code (now HMAC)
  - pseudorandom function (to expand the shared secret)
  - Additional alert codes
  - cipher suites (e.g. Fortezza removed)
  - client certificate types (to sign some parameters)
  - Certificate verify and finished message (minor changes)
  - cryptographic computations (calculation of master secret)
  - Padding
  - More changes in 1.3 removing support for insecure methods
- TLS version number 1.3 is current version (from August 2018)



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

Adoption of HTTPS

Website protocol support (August 2021)

Protocol version	Website support	Security
SSL 2.0	0.4%	Insecure
SSL 3.0	3.2%	Insecure
TLS 1.0	44.6%	Deprecated
TLS 1.1	48.9%	Deprecated
TLS 1.2	99.5%	Depends on cipher and client mitigations
TLS 1.3	47.8%	Secure

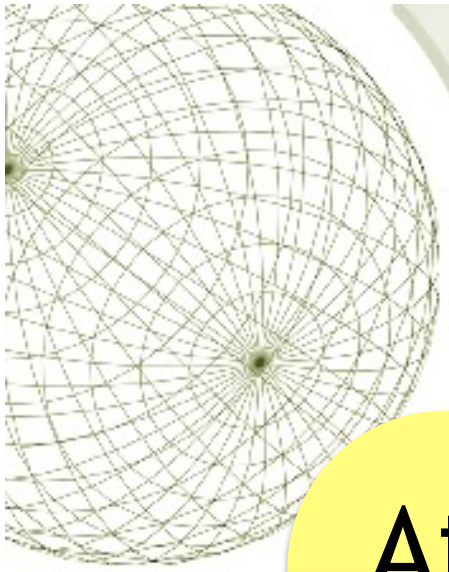
Web browsers support

The latest versions of all major web browsers support TLS 1.3

Ref: https://www.ssllabs.com/ssl-pulse/
http://en.wikipedia.org/wiki/Transport_Layer_Security (2019-09-12)

Heartbeat Protocol

- A heartbeat is a periodic signal generated by hardware or software to indicate normal operation or to synchronize other parts of a system
- A heartbeat protocol is typically used to monitor the availability of a protocol entity
- The heartbeat protocol runs on top of the TLS Record Protocol
  - Consists of two message types: heartbeat_request and heartbeat_response
- The heartbeat serves two purposes:
  - It assures the sender that the recipient is still alive, even though there may not have been any activity over the underlying TCP connection
  - It generates activity across the connection during idle periods, which avoids closure by a firewall that does not tolerate idle connections



SSL/TLS Attacks

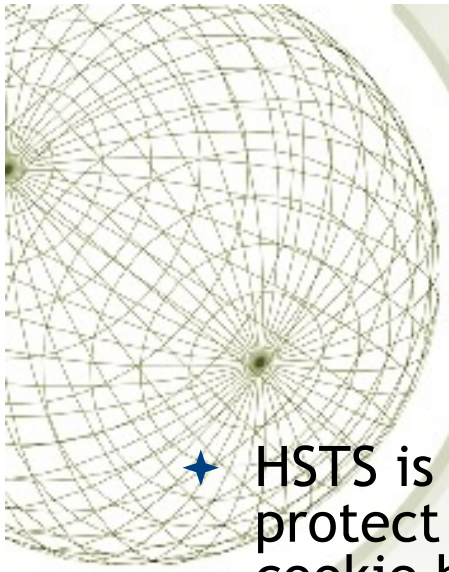
Attack categories

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

Heartbleed attack

- Heartbleed is a security bug in the OpenSSL cryptography library, which is a widely used implementation of the Transport Layer Security (TLS) protocol. It was introduced into the software in 2012 and publicly disclosed in April 2014.
- Heartbleed may be exploited regardless of whether the party is using a vulnerable OpenSSL instance for TLS as a server or a client. It results from improper input validation (due to a missing bounds check) in the implementation of the TLS heartbeat extension, thus the bug's name derives from heartbeat. The vulnerability is classified as a buffer over-read, a situation where more data can be read than should be allowed.

Källa: https://en.wikipedia.org/wiki/Heartbleed (2017-09-08)



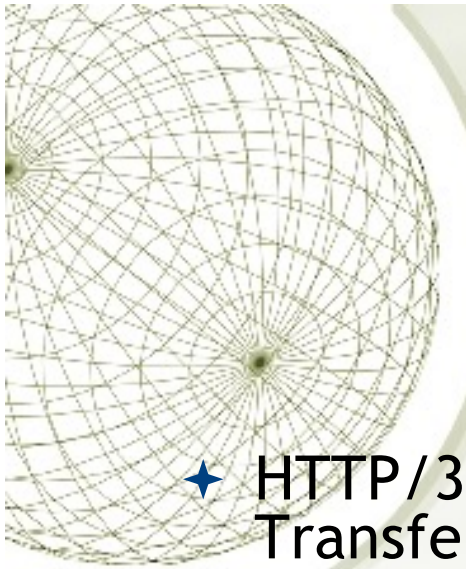
HTTP Strict Transport Security (HSTS)

- HSTS is a web security policy mechanism which helps to protect websites against protocol downgrade attacks and cookie hijacking. It allows web servers to declare that web browsers should only interact with it using secure HTTPS connections, and never via the insecure HTTP protocol.
- When a web application issues HSTS Policy to user agents, conformant user agents behave as follows:
  1. Automatically turn any insecure links referencing the web application into secure links. (For instance, http://example.com/some/page/ will be modified to https://example.com/some/page/ before accessing the server.)
  2. If the security of the connection cannot be ensured (e.g. the server's TLS certificate is not trusted), show an error message and do not allow the user to access the web application.
- Most browsers today enforce HTTPS making HSTS more or less obsolete

Datagram Transport Layer Security (DTLS)

- Datagram Transport Layer Security is a communications protocol that provides security for datagram-based applications by allowing them to communicate in a way that is designed to prevent eavesdropping, tampering, or message forgery.
- The DTLS protocol is based on the stream-oriented Transport Layer Security (TLS) protocol and is intended to provide similar security guarantees.
- The DTLS protocol datagram preserves the semantics of the underlying transport—the application does not suffer from the delays associated with stream protocols, but because it uses UDP, the application has to deal with packet reordering, loss of datagram and data larger than the size of a datagram network packet.

Source: https://en.wikipedia.org/wiki/Datagram_Transport_Layer_Security

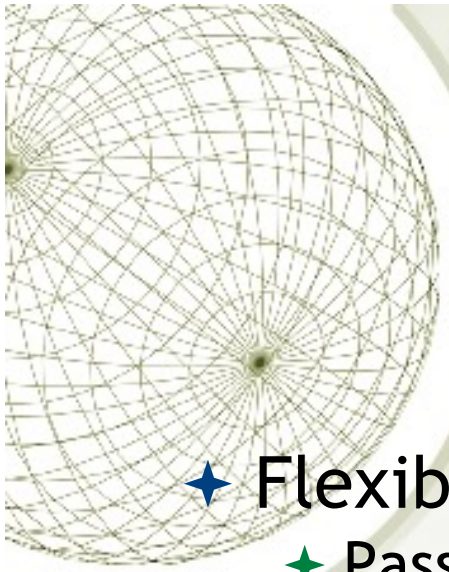


HTTP/3

- HTTP/3 is the third major version of the Hypertext Transfer Protocol used to exchange information on the World Wide Web, alongside HTTP/1.1 and HTTP/2.
- HTTP semantics are consistent across versions: the same request methods, status codes, and message fields are typically applicable to all versions.
- The differences are in the mapping of these semantics to underlying transports. Both HTTP/1.1 and HTTP/2 use TCP as their transport. HTTP/3 uses QUIC, a transport layer network protocol developed initially by Google where user space congestion control is used over the User Datagram Protocol (UDP)
- As of September 2021, the HTTP/3 protocol is an Internet-Draft and has multiple implementations.

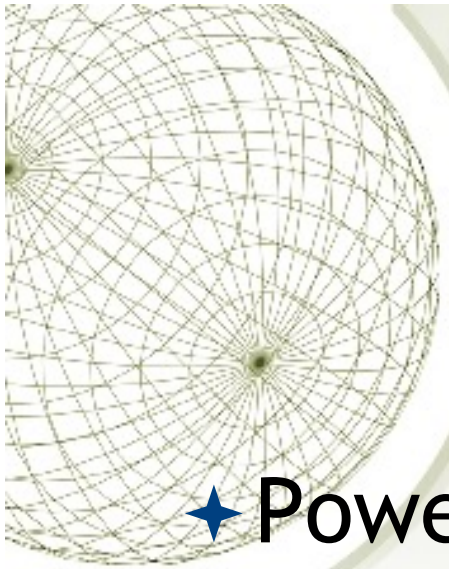
Secure Shell (SSH)

- Designed in 1995 by Tatu Ylonen
- Replaced in 1996 by SSHv2
  - Fixed security holes
  - Eventually standardized
- Less popular than SSL
  - But completely dominates a niche of the market



SSH features

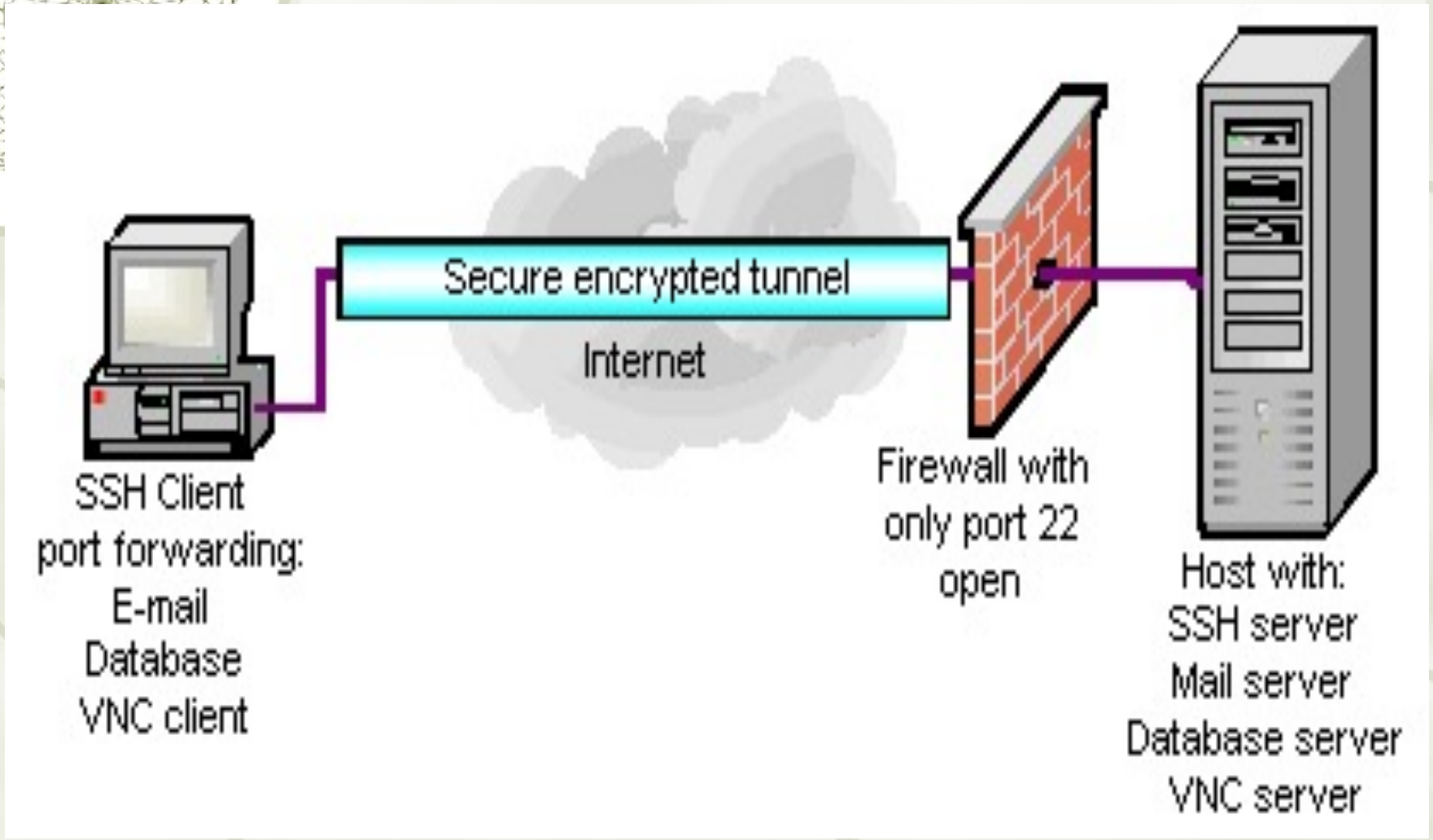
- Flexible authentication architecture
  - Password, public key, Secure ID, Kerberos, ...
- Perfect forward secrecy
  - Permanent (public) host key + temporary key
  - Host key signs temporary key
  - Key exchange done with temporary key
  - Even if the permanent key is compromised, no previous temporary keys will be compromised
- Port forwarding
  - Used to tunnel traffic on other ports
  - Especially X11 forwarding



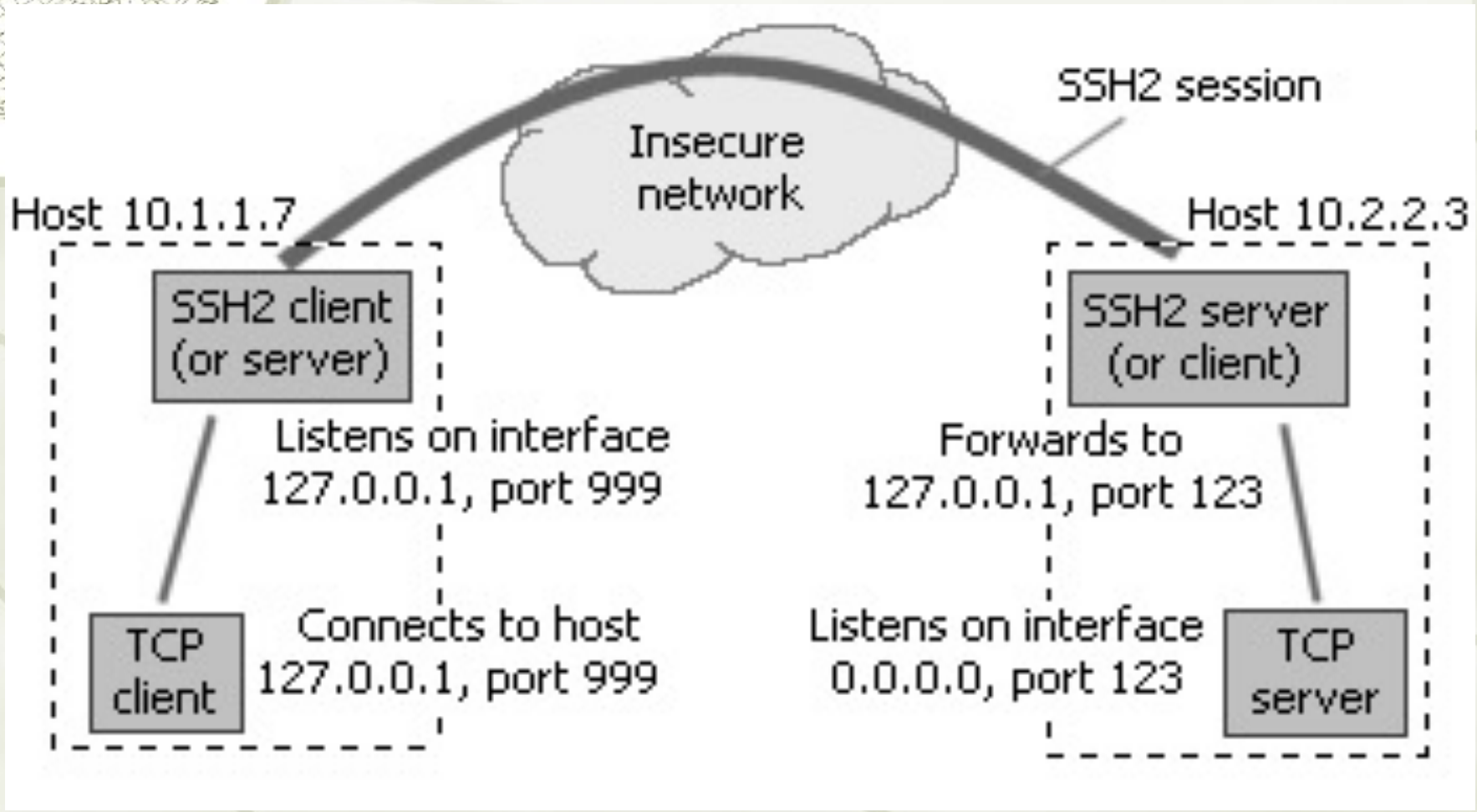
Port Forwarding

- 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.

Port Forwarding



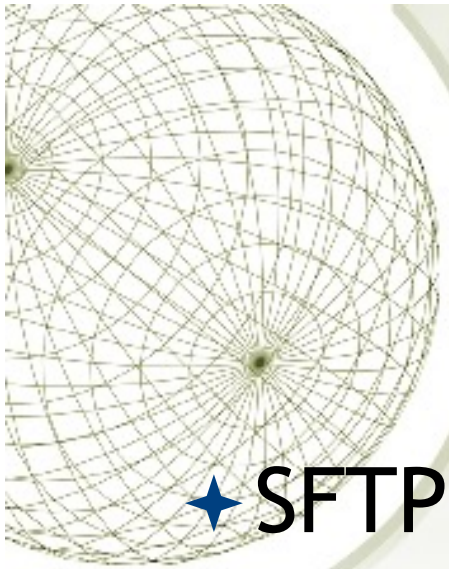
TCP/IP port forwarding details



Picture copied from http://www.bitvise.com/port-forwarding.html

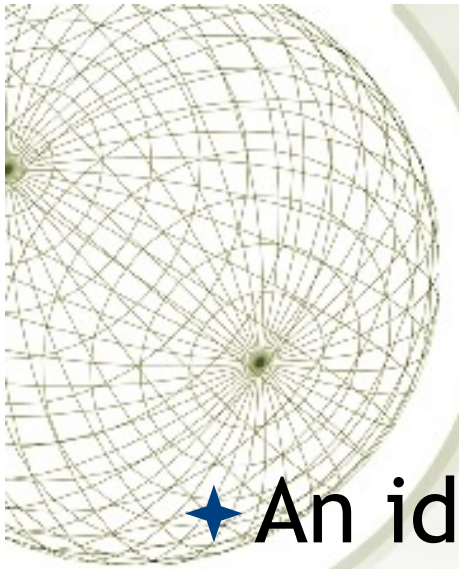
Secure File Transfer

- Secure File Transfer Protocol (SFTP) is a subsystem of the Secure Shell protocol.
- Separate protocol layered over the Secure Shell protocol to handle file transfers.



SFTP

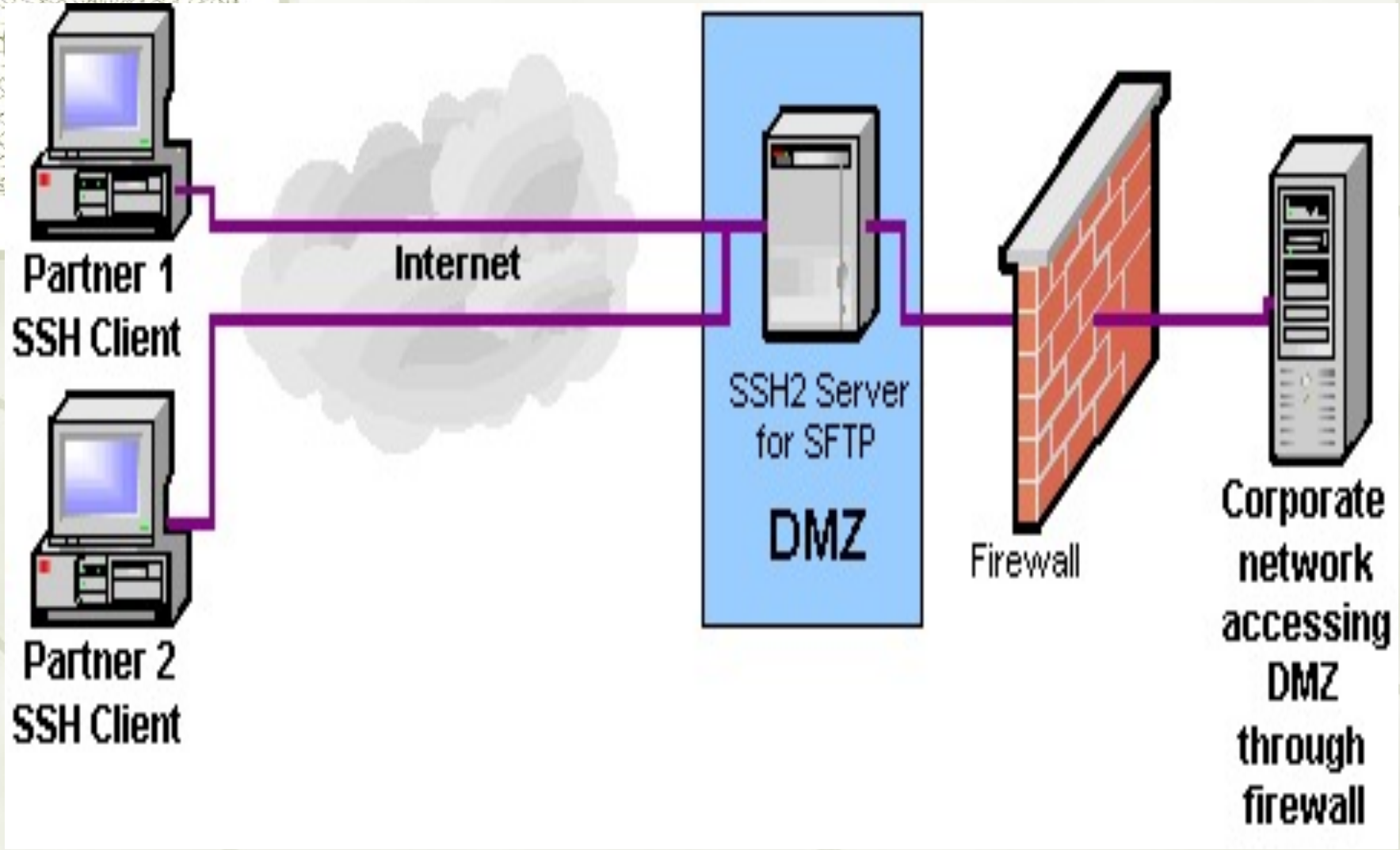
- SFTP encrypts both the username/password and the data being transferred.
- Uses the same port as the Secure Shell server, eliminating the need to open another port on the firewall or router.
- Using SFTP also avoids the network address translation (NAT) issues that can often be a problem with regular FTP.



SFTP

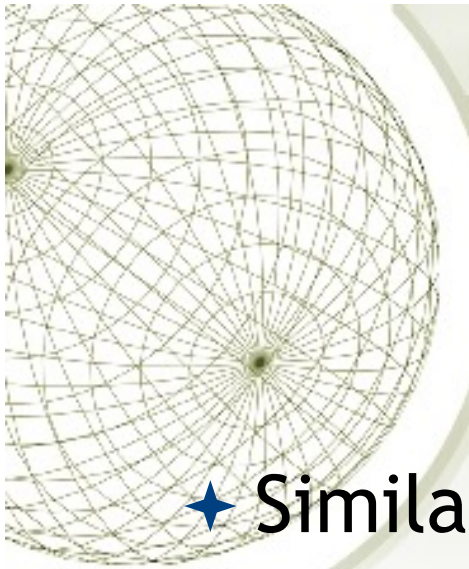
- An ideal use of SFTP is to fortify a server or servers outside the firewall or router accessible by remote users and/or partners (sometimes referred to as a secure extranet or DMZ).

Fortified DMZ file server



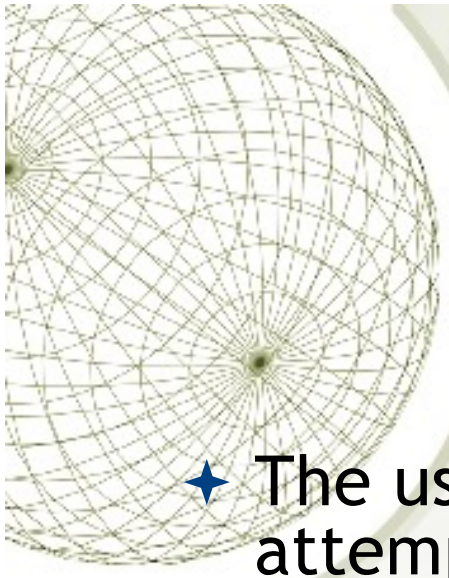
Components of Secure Shell

- SSHD Server: A program that allows incoming SSH connections to a machine, handling authentication, authorization.
- Clients: A program that connects to SSH servers and makes requests for service
- Session: An ongoing connection between a client and a server. It begins after the client successfully authenticates to a server and ends when the connection terminates.



SSH architecture

- Similar to SSL:
  - Clients, servers, socket-like interface
- Difference:
  - Initially no certificates (but still public keys)
  - Remember public key associated with host
- Intuition
  - MITM attacks are difficult
  - To evade detection, adversary must succeed at MITM every time

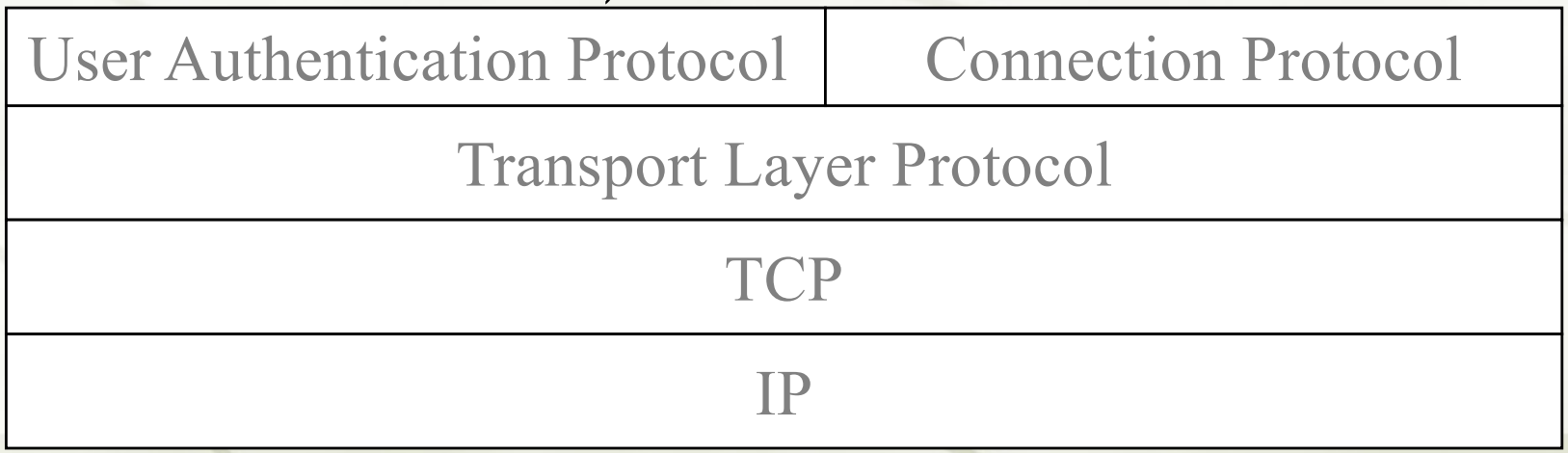


SSH Architecture

- The user initiates an SSH connection. SSH attempts to connect to port 22 on the remote host.
- If successful, SSHD on the machine Remote forks off a child SSHD process. This process will handle the SSH connection between the two machines.
- The child SSHD now forks off the command received from the original SSH client.
- The SSHD child process now encrypts every messages that has to be send to the SSH client.
- The SSH client decrypts the information and sends it to the user application.

SSH Protocol Stack

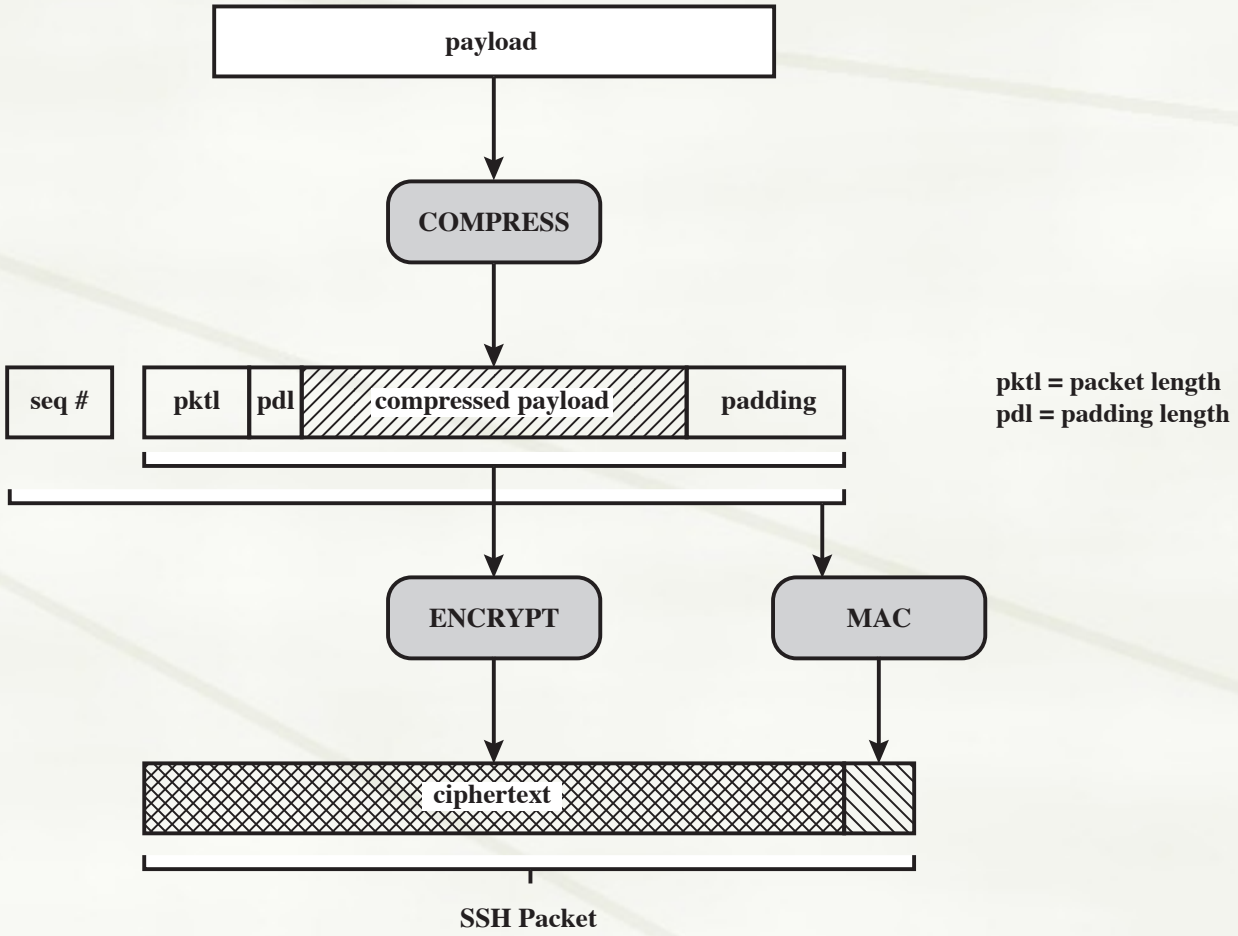
- SSH Transport Layer Protocol (TLP)
  - server authentication, confidentiality, and integrity
- SSH User Authentication Protocol (UAP)
  - run on top of the transport layer protocol
- SSH Connection Protocol (CP)
  - Provides interactive login sessions, remote execution of commands, forwarded TCP/IP connections, and forwarded X11 connections. (all these connections are called channels)



SSH Transport layer protocol

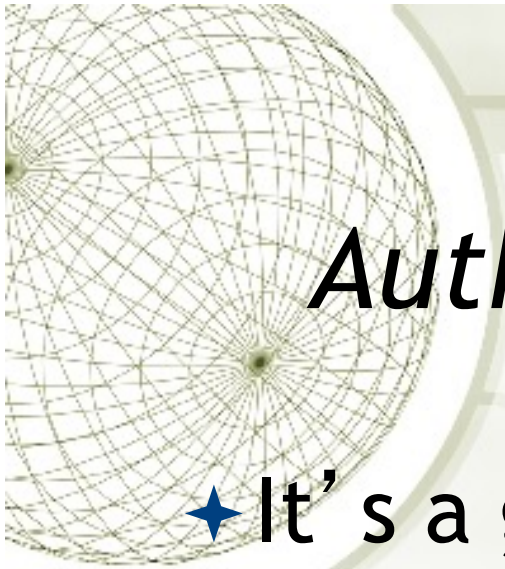
- Client initiates connection to the server
- Identification string exchange
- Algorithms exchange
- Key exchanges include host key sent to client
  - Diffie-Hellman key exchange
  - Client selects a random session key

SSH Transport Layer Protocol Packet Formation



SSH Transport Layer Cryptographic algorithms

- Cipher (examples)
  - AES with a 128-bit key (Recommended)
  - 3DES in CBC mode (Required)
- MAC algorithm (examples)
  - First 96 bits of HMAC-SHA1; Digest length = 12; Key length = 20 (Recommended)
  - HMAC-SHA1; Digest length = Key length = 20 (Required)
- Compression algorithm
  - None (Required)
  - Zlib (RFC 1950 and 1951)

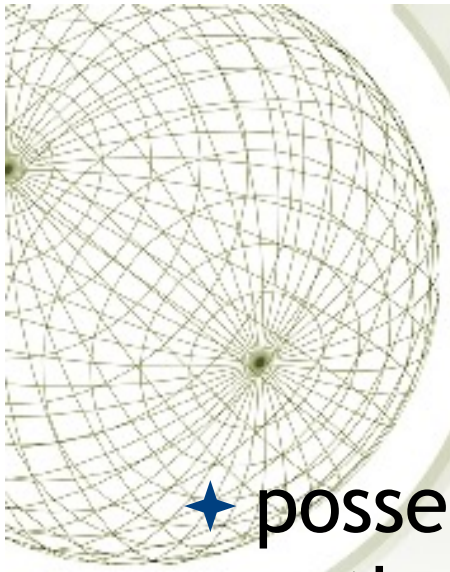


Authentication Protocol Basics

- It’s a general-purpose user authentication protocol
- It is intended to be run over the SSH transport layer protocol
- It assumes that the underlying protocols provide integrity and confidentiality protection

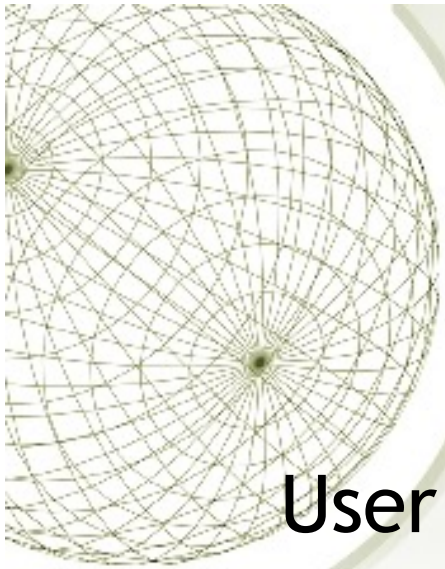
The Authentication Protocol Framework

- Client is a program running on behalf of the user
- The server has complete control over authentication as it tells the client which authentication methods can be used
- The client can choose from offered methods, making it flexible



Authentication methods Public Key method

- possession of a private key serves as authentication
- This method works by sending a signature created with a private key of the user
- User sends a request for use of a public-key algorithm
- Server rejects the request if it doesn‘t support that algorithm



Authentication methods Password Method

User sends the following packet

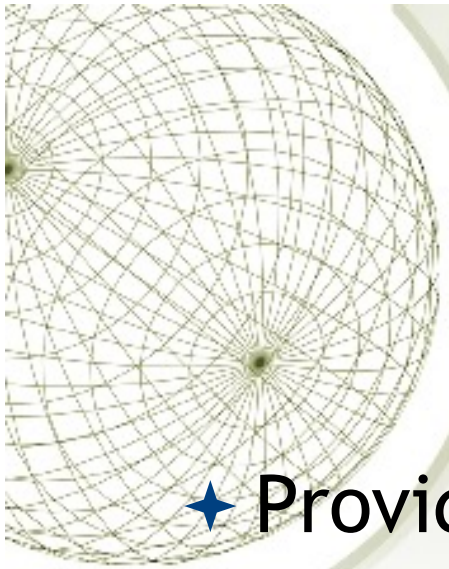
byte	SSH_MSG_USERAUTH_REQUEST
string	user name
string	service
string	"password"
boolean	FALSE
string	plaintext password

Even though the clear text password is transmitted in the packet, the entire packet is encrypted by the transport layer.

Authentication methods Host Based Authentication

- It works by having the client send a signature created with the private key of the client host, which the server checks with that host's public key
- Once the client host's identity is established, authorization (but no further authentication) is performed based on the user name

Care should be taken to ensure that a regular user doesn’t obtain the client host key.



SSH Connection Protocol

- Provides interactive login sessions, remote execution of commands, forwarded TCP/IP connections and forwarded X11 connections.
- All the channels are multiplexed into a single encrypted tunnel.
- Has been designed to run on top of the SSH transport layer and together with the user authentication protocols.

SSH attacks

- Man-in-the-middle
  - How does a client verify the host key?
    - Check local database
      - If exist, the server is authenticated, connection is established
      - If not exist, currently let user determine
        - (because public certificates are not used widely)
        - Build connection with saving the server host key
        - Build connection without saving.
  - Therefore there exists a possibility of man-in-the-middle attack.
- Denial of service
  - Attacker sets up many connections without user authentication.