A.D. 1308
unipg
DIPARTIMENTO
DI MATEMATICA E INFORMATICA

Security in Computer Networks and TLS/SSL

Corso di Introduzione alla Sicurezza Informatica

Chiara Luchini

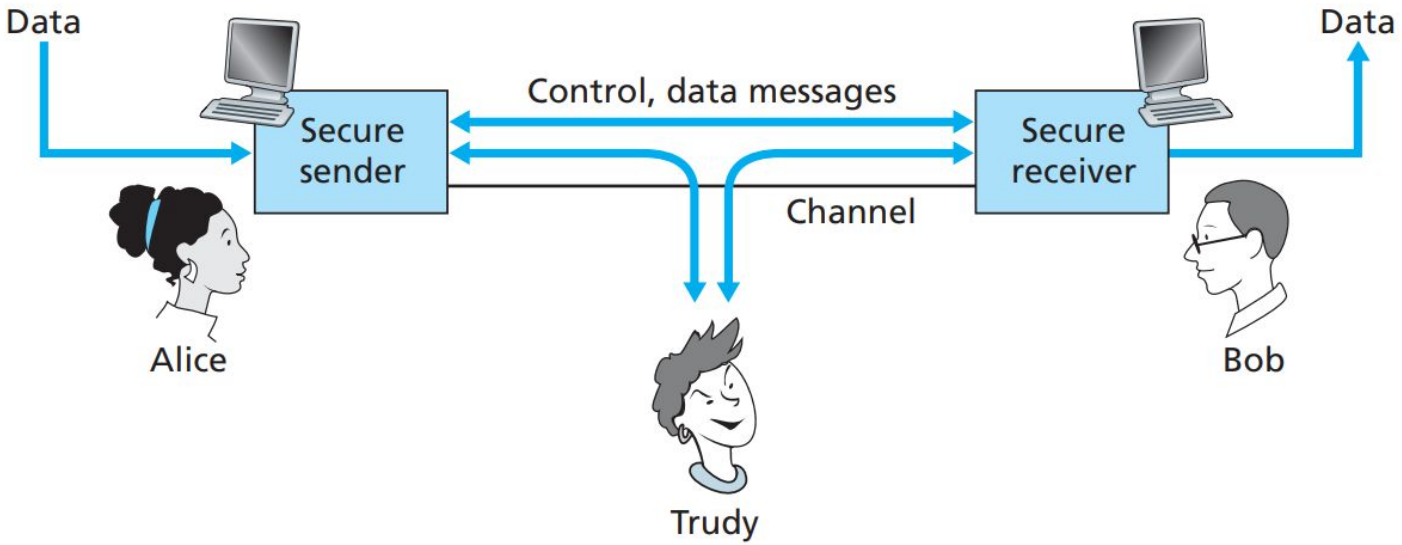
Anno Accademico 2024-2025

Network Security

Properties of Network Security

- **confidentiality**: only sender, intended receiver should “understand” message contents sender encrypts message receiver decrypts message.
- **authentication**: sender, receiver want to confirm identity of each other.
- **message integrity**: sender, receiver want to ensure message not altered (in transit, or afterwards) without detection.
- **access and availability**: services must be accessible and available to users.

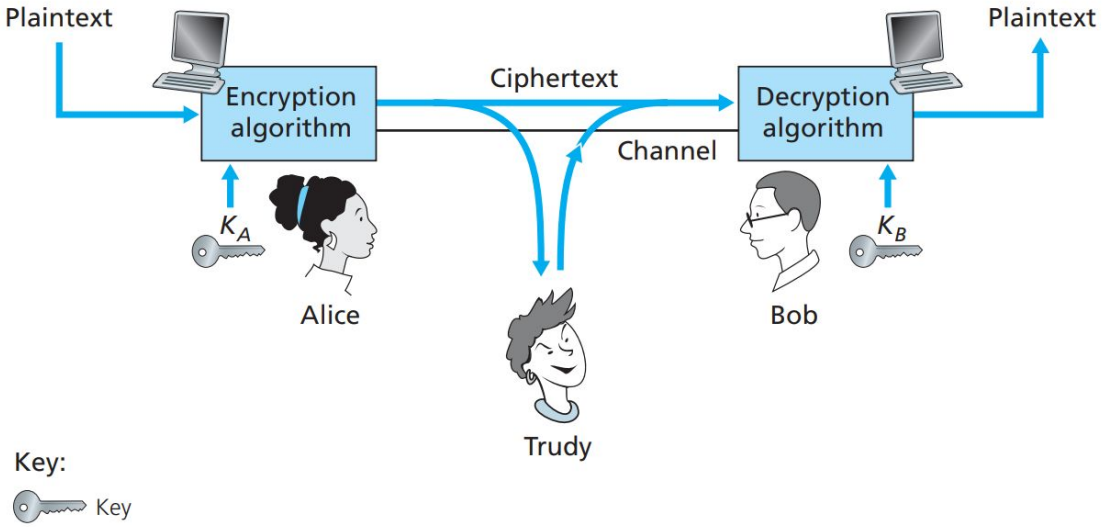
Alice, Bob and Trudy



Principles of Cryptography

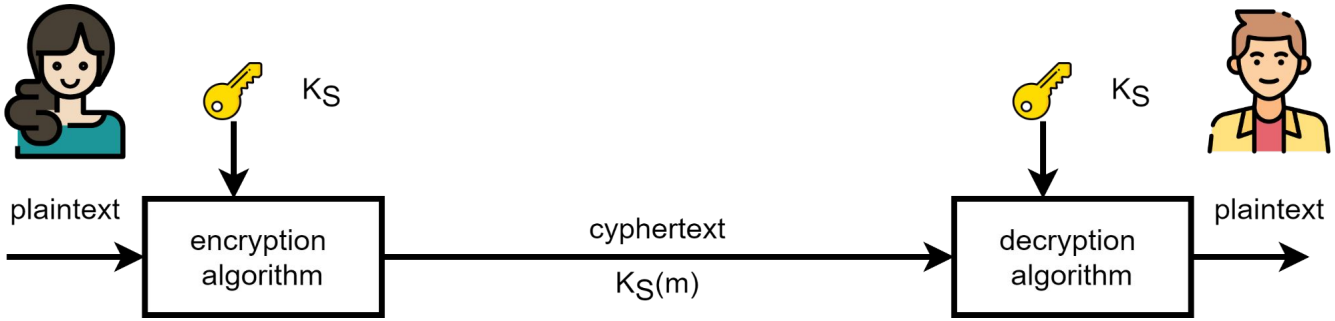
Cryptography components

- **m**= plaintext message
- $\mathbf{K_A(m)}$= ciphertext, encrypted with key K_A
- **m**= K_B (K_A(m))



Symmetric Key Cryptography

- **Symmetric key crypto:** Bob and Alice share same (symmetric) key (**DES**, **AES**)

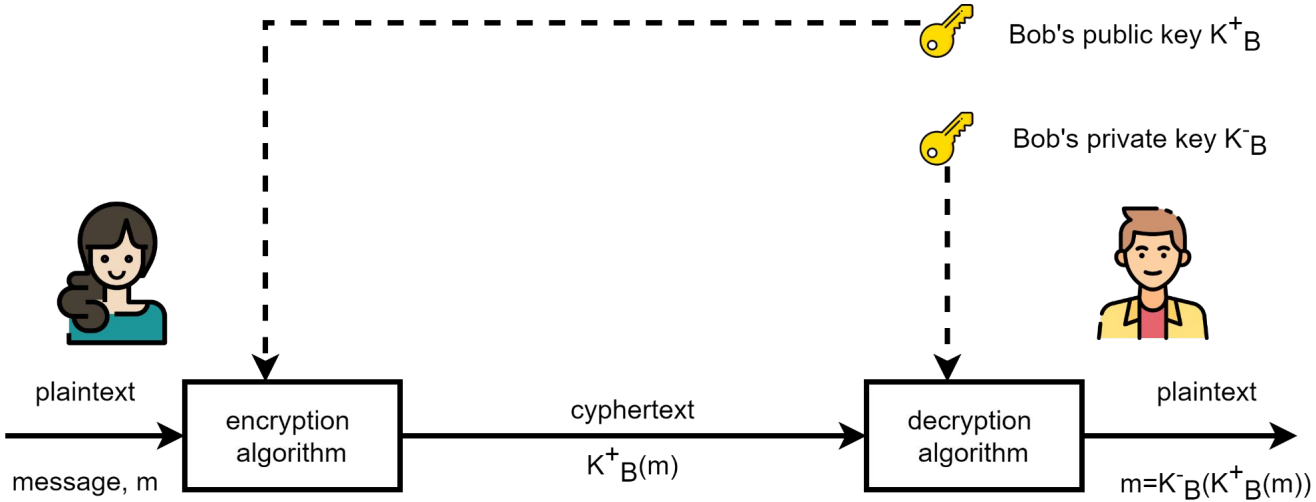



```
plaintext:   abcdefghijklmnopqrstuvwxyz
ciphertext:  mnbvcxzasdfghjklpoiuytrewq
```

e.g.: `Plaintext: bob. i love you. alice`

Public Key Cryptography

- Radically different approach (**Diffie-Hellman**, **RSA**)
- Sender, receiver **do** not share secret key
- **Public** encryption key known to **all**
- **Private** decryption key known only to **receiver**

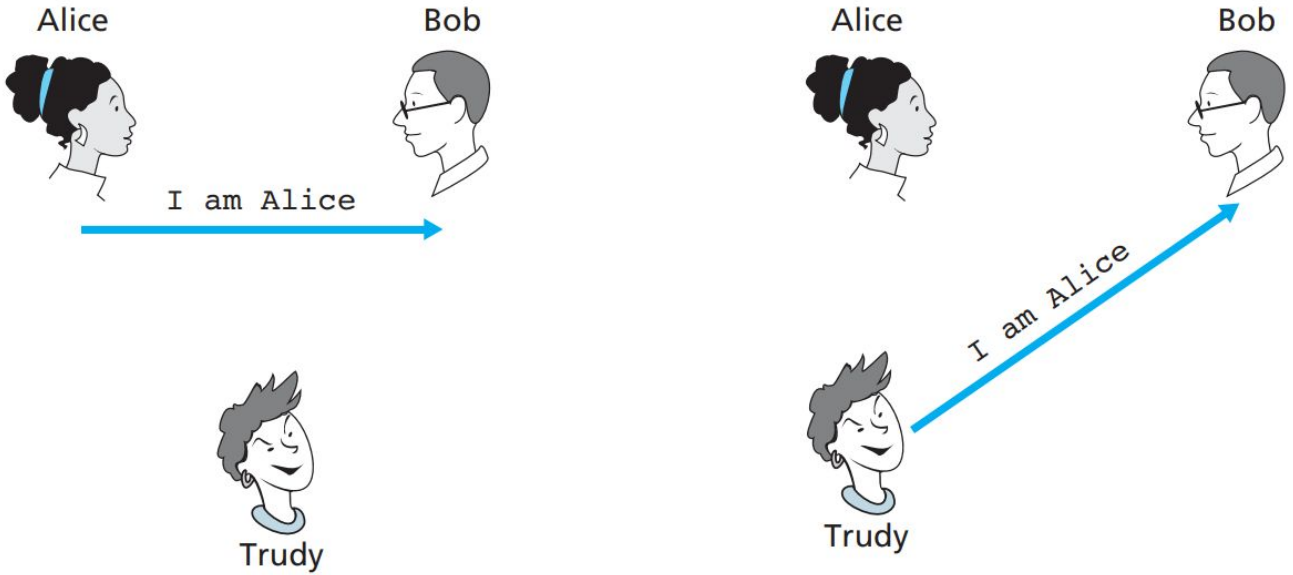


Authentication

Authentication

Goal: Bob wants Alice to “prove” her identity to him

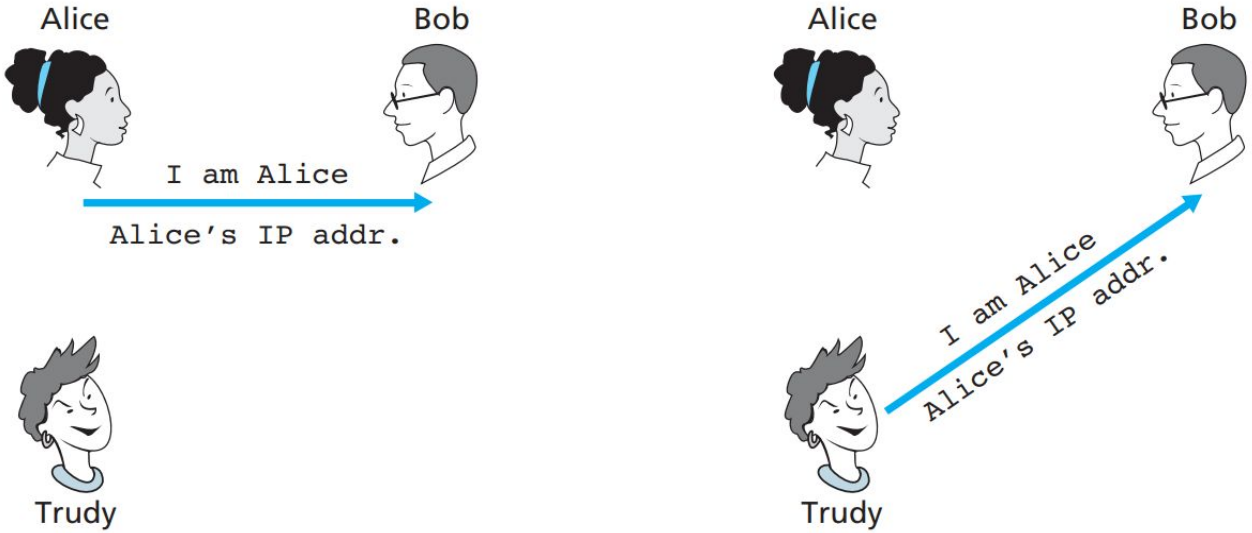
Protocol ap1.0: Alice says “I am Alice”



Authentication

Goal: Bob wants Alice to “prove” her identity to him

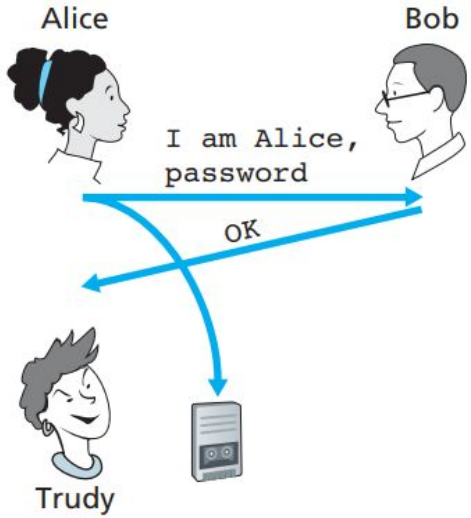
Protocol ap2.0: Alice says “I am Alice” in an IP packet containing her source IP address

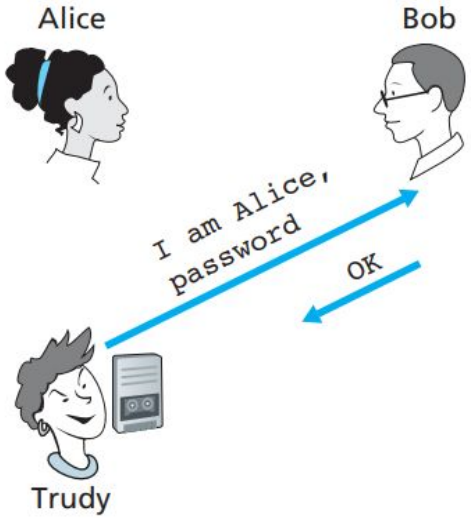


Authentication

Goal: Bob wants Alice to “prove” her identity to him

Protocol ap3.0: Alice says “I am Alice” and sends her encrypted secret password to “prove” it.





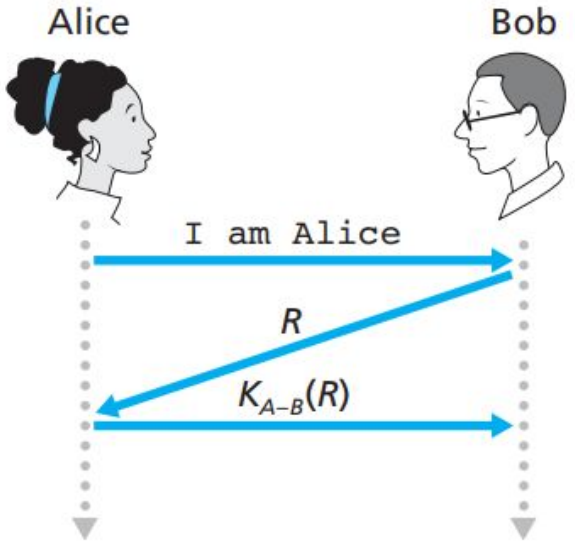
Authentication: symmetric key

Goal: avoid playback attack

nonce: number (R) used only once-in-a-lifetime

Protocol ap4.0: to prove Alice “live”, Bob sends Alice nonce, R

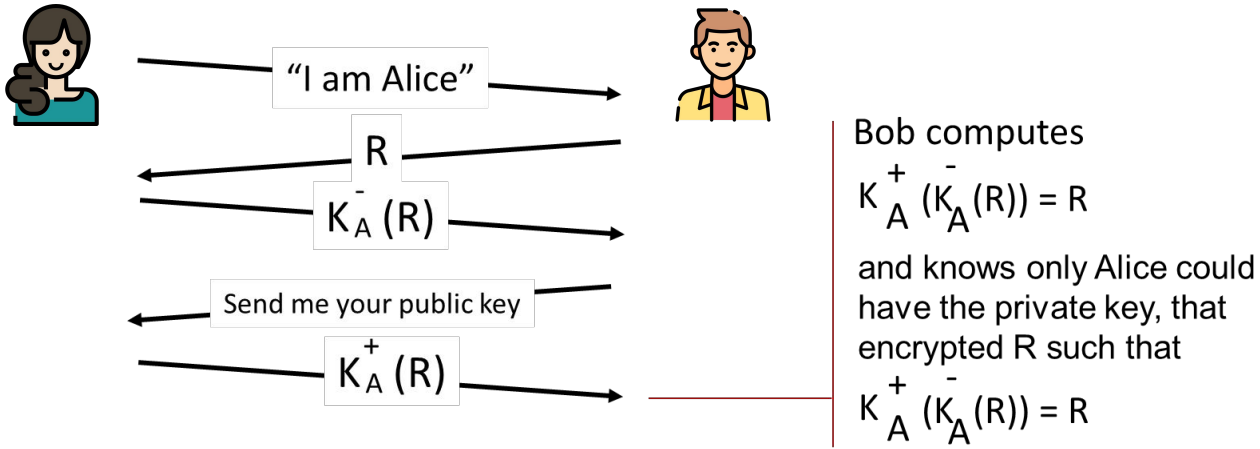
- Alice must return R, encrypted with shared secret key



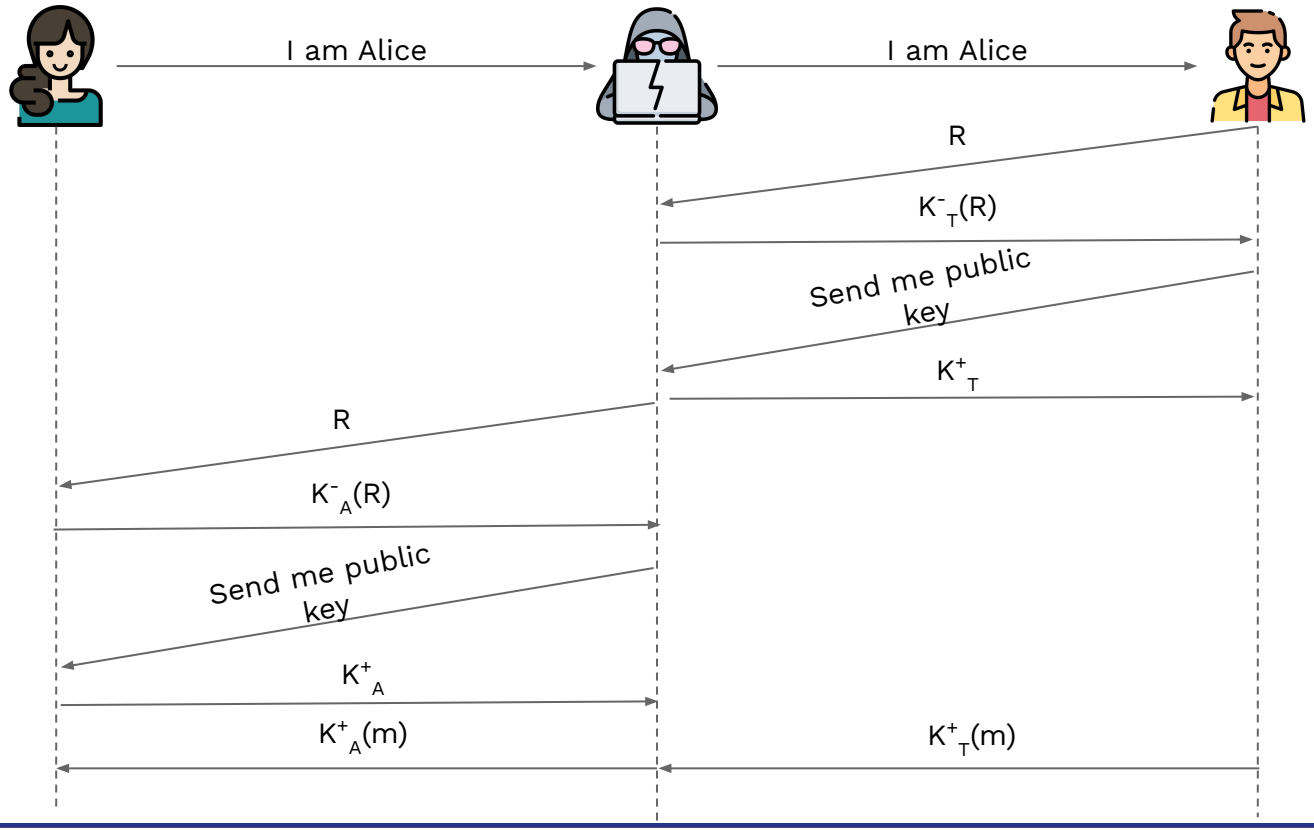
Authentication: public key

Can we authenticate using public key techniques?

ap5.0: use nonce, public key cryptography



Man in the middle



Digital Certificate

Certification Authority (CA)

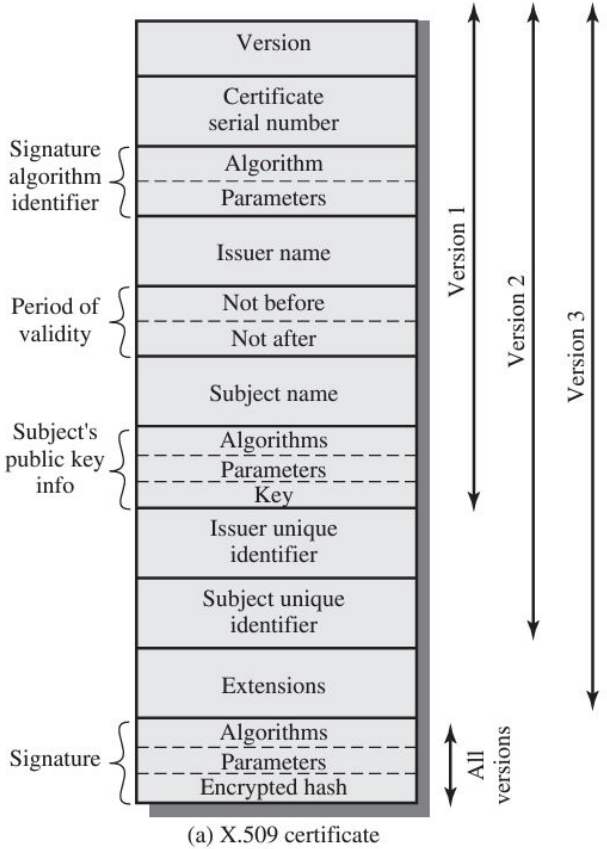
- Who guarantees that Bob's public key, which we obtain from a public registry, was released to Bob?
- A trusted third party: the **certification authority (CA)**, which certifies the user/public key link using a specific digital certificate

Certificate & Digital Certificate

- Physical certificate
  - Identity card, etc.
- Issued by a recognized authority
- Associate a **person's identity** (name, surname, date of birth, ...) with their **physical appearance** (photo)

- Electronic
  - Associates a **person's identity** with a **public key**
- Issued by a recognized **CA**
- Signed with the CA's private key
- Typical format: **X.509**
- Recommended by the ITU (International Telecommunication Union)

X.509 structure



(a) X.509 certificate

Figure 4.4 X.509 Formats

Certificate usage

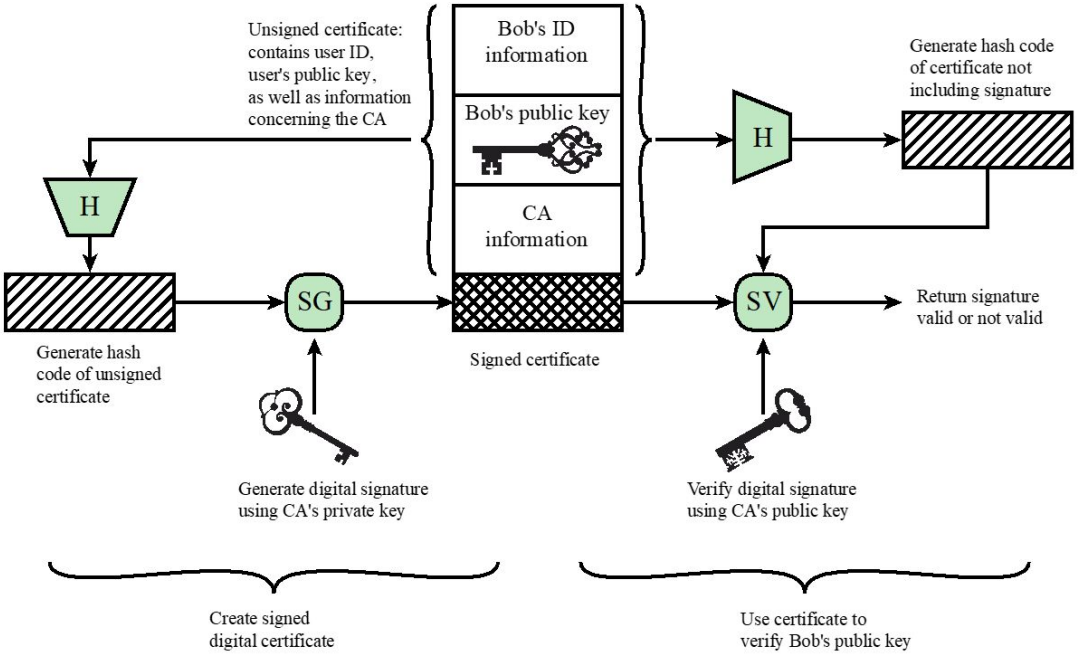


Figure 2.8 Public-Key Certificate Use

Obtain a Digital Certificate

- The user generates a **pair of keys** on his PC
  - Common browsers offer the service (Netscape, Explorer)
  - The private key is **stored locally** in a hidden file (or floppy disk)
  - Greater security: generate the key pair via SmartCard connected to the PC - the private key never leaves the SmartCard (protected by PIN)
- The user sends a **certificate signing request** (**CSR** or certification request) to the CA, along with the generated public key (unless the CA generates the key pair for the user)
- The request must be **self-signed** using the **applicant's private key**, which provides **proof-of-possession** of the private key

Certificate signing request (CSR)

DN[2]	Information	Description	Sample
`CN`	Common Name	This is fully qualified domain name that you wish to secure	*.wikipedia.org
`O`	Organization Name	Usually the legal name of a company or entity and should include any suffixes such as Ltd., Inc., or Corp.	Wikimedia Foundation, Inc.
`OU`	Organizational Unit	Internal organization department/division name	IT
`L`	Locality	Town, city, village, etc. name	San Francisco
`ST`	State	Province, region, county or state. This should not be abbreviated (e.g. West Sussex, Normandy, New Jersey).	California
`C`	Country	The two-letter ISO code for the country where your organization is located	US
`EMAIL`	Email Address	The organization contact, usually of the certificate administrator or IT department	

Obtain a Digital Certificate

- The CA authenticates the applicant, usually asking him to go in person to an **LVP (Local Validation Point)** desk connected to the CA
- Once the identity has been verified, the CA issues the certificate, sends it to the applicant via email and inserts the certified key into the public key register

PKI (Public Key Infrastructure)

- Minimum structure: CA+LVP. Multiple LVPs allowed
  - **LVP** is the one-stop desk for classic user authentication; **LVPO** is the operator
- Hierarchical structure: some CAs certify others, obtaining a "chain of trust"
  - Tree structure
  - The Root CA certifies top-level CAs
  - The first level CAs certify the second level CAs
  - The last level CAs certify the individual user

PKI (Public Key Infrastructure)

- **End entity:** A generic term used to denote end users, devices… identified by certificate.
- **Certification authority (CA):** The issuer of certificates and (usually) certificate revocation lists (CRLs).
- **Registration authority (RA):** An optional component that can assume a number of administrative functions from the CA.
- **CRL issuer:** An optional component that a CA can delegate to publish CRLs.
- **Repository:** A generic term used to denote any method for storing certificates and CRLs.

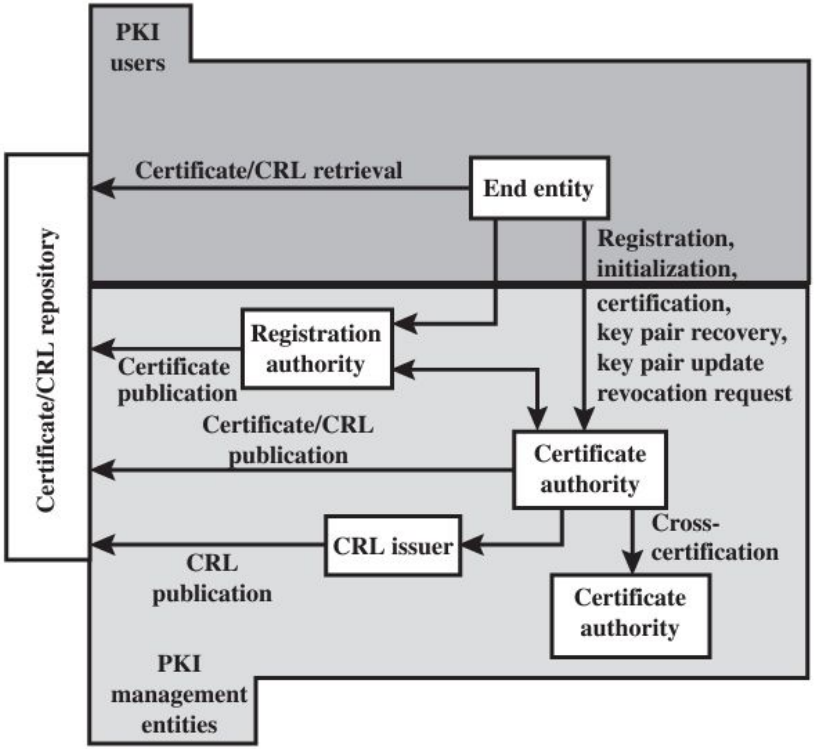


Figure 4.7 PKIX Architectural Model

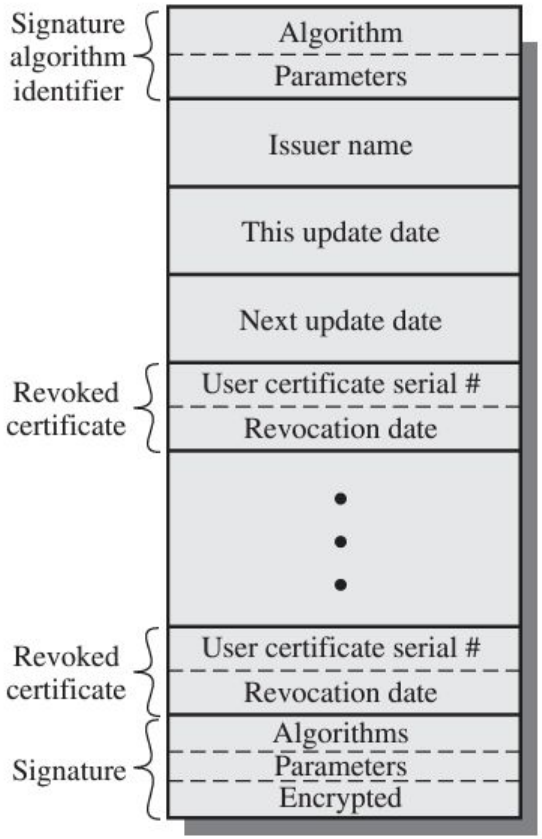
Certificate revocation

- Various reasons
  - Change of personal data (email, contact details, etc)
  - Dismissal, resignation
  - Private key compromise…
- Request for revocation (termination of validity)
  - By the user
  - From emitter (LVPO)
- Revocation via **CRL (Certificate Revocation List)**

CRL (Certificate Revocation List)

- List of certificates revoked before their natural expiration
- Digitally signed by the same CA that issued the now revoked certificate
- An LVPO issues a **CRR (Certificate Revocation Request)** for 1 particular certificate
- The relevant CA will issue the new CRL

CRL structure

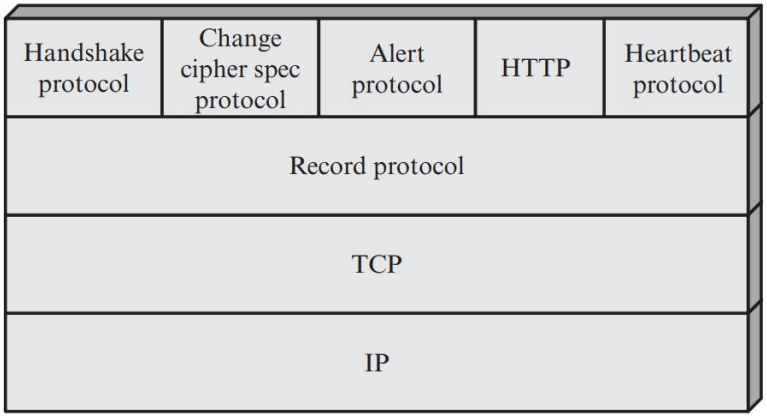


(b) Certificate revocation list

Transport-layer security (TLS)

Transport-layer security (TLS)

- TLS is an **Internet standard** that evolved from a commercial protocol known as **Secure Sockets Layer (SSL).** Widely deployed security protocol above the transport layer (TCP)
  - supported by almost all browsers, web servers: https (port 443)
- provides:
  - **confidentiality**: via symmetric encryption
  - **integrity**: via cryptographic hashing
  - **authentication**: via public key cryptography



TLS Version

SSL and TLS protocols

Protocol	Published	Status
SSL 1.0	Unpublished	Unpublished
SSL 2.0	1995	Deprecated in 2011 (RFC 6176)
SSL 3.0	1996	Deprecated in 2015 (RFC 7568)
TLS 1.0	1999	Deprecated in 2021 (RFC 8996)[20][21][22]
TLS 1.1	2006	Deprecated in 2021 (RFC 8996)[20][21][22]
TLS 1.2	2008	In use since 2008[23][24]
TLS 1.3	2018	In use since 2018[24][25]

TLS ARCHITECTURE

TLS Connection

- A **transport** that provides a suitable **type of service**
- For TLS such connections are **peer-to-peer relationships**
- Connections are **transient**
- Every connection is associated with **one session**

TLS Session

- An **association** between a **client** and a **server**
- Created by the **Handshake Protocol**
- Define a **set of cryptographic security parameters** which can be shared among multiple connections
- Are used to **avoid** the **expensive negotiation** of new security parameters for each connection

Session state 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
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

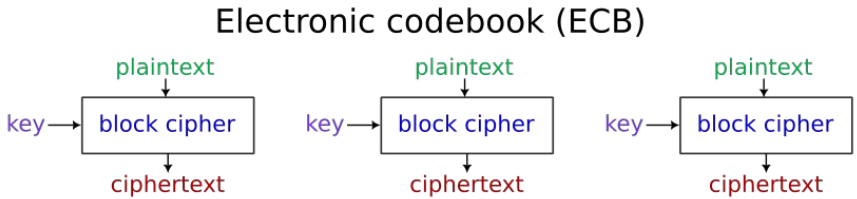
Connection state parameters

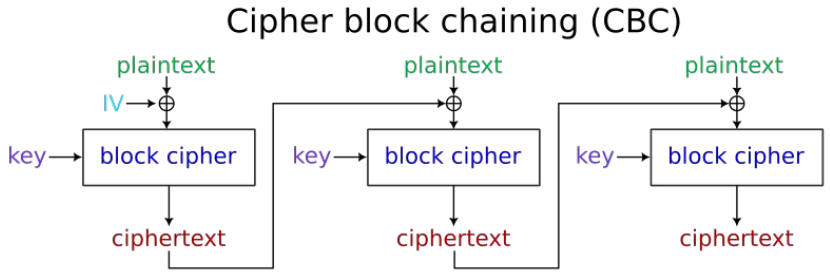
Server and client random	Byte sequences that are chosen by the server and client for each connection
Server writes MAC secret	The secret key used in MAC operations on data sent by the server
Client writes MAC secret	The secret key used in MAC operations on data sent by the client
Server writes key	The secret encryption key for data encrypted by the server and decrypted by the client
Client writes 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
Sequence numbers	Each party maintains separate sequence numbers for transmitted and received messages for each connection

Cipher block chaining (CBC)

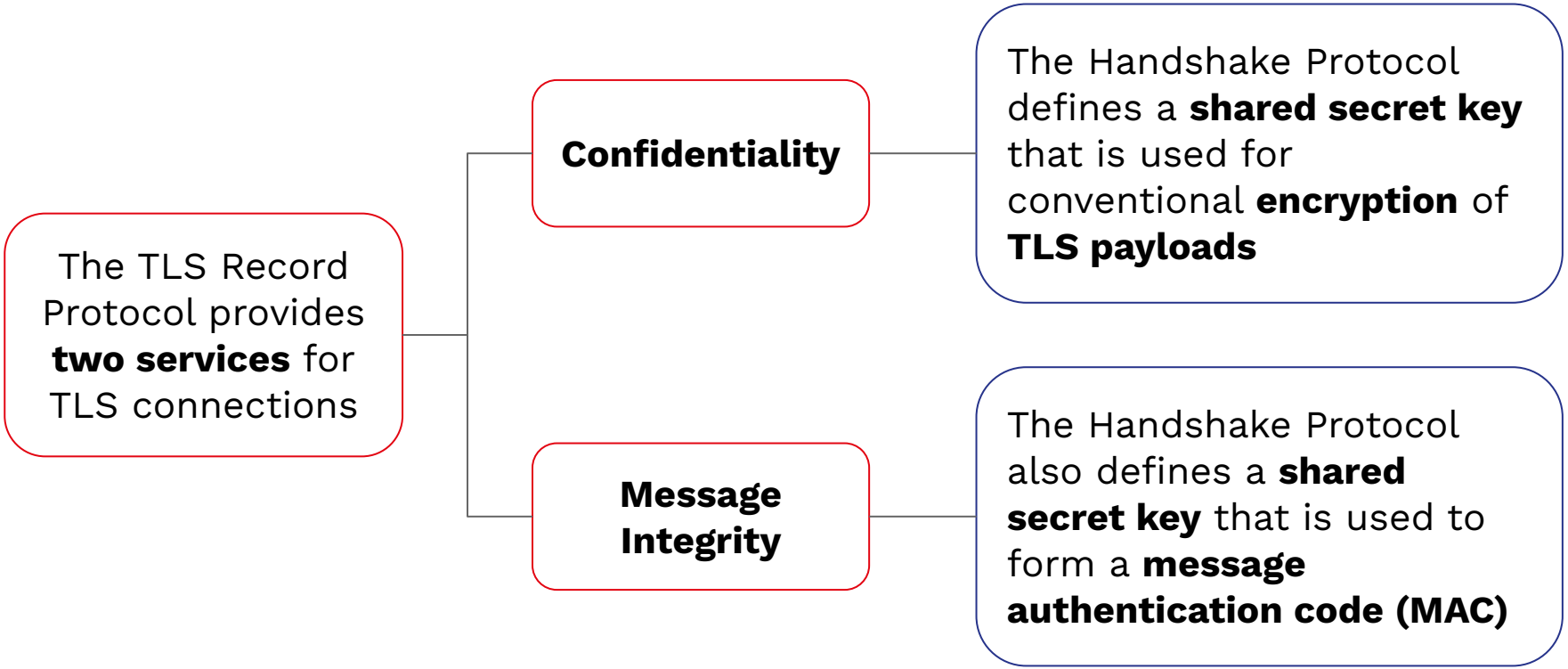
An **initialization vector (IV)** is a block of bits that is used by several modes to **randomize the encryption** and hence to produce **distinct ciphertexts** even if the **same plaintext** is encrypted multiple times.

- Each block of plaintext is **XORed** with the **previous ciphertext block** before being encrypted.
- This way, each ciphertext block depends on all plaintext blocks processed up to that point.
- To make each **message unique**, an **initialization vector** must be used in the first block.





TLS Record Protocol



TLS Record Protocol Operation

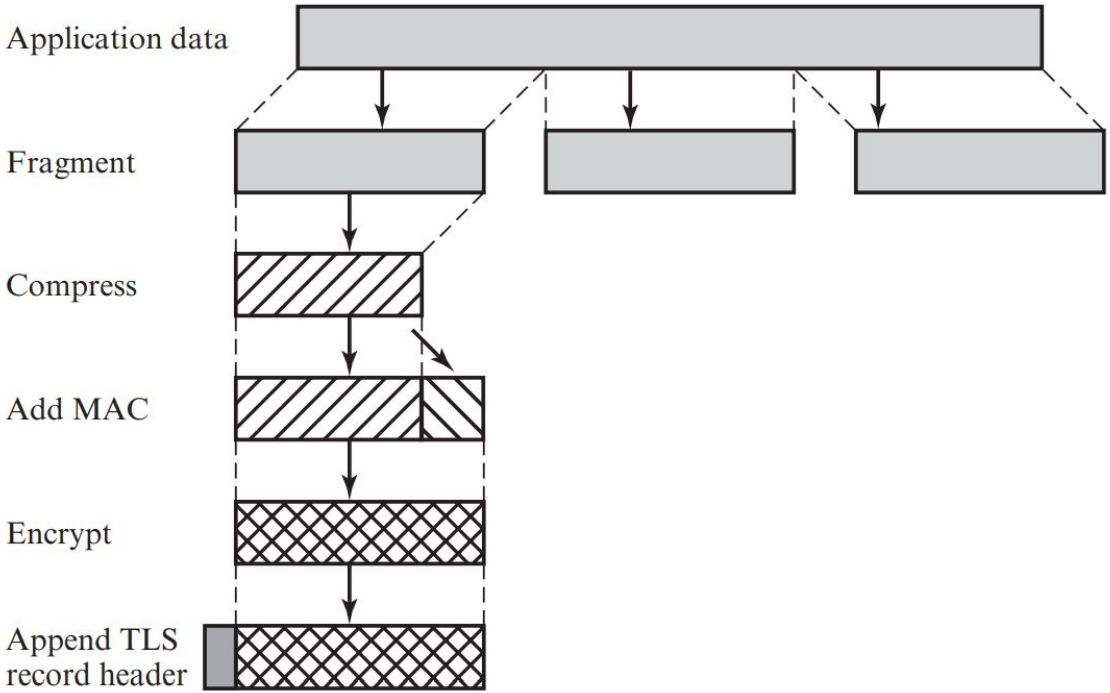
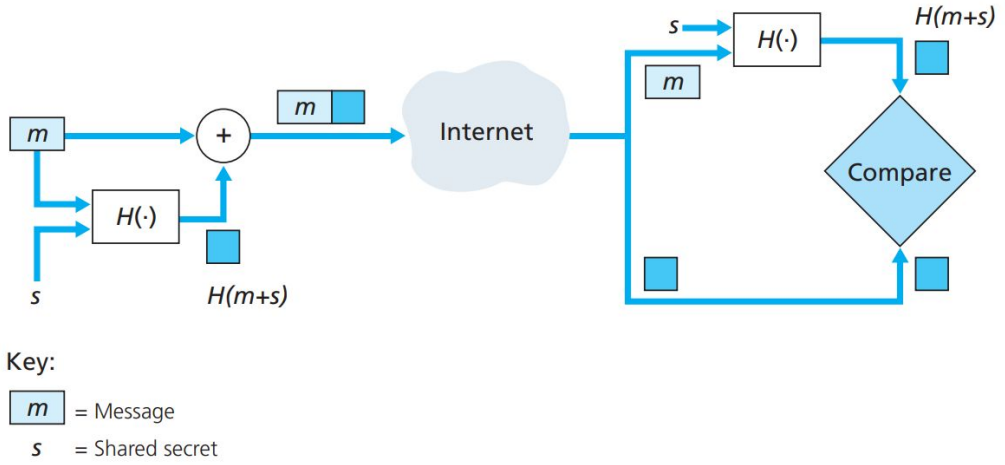


Figure 6.3 TLS Record Protocol Operation

Message Authentication Code (MAC)



- Alice creates message **m**, concatenates **s** with **m** to create **m** + **s**, and calculates the **hash *H(m + s)***. ***H(m + s)*** is called the **message authentication code (MAC)**
- Alice creates an extended message **(m, *H(m + s)*)** and sends it to Bob
- Bob the message **(m, h)** and knowing **s**, calculates the MAC ***H(m + s)***. If ***H(m + s)*** = **h**, Bob concludes that everything is fine.

TLS Record Protocol: Record Header

- **Content Type (8 bits)**: the higher-layer protocol used to process the enclosed fragment.
- **Major Version (8 bits)**: indicate the major version of TLS in use. For TLSv2, the value is 3.
- **Minor Version (8 bits):** indicate minor version in use. For TLSv2, the value is 1.
- **Compressed Length (16 bits)**: the length in bytes of the plaintext fragment (or compressed). Maximum value 2^{14} + 2048.

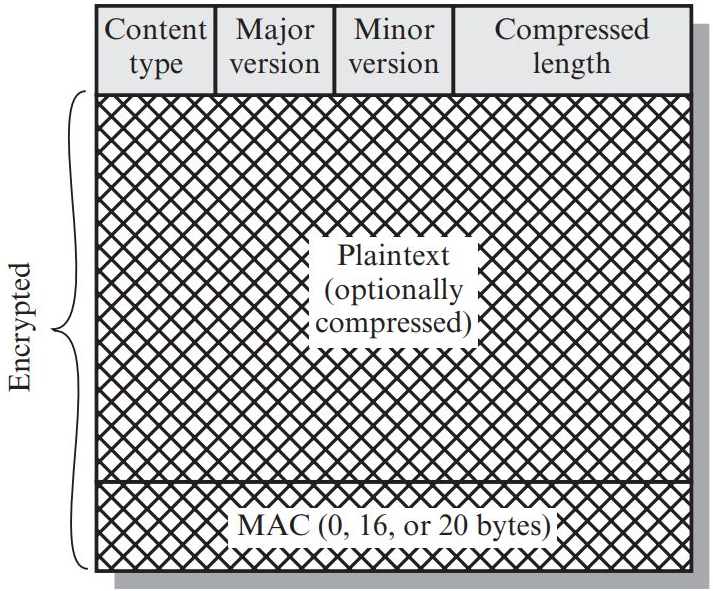


Figure 6.4 TLS Record Format

Content types: Change Cipher and Alert

- **change_cipher_spec**: TLS specific protocol, it consists of a **single byte message** with value 1. The sole purpose is to **update** the **cipher suite** to be used on this connection.
- **alert**: is used to convey **TLS-related alerts** to the peer entity. Each message in this protocol consists of **two bytes.** The first byte takes the value **warning (1)** or **fatal (2)** to convey the severity of the message. The **second byte** contains a code that indicates the **specific alert**.

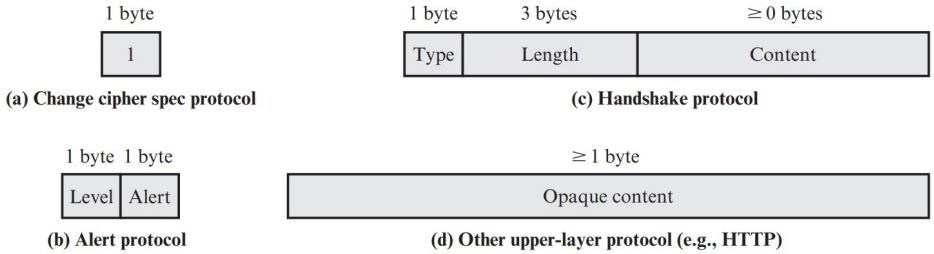


Figure 6.5 TLS Record Protocol Payload

Content types: Handshake Protocol

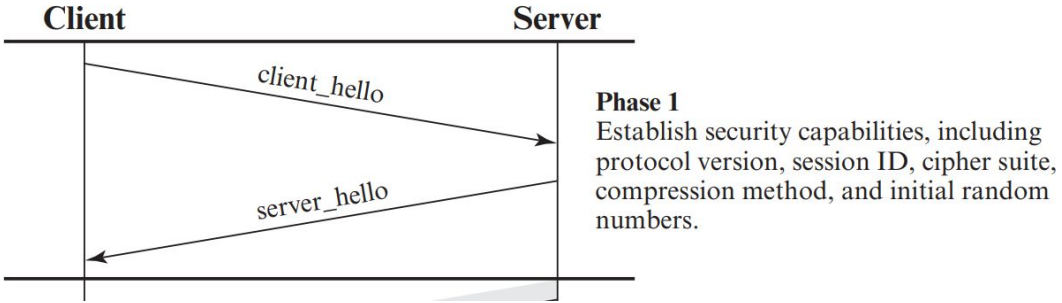
Table 6.2 TLS Handshake Protocol Message Types

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

The most complex part of TLS is the **Handshake Protocol** . This protocol allows the server and client to **authenticate** each other and to **negotiate an encryption** and **MAC algorithm** and **cryptographic keys** to be used to protect data sent in a TLS record. Each message has 3 field:

- **Type (1 byte)**: Indicates one of 10 messages.
- **Length (3 bytes)**: The length of the message in bytes
- **Content (≥ 0 bytes)**: The parameters associated with this message

Handshake Protocol: Phase 1



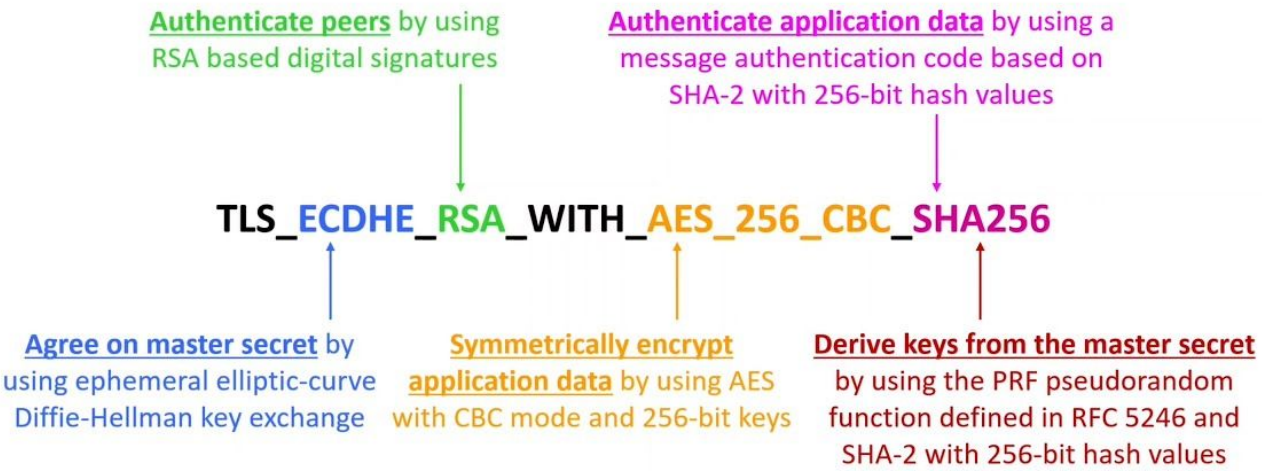
client_hello:

- The **TLS versions** supported by the client.
- 32 bytes of random data **(nonce)** generated by the client.
- A **session ID** created by the client.
- A list of **supported ciphers** by the client.
- A list of **supported compression methods** by the client.

server_hello:

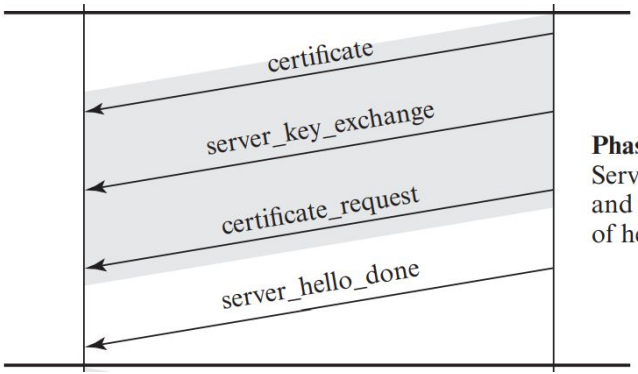
- version: client's lowest and server's highest version supported
- server's random field
- session ID
- CipherSuite/Compression method selected

Handshake Protocol: cipher suites



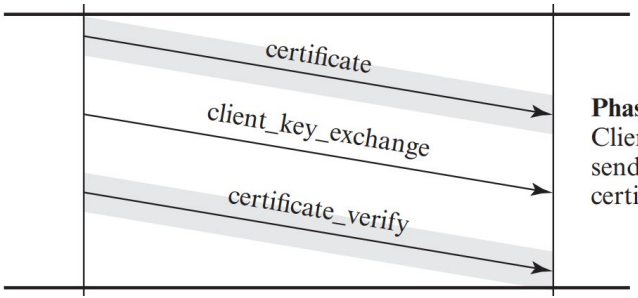
Handshake Protocol: Phase 2

- **certificate:** server sends its certificate if it needs to be authenticated; the message contains one or a chain of X.509 certificates.
- **server_key_exchange:** may be sent if it is required (depends on the cipher suite).
- **certificate_request:** server can request a certificate from the client.
- **server_done:** sent by the server to indicate the end of the server hello and associated messages



Phase 2
Server may send certificate, key exchange, and request certificate. Server signals end of hello message phase.

Handshake Protocol: Phase 3



Phase 3
Client sends certificate if requested. Client sends key exchange. Client may send certificate verification.

- **certificate:** if server requested a certificate, clients sends its certificate.
- **client_key_exchange:** client generates 48 bytes of random data (**pre-master secret**) and encrypts them with server public key.
- **certificate_verify message:** verification of client certificate if sent before. Client hashes all messages sent from client_hello and signs them with its private key – the server validates its signature.

Pre Master Secret & Master Secret

Pre-Master Secret

- The client generates a **48-bit pre-master secret** and sends it to the server mainly to compute master secrets.
- Encrypts it with the server public key and sends it to the server.

Master Secret

- Server decrypts it with its private key.
- The client and server use the **master secret** to generate the session key.
- The client and the server use a **Pseudo-Random Function (PRF)** to calculate the master secret key

```
master_secret = PRF(pre-master_secret, "Master Secret", ClientHello.random + ServerHello.random)
[0..47];
```

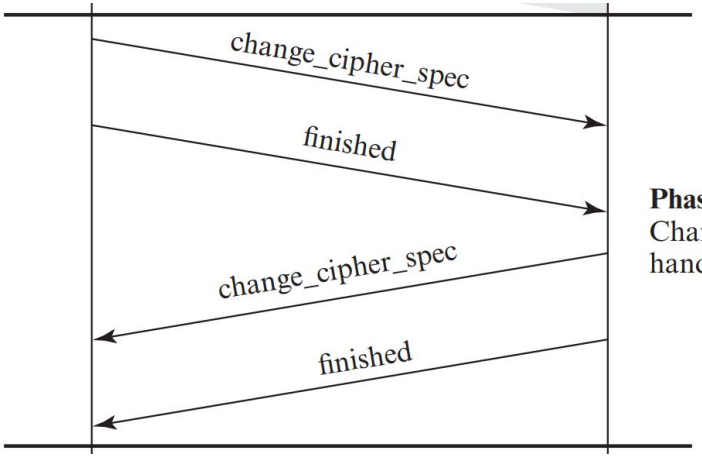
Key Derivation

- From the MS client and server generate 4 keys for encryption and message integrity:
  - **E_c** = session encryption key for data sent from client to server;
  - **M_c** = session MAC key for data sent from client to server;
  - **E_s** = session encryption for data sent from server to client;
  - **M_s** = session MAC key for data sent from server to client.
- In order to prevent a replay attack and a man-in-the-middle attack, server and client use a **sequence number** which is a counter for every TLS record sent by the server/client.

MAC= H(m, MAC key (M_c or M_s), sequence number)

Handshake Protocol: Phase 4

- **change_cipher_spec (client):** client copies the pending CipherSpec into the current CipherSpec indicating that future communications will be handled with these ciphers and parameters. Then it sends a finished message under the new algorithms, keys and secrets.
- **change_cipher_spec (server):** server does the same thing.



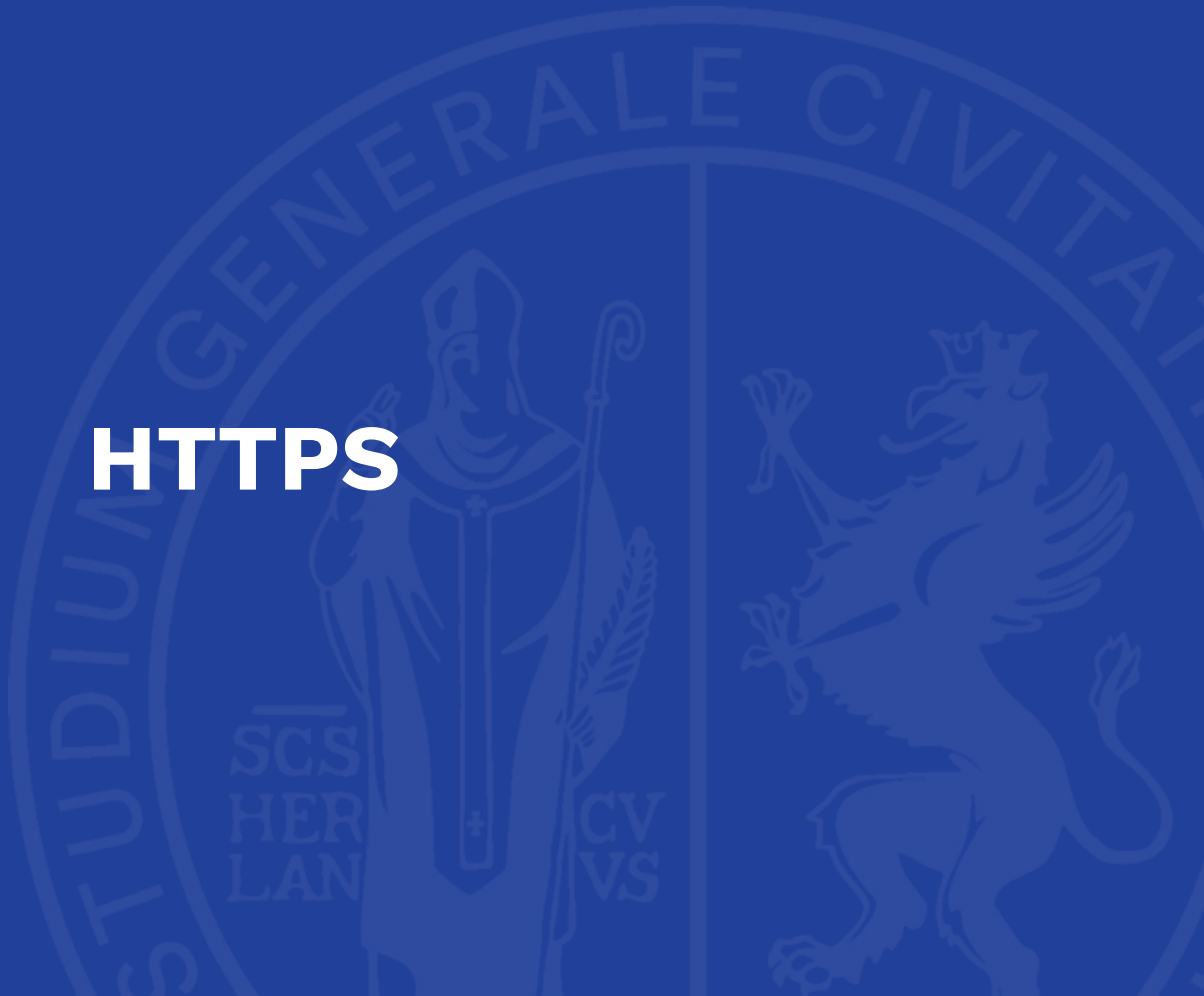
Phase 4
Change cipher suite and finish handshake protocol.

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

Check your website

HTTPS



HTTPS (HTTP over SSL)

- Refers to the **combination of HTTP and SSL** to implement secure communication between a Web browser and a Web server
- The **HTTPS** capability is **built** into all modern **Web browsers**
- A user of a Web browser will see URL addresses that begin with https:// rather than http://
- If HTTPS is specified, port **443** is used, which invokes SSL
- Documented in RFC 2818, HTTP Over TLS
- There is no fundamental change in using HTTP over either SSL or TLS and both implementations are referred to as HTTPS
- When HTTPS is used, the following elements of the communication are encrypted:
    - URL of the requested document
    - Contents of the document
    - Contents of browser forms
    - Cookies sent from browser to server and from server to browser
    - Contents of HTTP header

Connection Initiation

For HTTPS, the agent acting as the **HTTP client** also acts as the **TLS client**

- The client initiates a connection to the server on the appropriate port and then sends the **TLS ClientHello** to begin the TLS handshake
- When the **TLS handshake** has **finished**, the client may then initiate the first **HTTP request**
- All **HTTP data** is to be sent as **TLS application data**

There are three levels of awareness of a connection in HTTPS:

- At the HTTP level, an HTTP client requests a connection to an HTTP server by sending a connection request to the next lowest layer
  - Typically the next lowest layer is TCP, but it may also be TLS/SSL
- At the level of TLS, a session is established between a TLS client and a TLS server
  - This session can support one or more connections at any time

Connection closure

- An HTTP client or server can indicate the closing of a connection by including the line **Connection: close in an HTTP record**
- 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
- **TLS implementations** must initiate an exchange of **closure alerts** before closing a connection
- A TLS implementation may, after sending a closure alert, close the connection without waiting for the peer to send its closure alert, generating an “incomplete close”
- An **unannounced TCP closure** could be **evidence** of some **sort of attack** so the HTTPS client should issue some sort of security warning when this occurs

References

- Kurose, J. F., & Ross, K. W. (2021). *Computer networking : a top-down approach* (8th ed.). Pearson.
- Stallings, W. (2017). *Network security essentials : applications and standards*. Pearson Education, Inc.
- Gössi, C. (2023, March 14). *TLS Essentials 10: TLS cipher suites explained*. Youtube Video. https://www.youtube.com/watch?app=desktop&v=mFdDap9A9-Q&ab_channel=CyrillG%C3%B6ssi
- Rescorla, E., & Dierks, T. (2008, August). *The Transport Layer Security (TLS) Protocol Version 1.2*. IETF. https://datatracker.ietf.org/doc/html/rfc5246
- Rescorla, E. (2018, August). *The Transport Layer Security (TLS) Protocol Version 1.3*. IETF. https://datatracker.ietf.org/doc/html/rfc8446

References

- Wikipedia contributors. (2024, September 16). Certificate signing request. In Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/w/index.php?title=Certificate_signing_request&oldid=1245977820
- Wikipedia contributors. (2024, September 20). Block cipher mode of operation. In Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/w/index.php?title=Block_cipher_mode_of_operation&oldid=1246703154