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

- One of the most widely used security services
- General-purpose service implemented as a set of protocols that rely on TCP
- Subsequently became Internet standard RFC4346: Transport Layer Security (TLS)

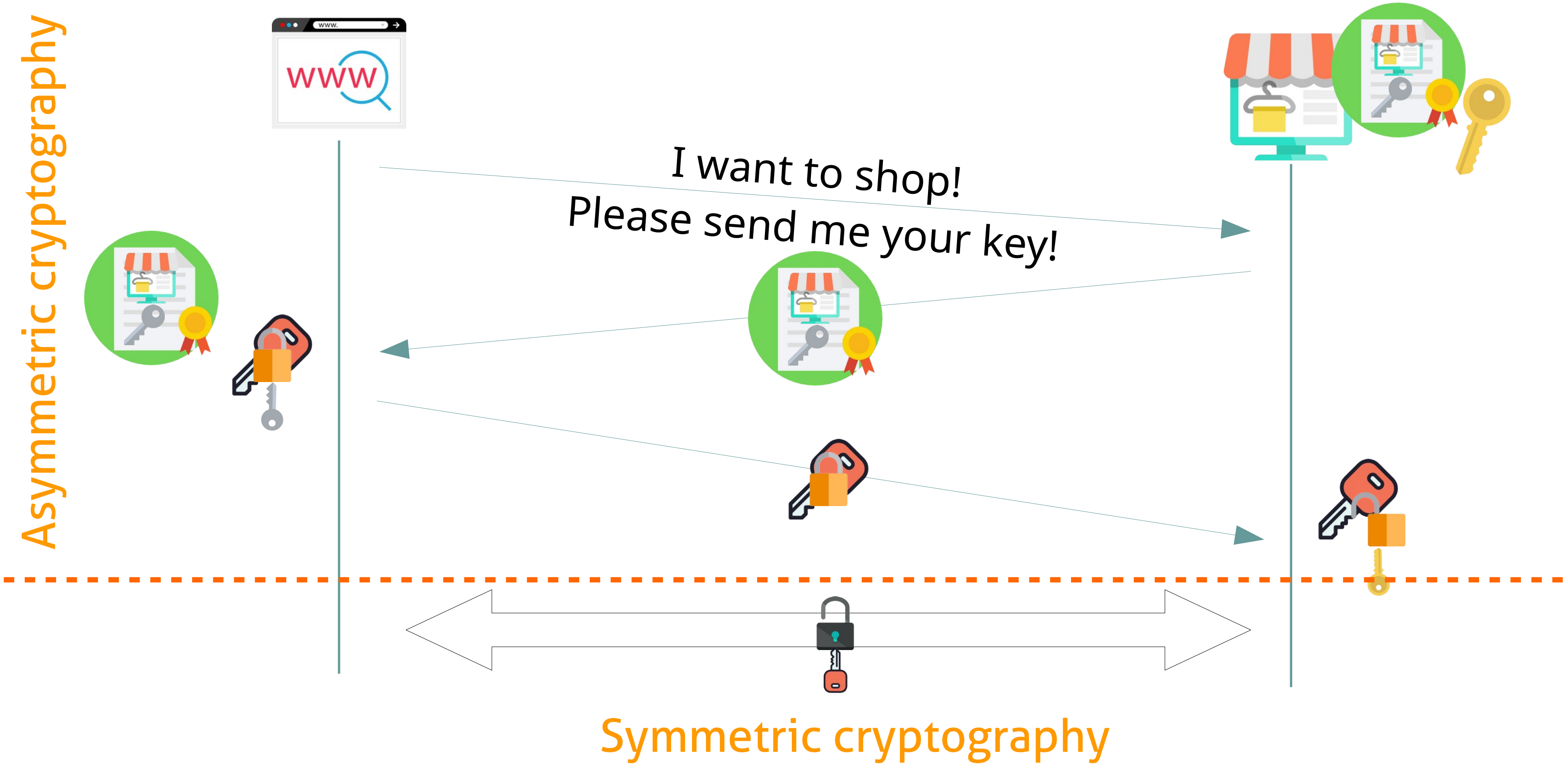
Two implementation choices:

- Provided as part of the underlying protocol suite
- Embedded in specific packages

SSL/TLS

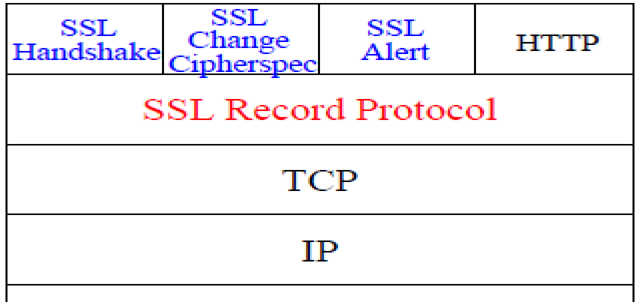
- SSL 3.0 has become TLS standard (RFC 2246) with small changes
- Applies security in the Transport layer.
- Originally designed (by Netscape) to offer security for client-server sessions.
- If implemented on boundary routers (or proxies), can provide a tunnel between two sites – typically LANs.
- Placed on top of TCP, so no need to change TCP/IP stack or OS.
- Provides secure channel (byte stream)
  - Any TCP-based protocol
  - https:// URIs, port 443
  - NNTP, SIP, SMTP...
- Optional server authentication with public key certificates
  - Common on commercial sites

How HTTPS (HTTP on top of TLS) works

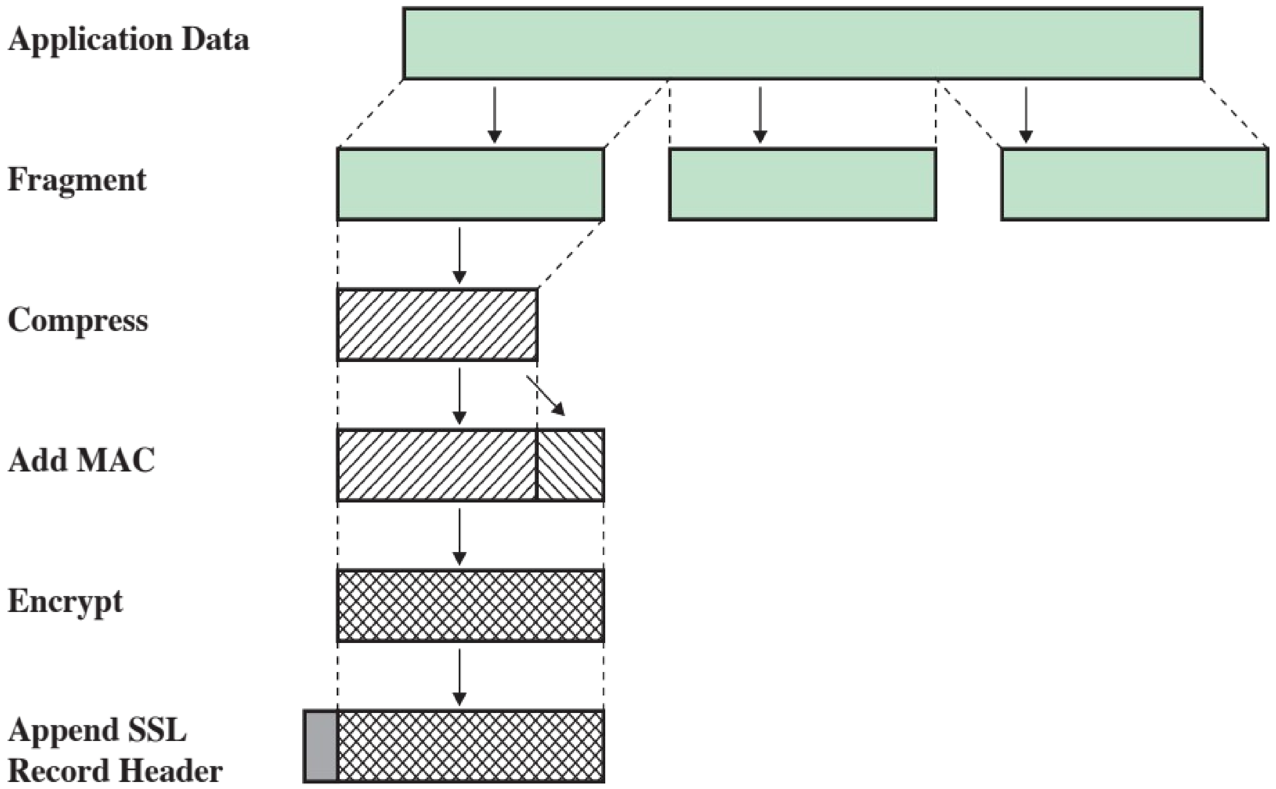


SSL protocol Architecture

- Adds extra layer between T- and A-layers, and extra elements to A-layer
- Record Protocol: Protocol offering basic encryption and integrity services to applications
- Application Protocols: control operation of the record protocol
  - Handshake: Used to authenticate server (and optionally client) and to agree on encryption keys and algorithms.
  - Change cipher spec: Selects agreed keys and encryption algorithm until further notice.
  - Alert: Transfers information about failures.

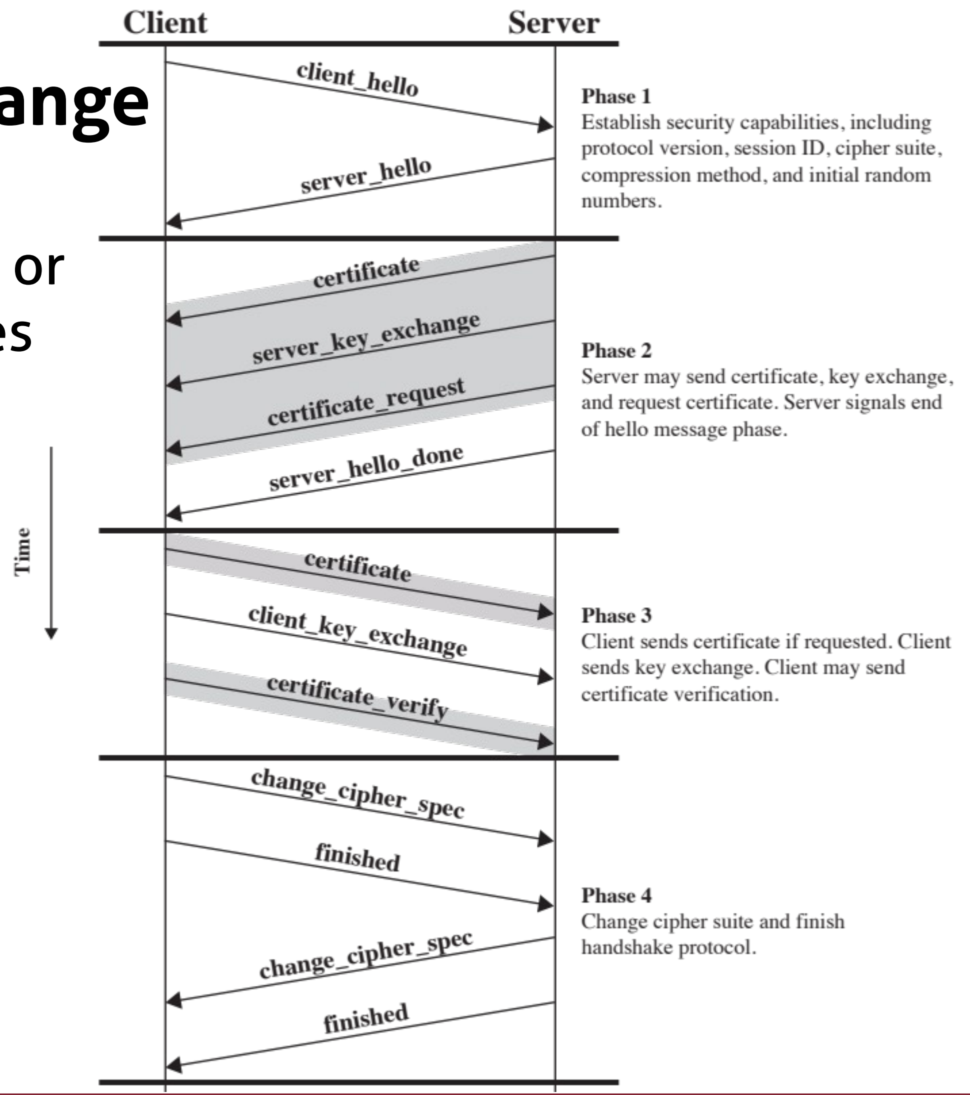


TLS Record protocol operation



Handshake protocol exchange

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

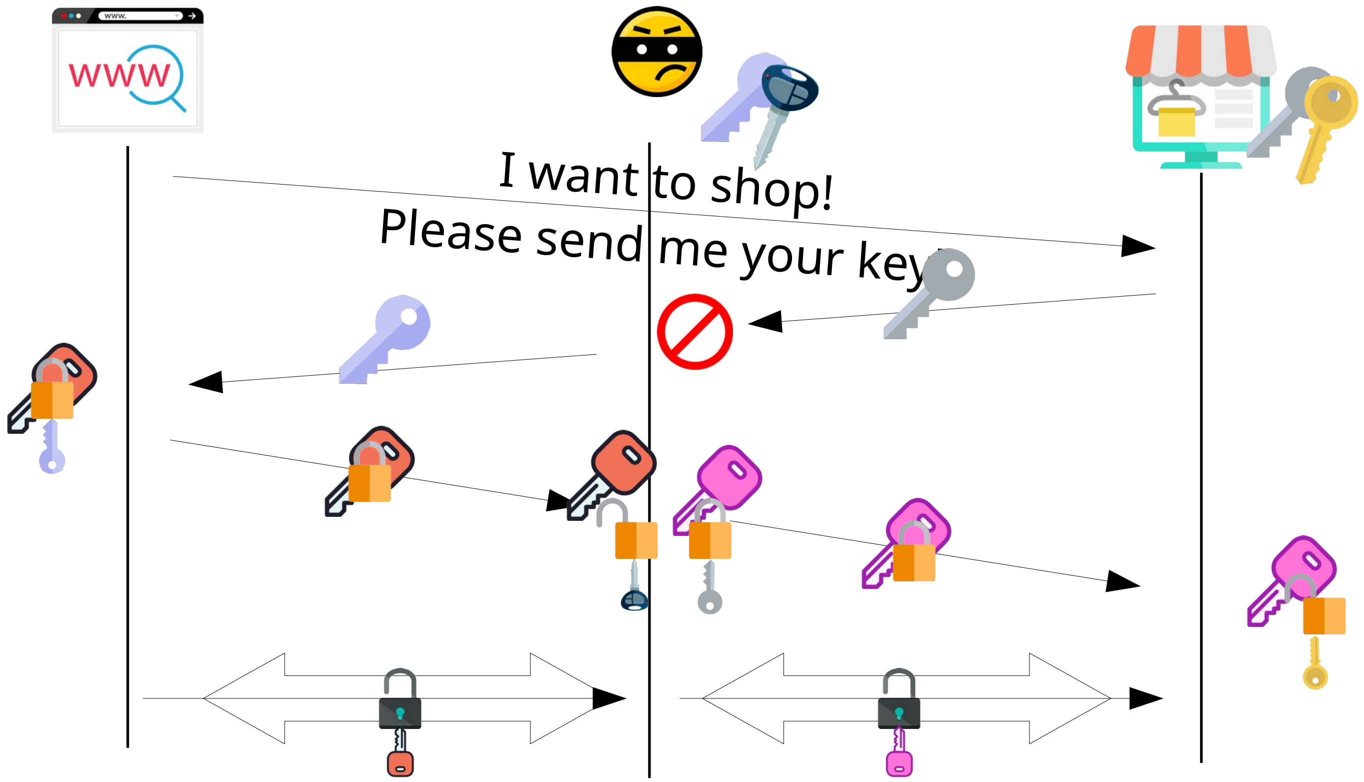

Client
Server
client_hello
server_hello
certificate
server_key_exchange
certificate_request
server_hello_done
certificate
client_key_exchange
certificate_verify
change_cipher_spec
finished
change_cipher_spec
finished
Time
Phase 1
Establish security capabilities, including protocol version, session ID, 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.

SSL/TLS Handshake Protocol

4-phase “Client/Server” protocol to establish parameters of the secure connection (“Client” is the initiator):

1) **Hello**: Establishment of security capabilities: Client sends list of possibilities, in order of preference. Server selects one, and informs Client of its choice. Parties also exchange random noise for use in key generation.
2) **Server authentication and key exchange**: Server executes selected key exchange protocol (if needed). Server sends authentication info. (e.g. X.509 cert.) to Client.
3) **Client authentication and key exchange**: Client executes selected key exchange protocol (mandatory). Client sends authentication info. to Server (optional).
4) **Finish**: Shared secret key is derived from pre-secrets exch. in 2, 3. Change Cipher Spec. protocol is activated. Summaries of progress of Handshake Protocol are exchanged and checked by both parties.

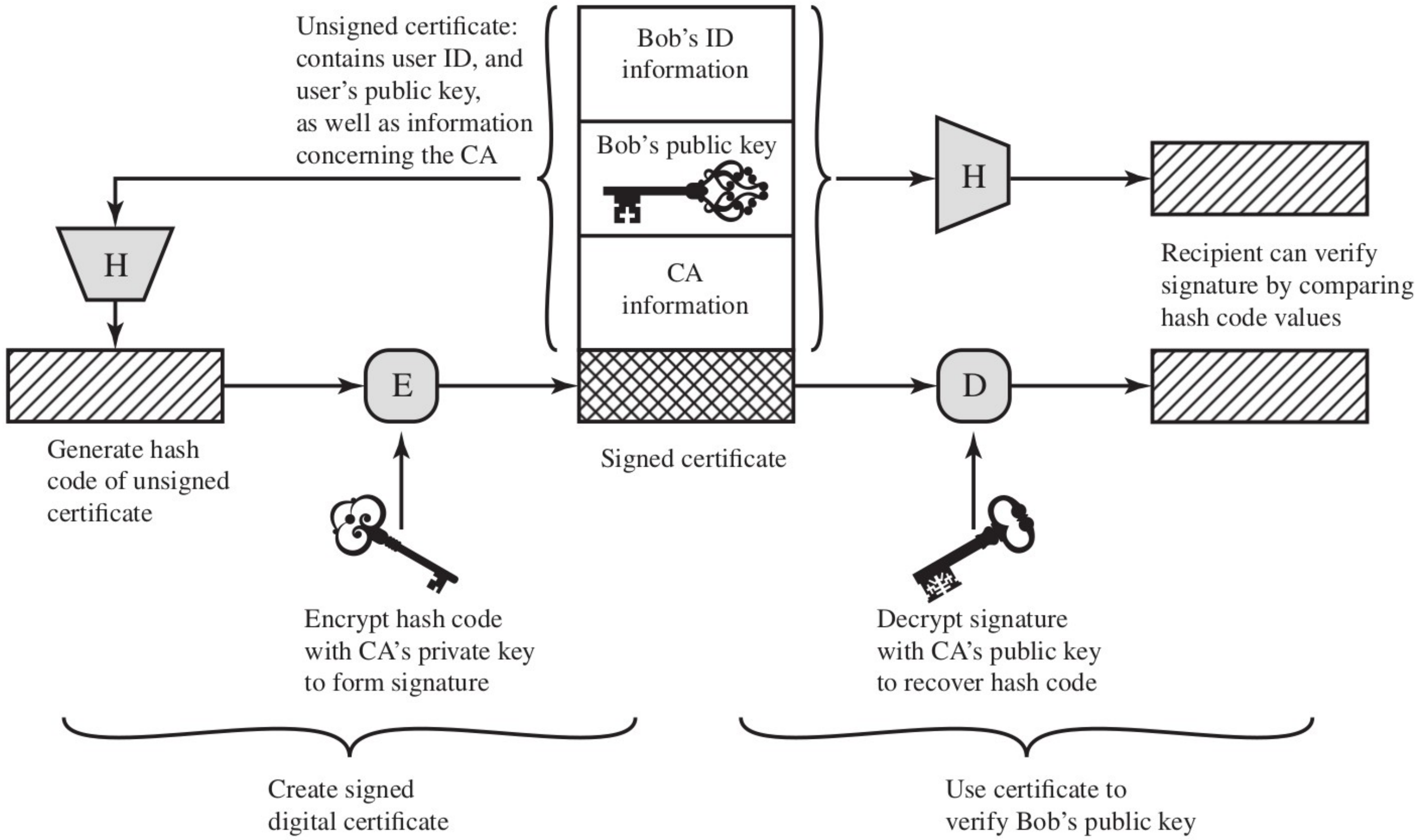
Can we trust a public key?



Digital certificates

- A document that certifies the relation between a public key and its owner
- How? With a digital signature…
- But, to verify a digital signature, we need another public key!
- Then??
- We need a public key that we **trust**
- Trusted public keys are stored in certificates of **Certification Autorities (CA)**

Public key certificate use

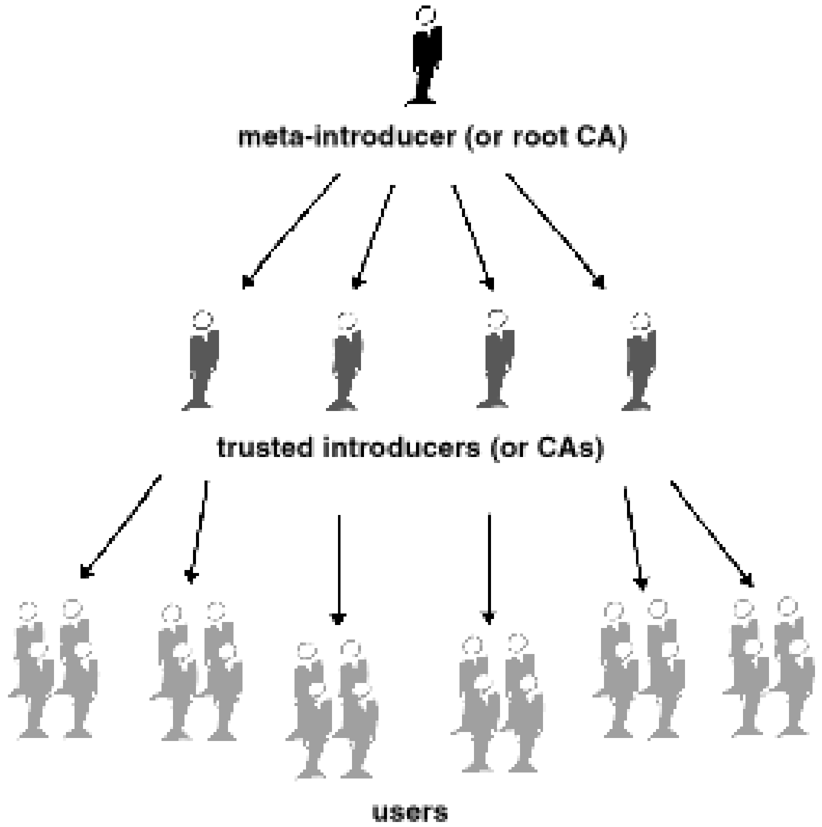


Certification Authority (CA)

- An organization that issues digital certificates
- The CA performs many tasks:
  - Receive application for keys.
  - Verify applicant's identity, conduct due diligence appropriate to the trust level, and issue key pairs.
  - Store public keys and protect them from unauthorized modification.
  - Keep a register of valid keys.
  - Revoke and delete keys that are invalid or expired. Maintain a certificate revocation list (CRL).
- Certificates of CAs are stored in any computer that want to use internet securely

PKI: Public Key Infrastructure

- Certification authorities are organized in a hierarchy, called Public Key Infrastructure
- To verify a certificate, one needs to verify all the signatures up to the top of the hierarchy
- X.509 is the standard



Certificate Authority (CA)

Certificate consists of:

A public key with the identity of the key's owner

Signed by a trusted third party

Typically the third party is a CA that is trusted by the user community (such as a government agency, telecommunications company, financial institution, or other trusted peak organization)

User can present his or her public key to the authority in a secure manner and obtain a certificate

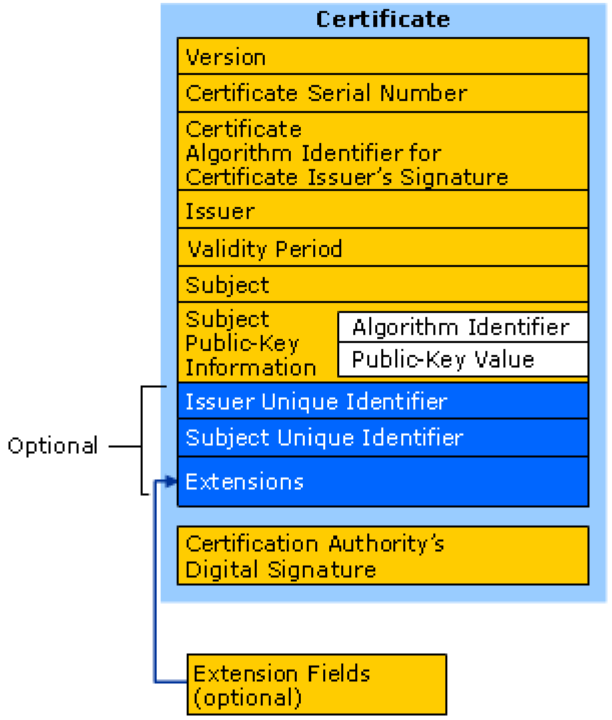
User can then publish the certificate or send it to others

Anyone needing this user's public key can obtain the certificate and verify that it is valid by way of the attached trusted signature

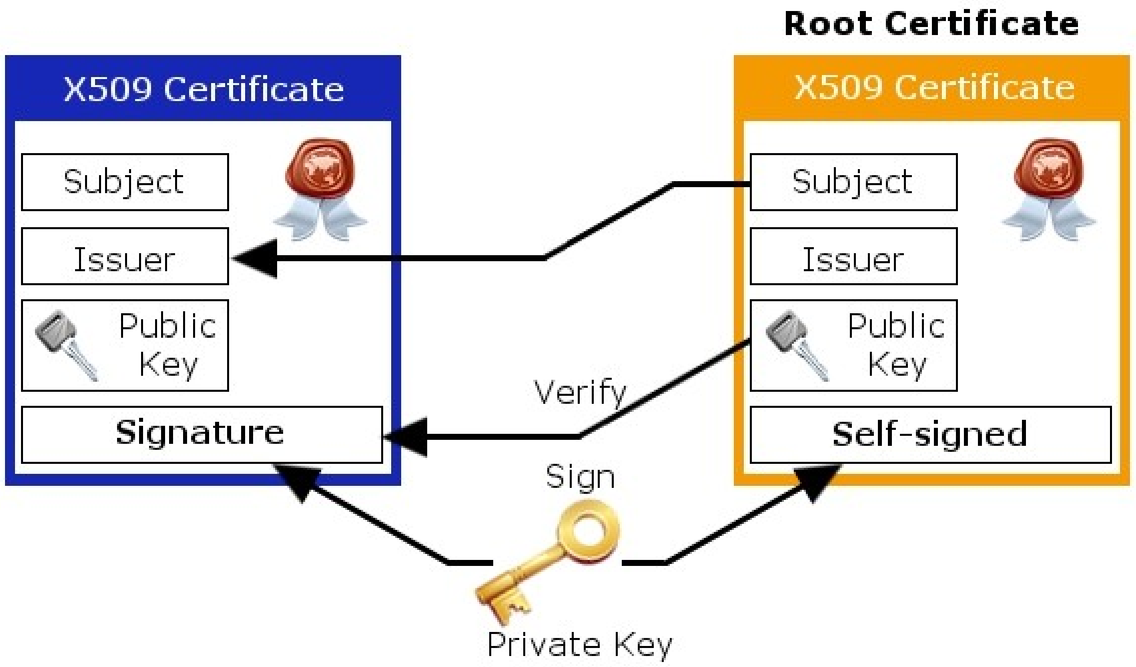
X.509

- Specified in RFC 5280
- The most widely accepted format for public-key certificates
- Certificates are used in most network security applications, including:
  - IP security (IPSEC)
  - Secure sockets layer (SSL)
  - Secure electronic transactions (SET)
  - S/MIME
  - eBusiness applications

X.509 certificate



Certificate
Version
Certificate Serial Number
Certificate Algorithm Identifier for Certificate Issuer's Signature
Issuer
Validity Period
Subject
Subject Public-Key Information
Algorithm Identifier
Public-Key Value
Issuer Unique Identifier
Subject Unique Identifier
Extensions
Optional
Certification Authority's Digital Signature
Extension Fields (optional)



SSL/TLS Security Capabilities

- Conventionally expressed by a descriptive string, specifying:
  - Version of SSL/TLS
  - Key exchange algorithm
  - Grade of encryption (previous to TLSv1.1)
  - Encryption algorithm
  - Mode of block encryption (if block cipher used)
  - Cryptographic checksum algorithm
- Example: TLS_RSA_WITH_AES_128_CBC_SHA
  - TLS → (Latest version of) TLS
  - RSA → RSA key exchange
  - WITH → (merely filler...)
  - AES_128 → 128-bit AES encryption
  - CBC → Cipher Block Chaining
  - SHA → Use HMAC-SHA digest

Key exchange and authentication

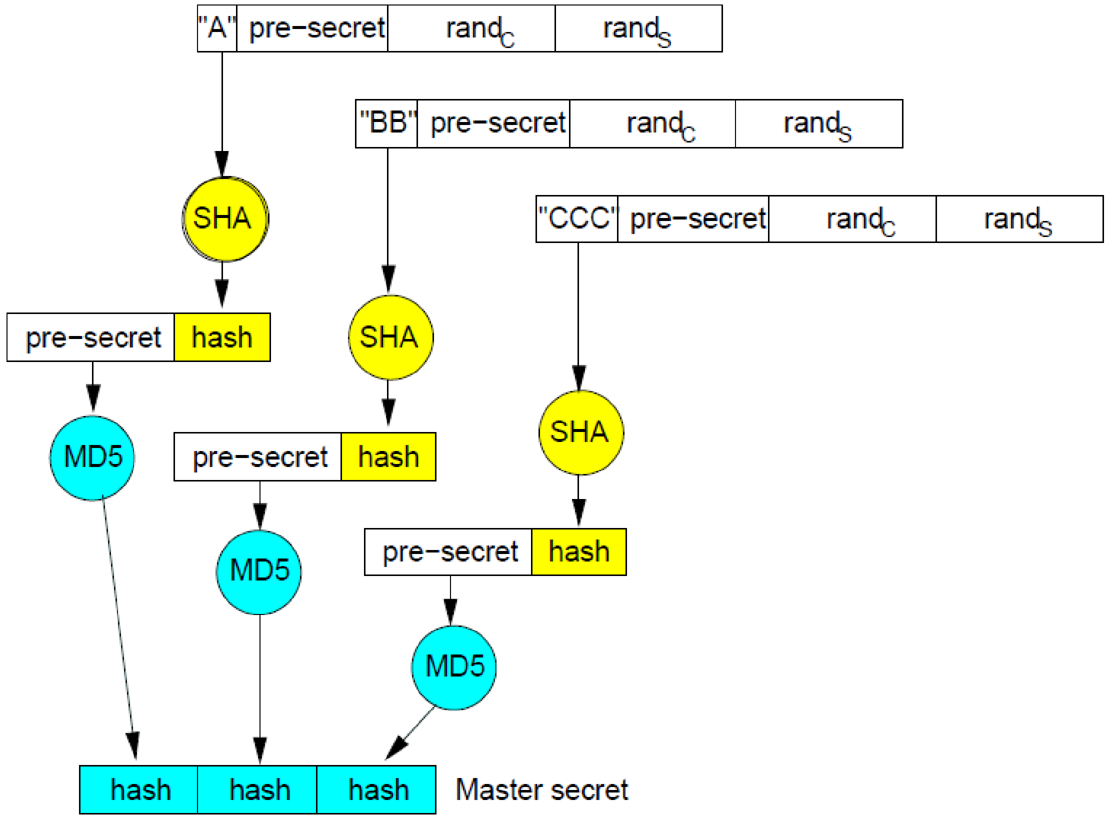
Possible ways of agreeing on secrets in TLS are:

- RSA: RSA key exch. (secret encrypted with recipient’s publ. key)
- DHE RSA: Ephemeral Diffie-Hellman with RSA signatures
- DHE DSS: Ephemeral Diffie-Hellman with DSS signatures
- DH DSS: Diffie-Hellman with DSS certificates
- DH RSA: Diffie-Hellman with RSA certificates
- DH anon: Anonymous Diffie-Hellman (no authentication)
- NULL No key exch.

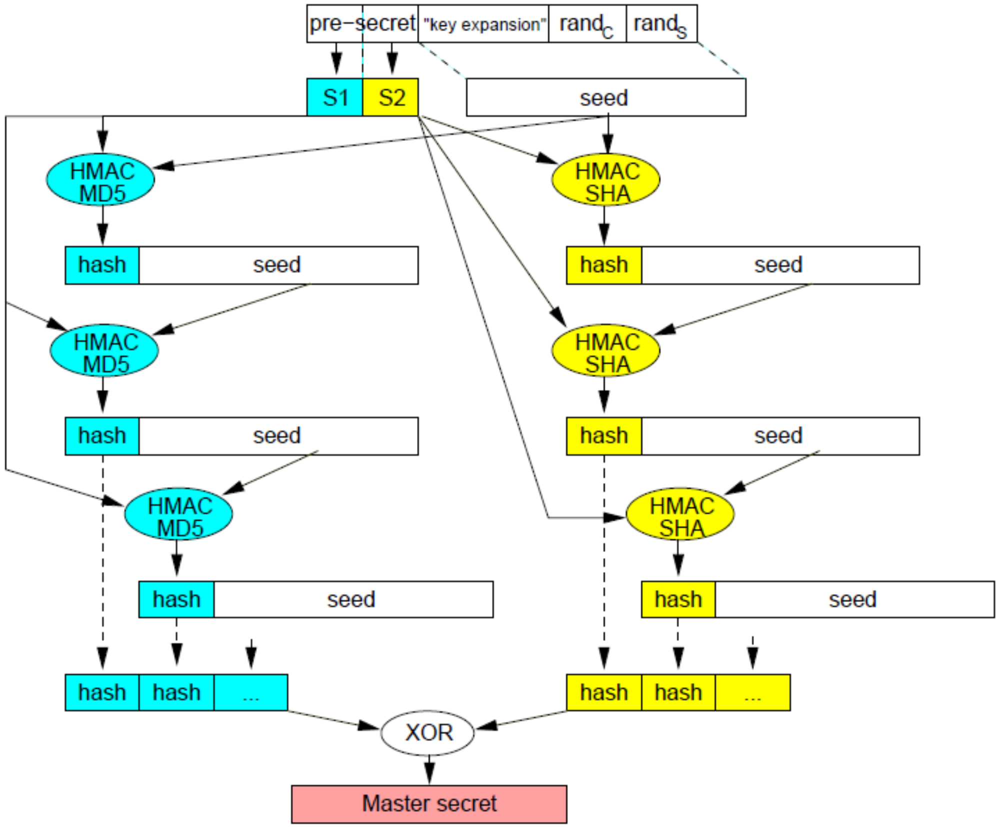
Variant: If followed by “EXPORT_”, weak encryption is used. (This option only available prior to TLSv1.1)

- **Note**: “Key exchange” only establishes a pre-secret! From this, a master secret is derived by a pseudo-random function (PRF). Shared secret encryption key is derived by expansion of master secret with another PRF. (In TLS several keys are derived for different purposes.)

SSL Master Secret



TLS Master Secret



SSL/TLS Heartbeat

- It is an extension (RFC 6520) that allows to keep an established session alive
  - That is, as soon as the data exchange between two endpoints terminates, the session will also terminate
- To avoid the re-negotiation of the security parameters for establishing a secure session, we can keep using the same parameters even if there is no exchange of data
- It introduces two messages: **HeartbeatRequest** and **HeartbeatResponse**

Heartbeat exchange

- When one endpoint sends a HeartbeatRequest message to the other endpoints, the former also starts what is known as the **retransmit timer**
  - During the time interval of the retransmit timer, the sending endpoint will not send another HeartbeatRequest message.
- An SSL/TLS session is considered to have terminated in the absence of a HeartbeatResponse packet within a time interval

Heartbeat payload

- As a protection against a replay attack, HeartbeatRequest packets include a payload that must be returned without change by the receiver in its HeartbeatResponse packet
- The Heartbeat message is defined as

```
struct {
      HeartbeatMessageType type;
      uint16 payload_length;
      opaque payload[HeartbeatMessage.payload_length];
      opaque padding[padding_length];
} HeartbeatMessage;
```

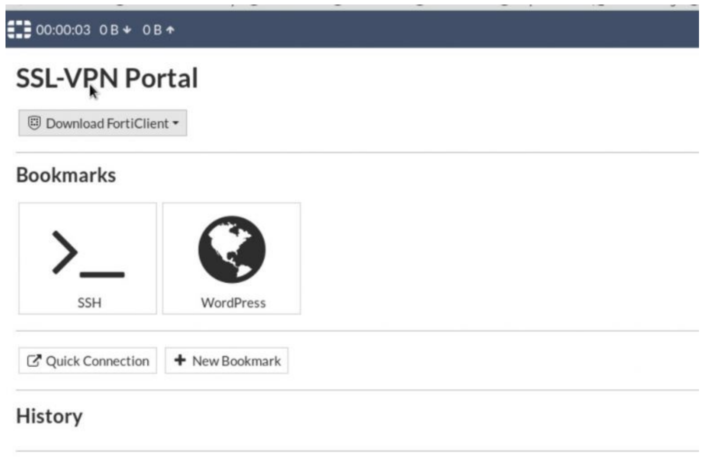
Heartbleed bug

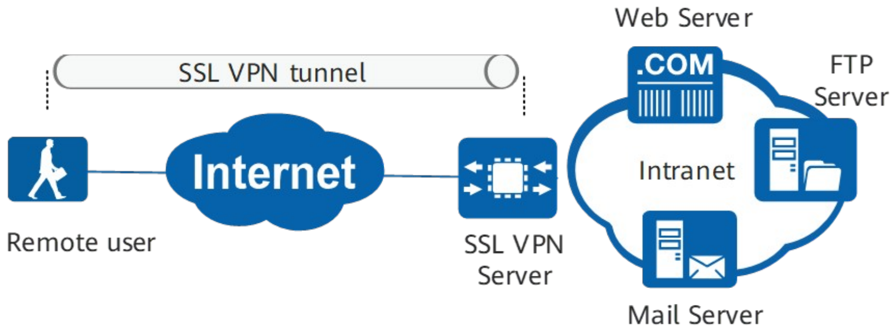
- Bug in OpenSSL library (4/4/2014)
- The receiver of request did not check that the **size of the payload in the packet** actually equaled the **value given** by the sender to the payload length field in the request packet
  - The attacker sends little data but sets the size to **max**
  - The receiver allcates that amount of memory for the response and copied **max bytes** from the mem location where the request packet was received
  - Then, the actual payload returned could potentially include objects in the memory that **had nothing to do** with the received payload
    - Objects could be private keys, passwords, and such…

SSL VPN Architecture

Two primary models:

- SSL Portal VPN
  - Remote users can access web-based services provided on the gateway
  - VPN gateway is reachable from a Web browser
- SSL Tunnel VPN
  - Remote users can access network services protected by VPN gateway
  - More capabilities than portal VPNs, as easier to provide more services





SSL VPN functionalities

Most SSL VPNs offer one or more core functionalities:

- Proxying
  - Intermediate device appears as true server to client
- Application translation
  - Conversion of information from one protocol to another.
    - e.g. Portal VPN offers translation for applications which are not Web-enabled, so users can use Web browser to access applications with no Web interface.
- Network extension
  - Provision of partial or complete network access to remote users, typically via Tunnel VPN

SSL VPN Securty Services

Typical services include:

- **Authentication** Via strong authentication methods, such as two-factor authent., X.509 certificates, smartcards, security tokens etc. May be integrated in VPN device or external authent. server.
- **Encryption** and integrity protection: Via the use of the SSL/TLS protocol.
- **Access control**: May be per-user, per-group or per-resource.
- **Endpoint security controls**: Validate the security compliance of clients attempting to use the VPN.
    - e.g. presence of antivirus system, updated patches etc.
- **Intrusion prevention**: Evaluates decrypted data for malicious attacks, malware etc.