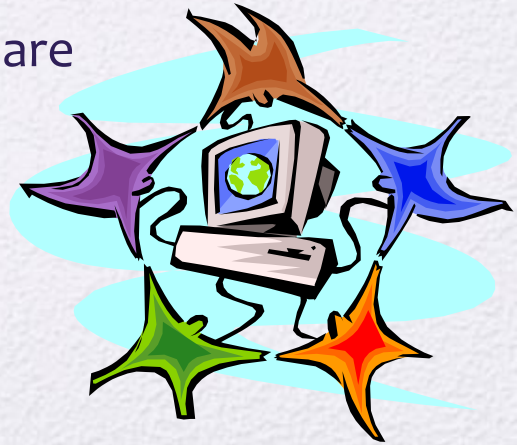
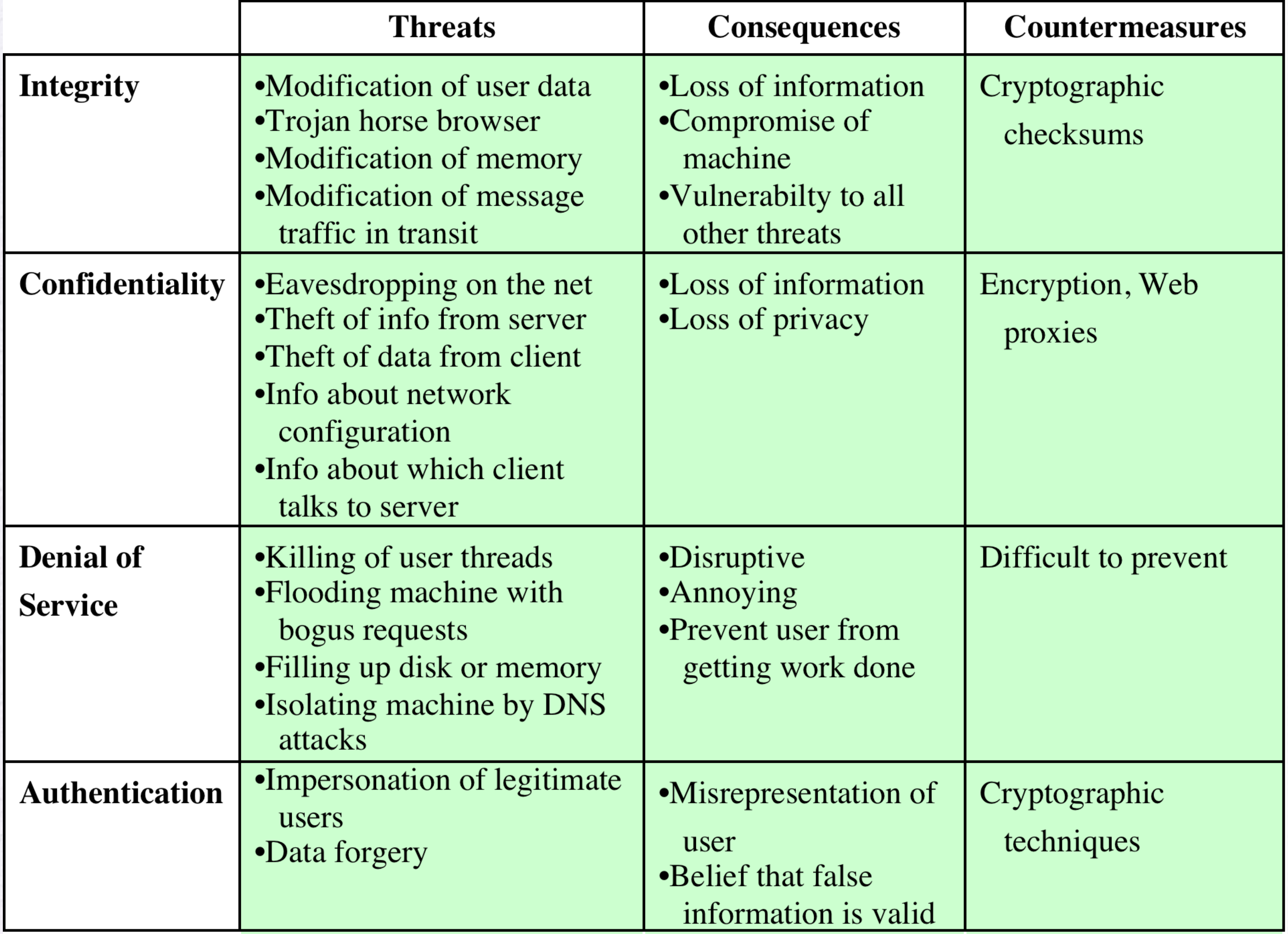
Web Security Considerations

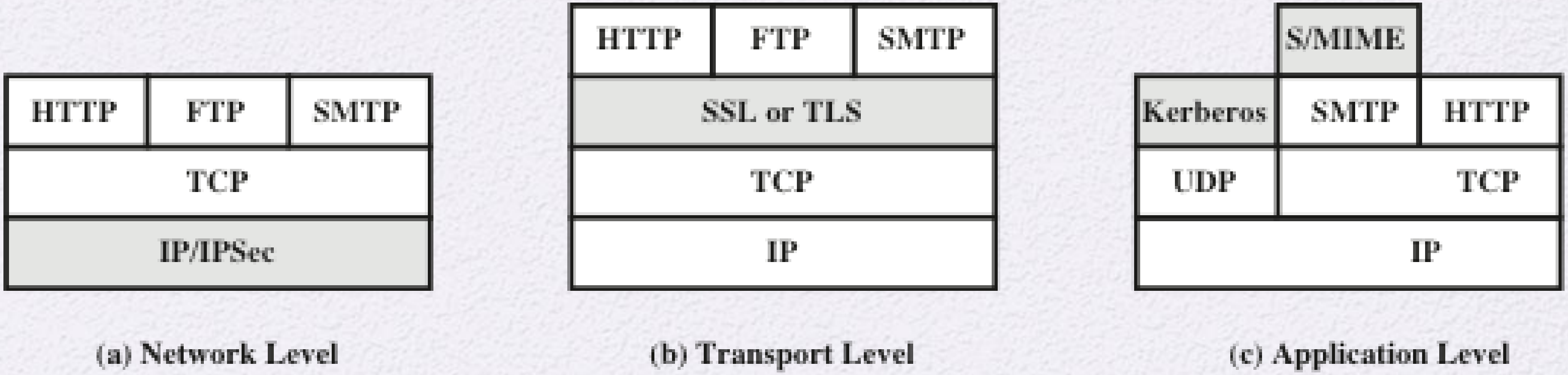
- The World Wide Web is fundamentally a client/server application running over the Internet and TCP/IP intranets
- The following characteristics of Web usage suggest the need for tailored security tools:
  - Web servers are relatively easy to configure and manage
  - Web content is increasingly easy to develop
  - The underlying software is extraordinarily complex
    - May hide many potential security flaws
  - A Web server can be exploited as a launching pad into the corporation's or agency's entire computer complex
  - Casual and untrained (in security matters) users are common clients for Web-based services
    - Such users are not necessarily aware of the security risks that exist and do not have the tools or knowledge to take effective countermeasures





	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

Table 17.1 A Comparison of Threats on the Web



(a) Network Level

(b) Transport Level

(c) Application Level

Figure 17.1 Relative Location of Security Facilities in the TCP/IP Protocol Stack

Transport Layer Security (TLS)

One of the most widely used security services

Defined in RFC 5246

Is an Internet standard that evolved from a commercial protocol known as Secure Sockets Layer (SSL)

Is a general purpose service implemented as a set of protocols that rely on TCP

Could be provided as part of the underlying protocol suite and therefore be transparent to applications

Can be embedded in specific packages

Most browsers come equipped with TLS, and most Web servers have implemented the protocol

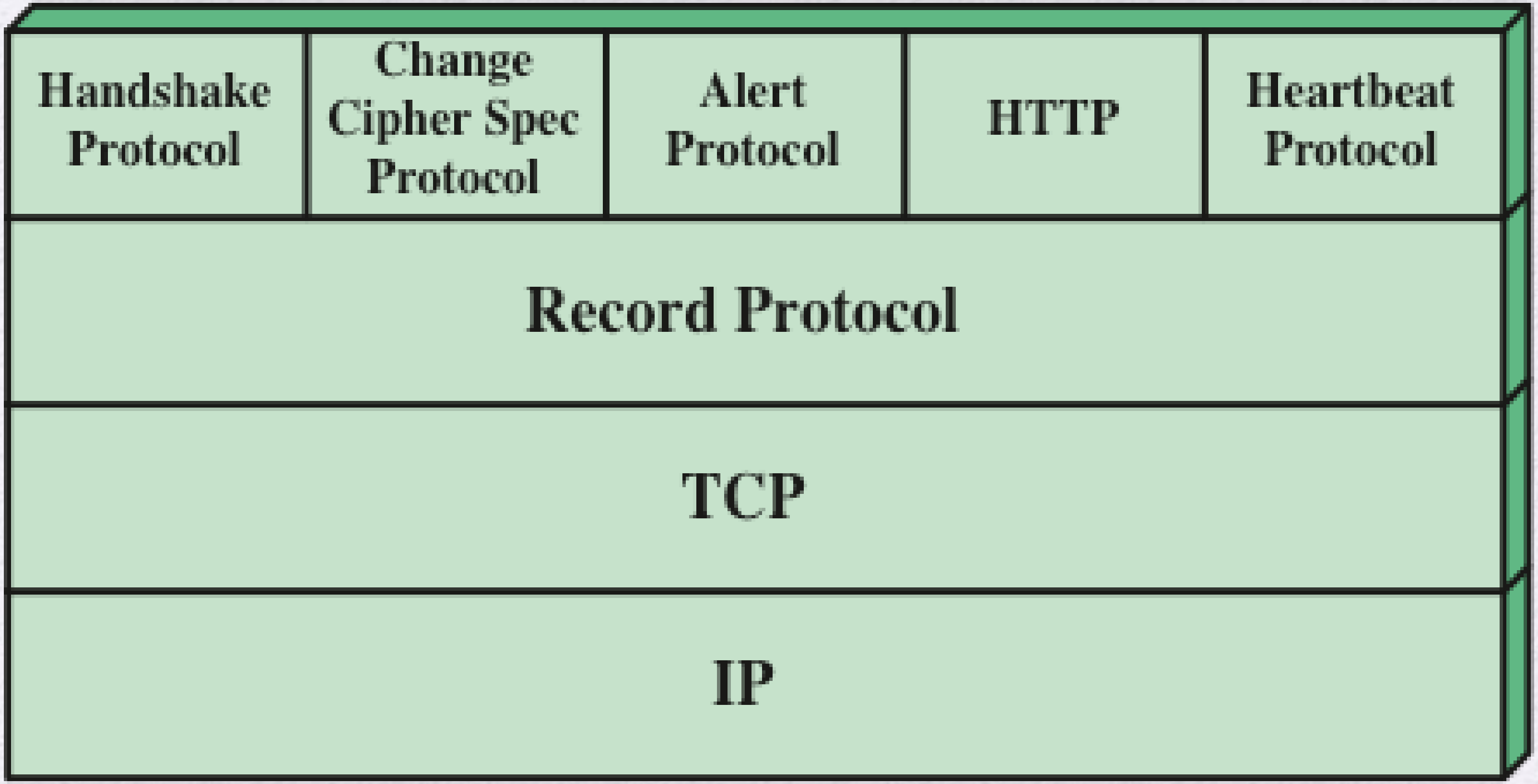
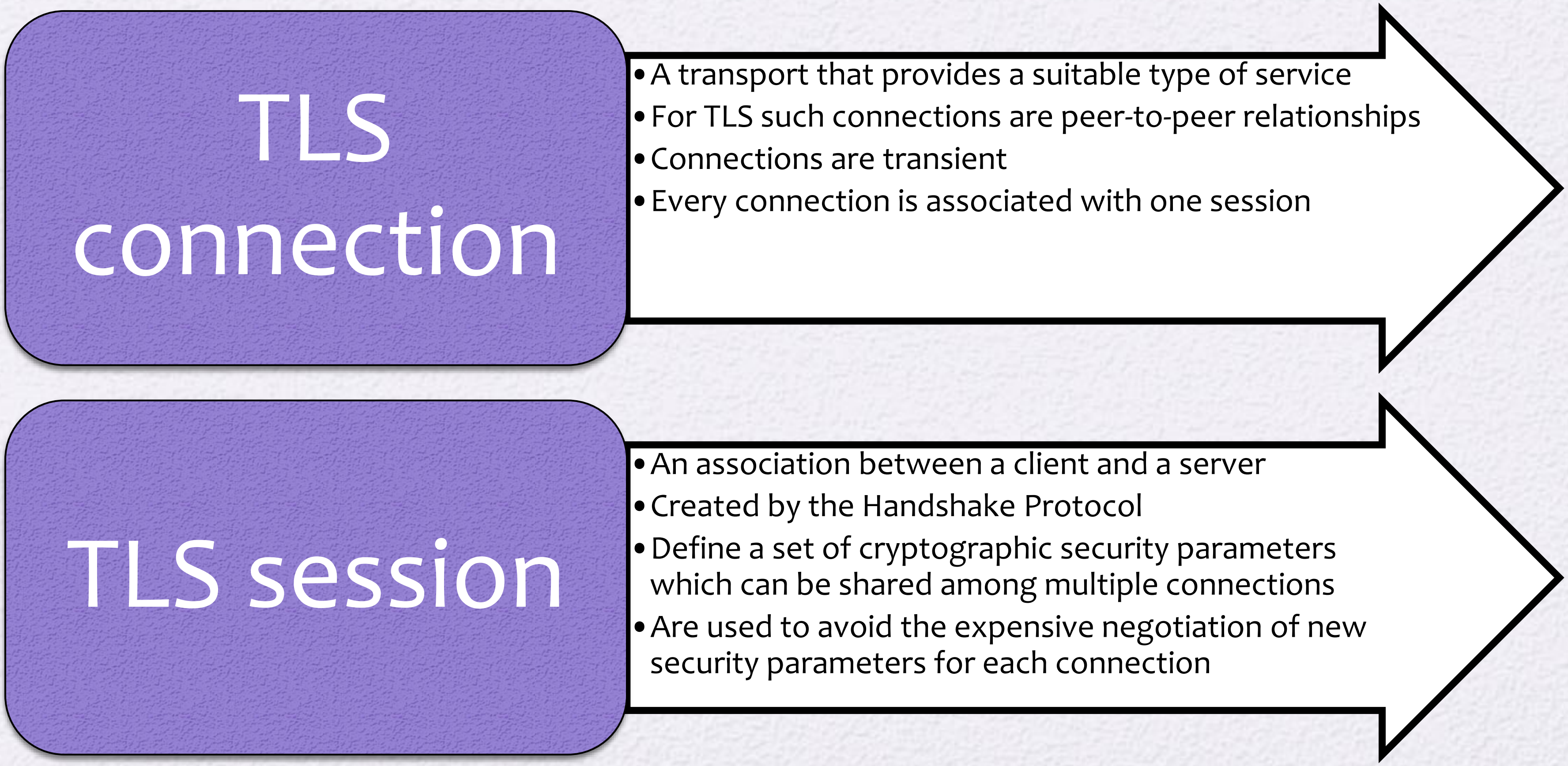


Figure 17.2 SSL/TLS Protocol Stack

TLS Architecture

- Two important TLS concepts are:



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

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

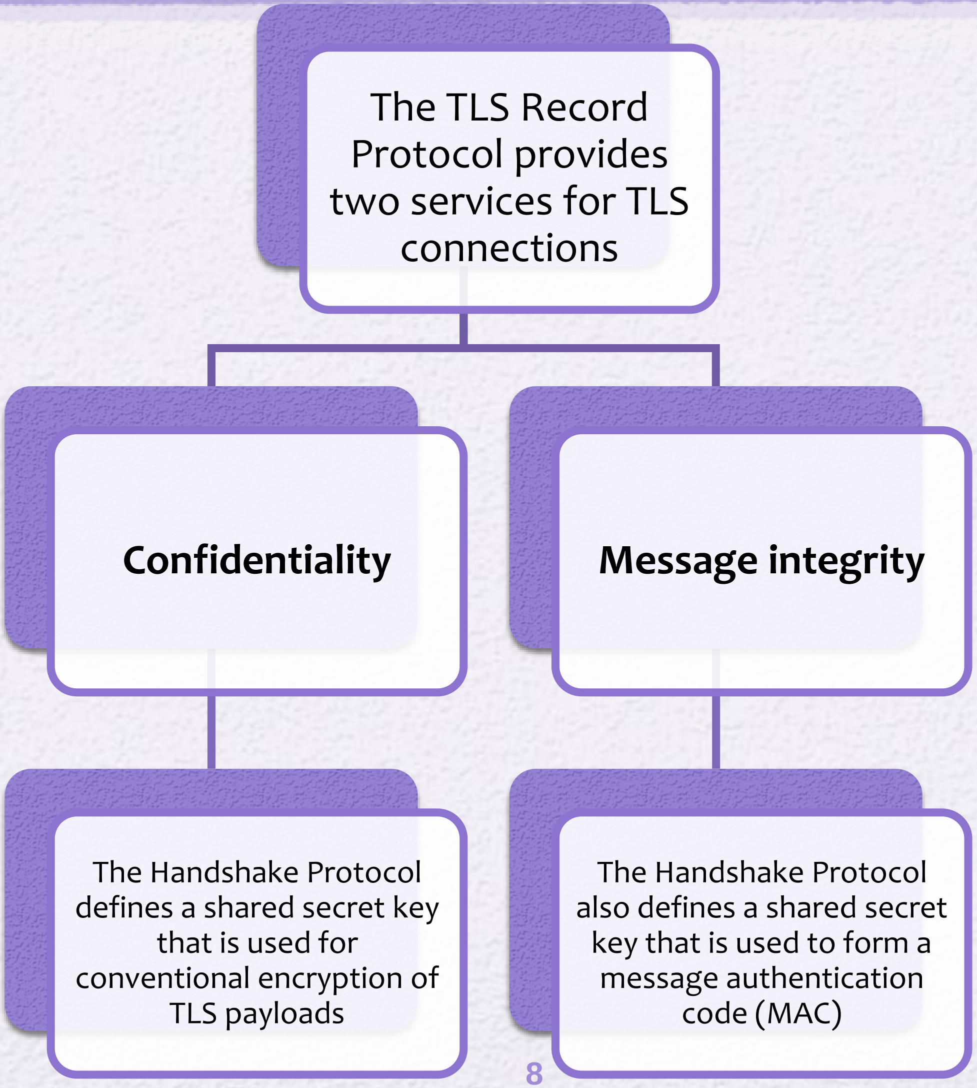
Master secret

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

TLS Record Protocol



The TLS Record Protocol provides two services for TLS connections

Confidentiality

The Handshake Protocol defines a shared secret key that is used for conventional encryption of TLS payloads

Message integrity

The Handshake Protocol also defines a shared secret key that is used to form a message authentication code (MAC)

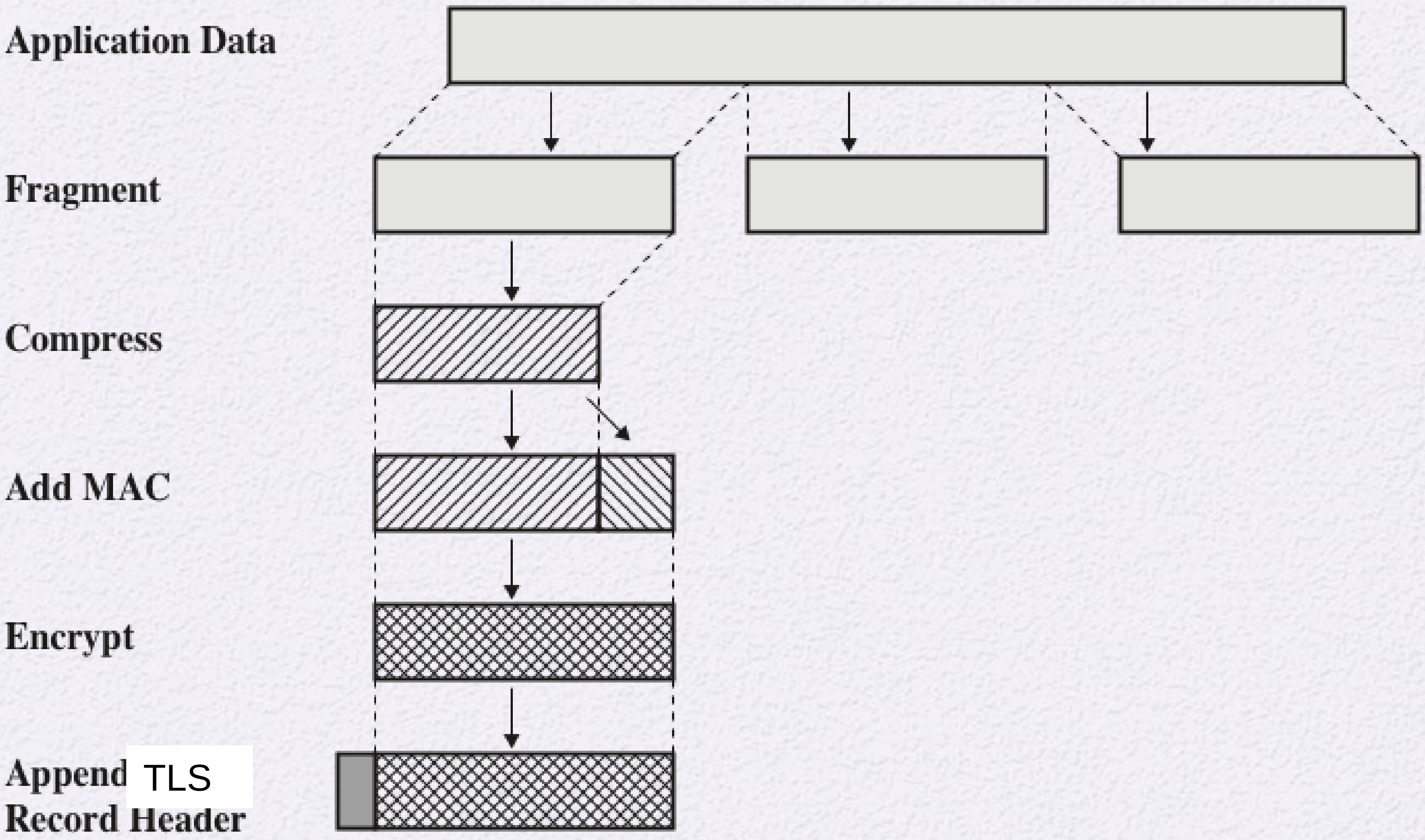


Figure 17.3 TLS Record Protocol Operation

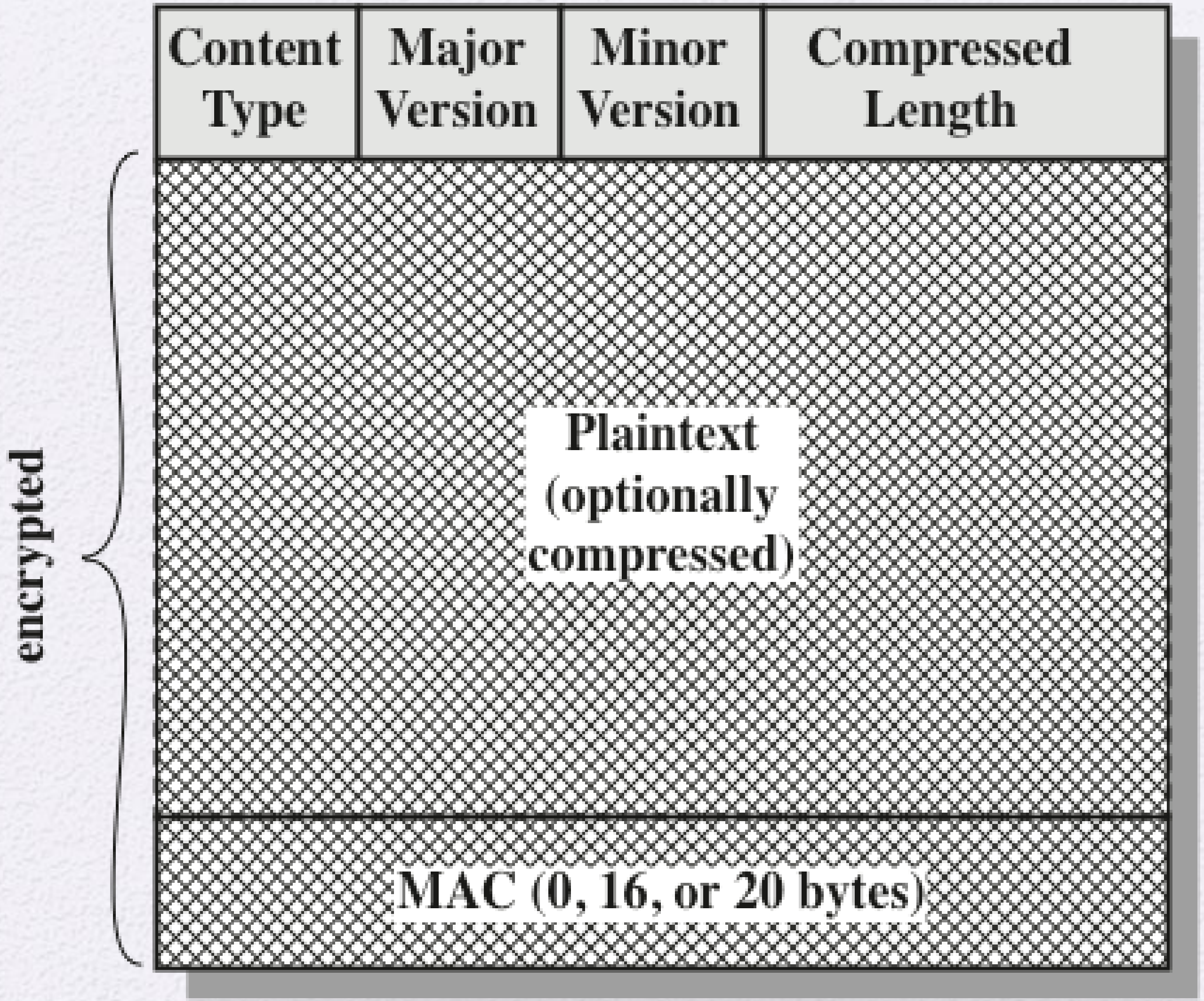
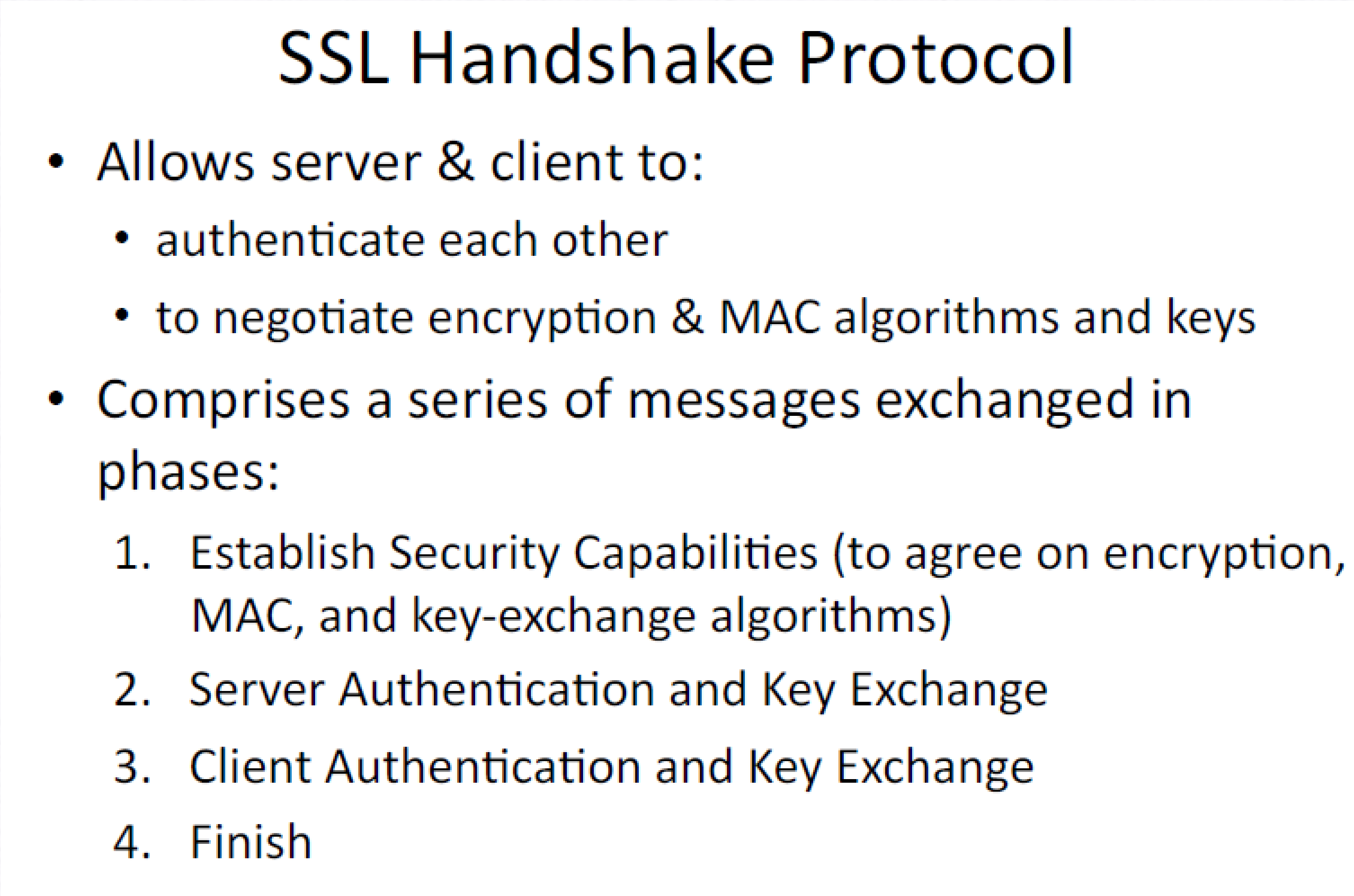


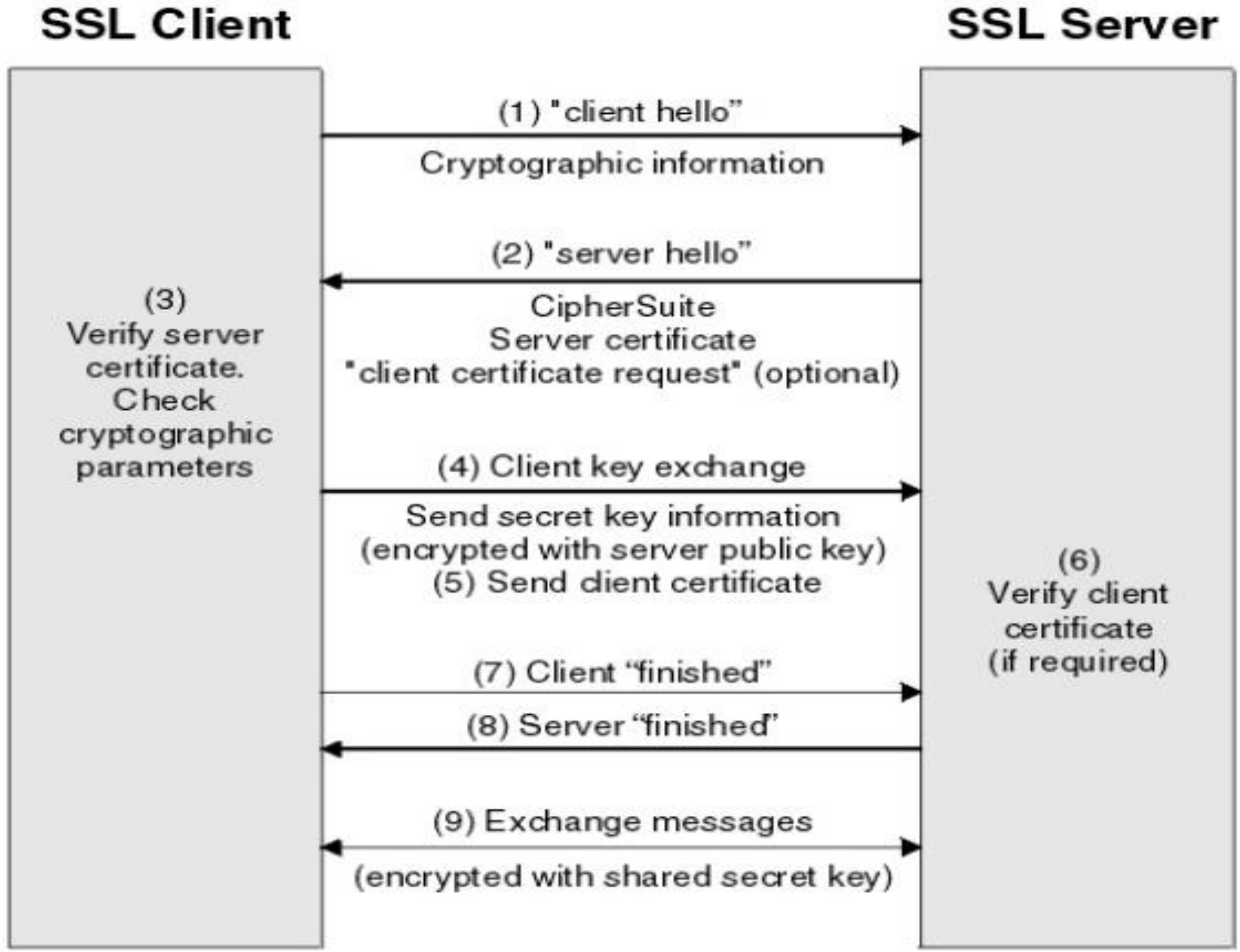
Figure 17.4 TLS Record Format



SSL Handshake Protocol

- Allows server & client to:
  - authenticate each other
  - to negotiate encryption & MAC algorithms and keys
- Comprises a series of messages exchanged in phases:
  1. Establish Security Capabilities (to agree on encryption, MAC, and key-exchange algorithms)
  2. Server Authentication and Key Exchange
  3. Client Authentication and Key Exchange
  4. Finish

Figure 1. Overview of the SSL or TLS handshake



SSL Client

SSL Server

(1) "client hello"

Cryptographic information

(2) "server hello"

CipherSuite
Server certificate
"client certificate request" (optional)

(3)
Verify server certificate.
Check cryptographic parameters

(4) Client key exchange

Send secret key information
(encrypted with server public key)
(5) Send client certificate

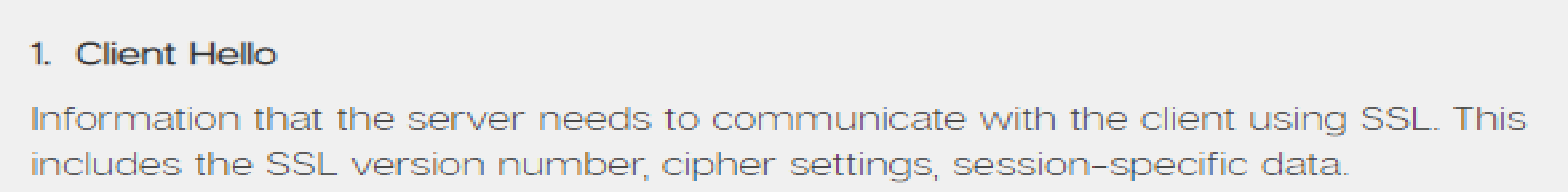
(6)
Verify client certificate
(if required)

(7) Client "finished"

(8) Server "finished"

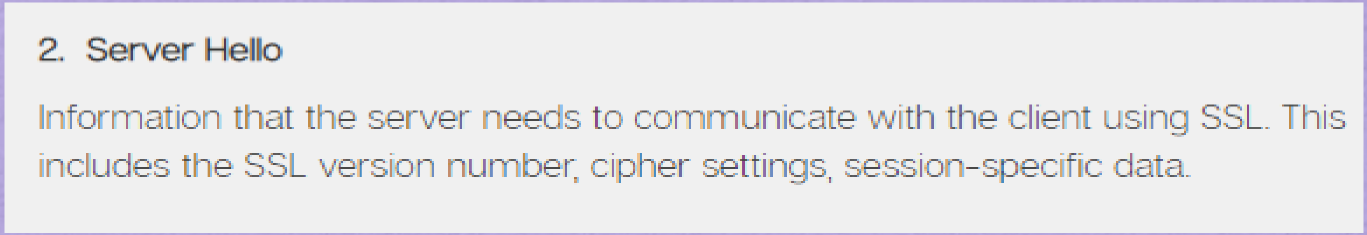
(9) Exchange messages

(encrypted with shared secret key)



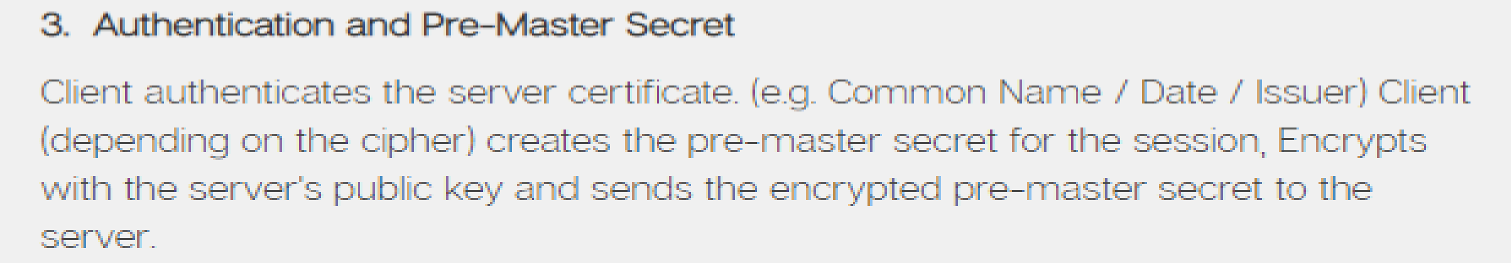
1. **Client Hello**

Information that the server needs to communicate with the client using SSL. This includes the SSL version number, cipher settings, session-specific data.



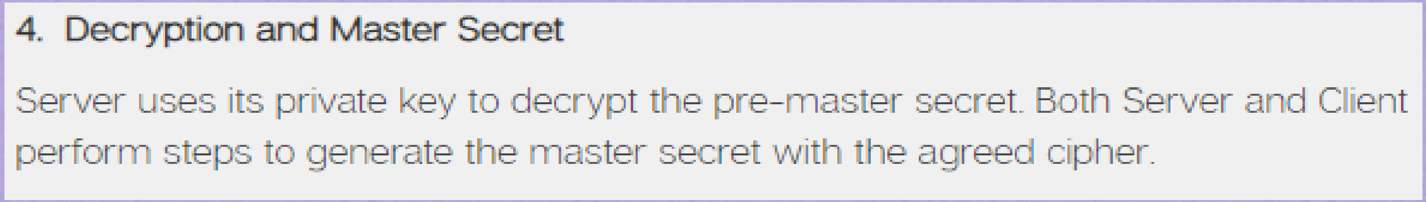
2. **Server Hello**

Information that the server needs to communicate with the client using SSL. This includes the SSL version number, cipher settings, session-specific data.



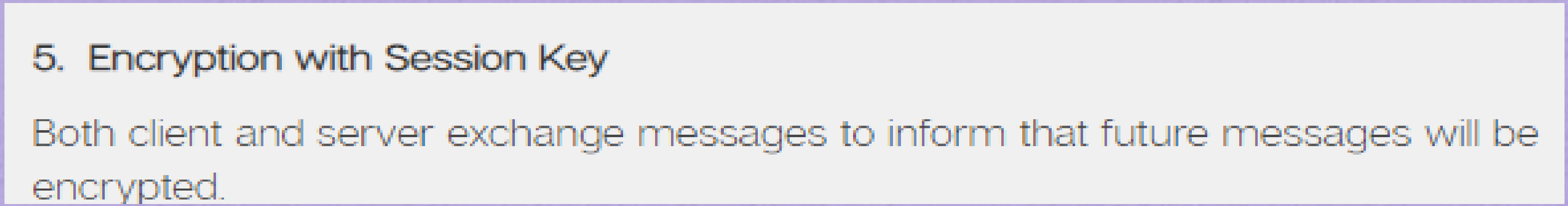
3. **Authentication and Pre-Master Secret**

Client authenticates the server certificate. (e.g. Common Name / Date / Issuer) Client (depending on the cipher) creates the pre-master secret for the session, Encrypts with the server's public key and sends the encrypted pre-master secret to the server.



4. **Decryption and Master Secret**

Server uses its private key to decrypt the pre-master secret. Both Server and Client perform steps to generate the master secret with the agreed cipher.



5. **Encryption with Session Key**

Both client and server exchange messages to inform that future messages will be encrypted.

SSL/TLS Attacks

- The attacks can be grouped into four general categories:
  - Attacks on the handshake protocol
  - Attacks on the record and application data protocols
  - Attacks on the PKI
  - Other attacks

- The constant back-and-forth between threats and countermeasures determines the evolution of Internet-based protocols

TLSv1.3

- Primary aim is to improve the security of TLS
- Significant changes from version 1.2 are:
  - TLSv1.3 removes support for a number of options and functions
    - Deleted items include:
      - Compression
      - Ciphers that do not offer authenticated encryption
      - Static RSA and DH key exchange
      - 32-bit timestamp as part of the Random parameter in the client_hello message
      - Renegotiation
      - Change Cipher Spec Protocol
      - RC4
      - Use of MD5 and SHA-224 hashes with signatures
  - TLSv1.3 uses Diffie-Hellman or Elleptic Curve Diffie-Hellman for key exchange and does not permit RSA
  - TLSv1.3 allows for a "1 round trip time" handshake by changing the order of message sent with establishing a secure connection

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 is 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

A TLS request to establish a connection begins with the establishment of a TCP connection between the TCP entity on the client side and the TCP entity on the server side

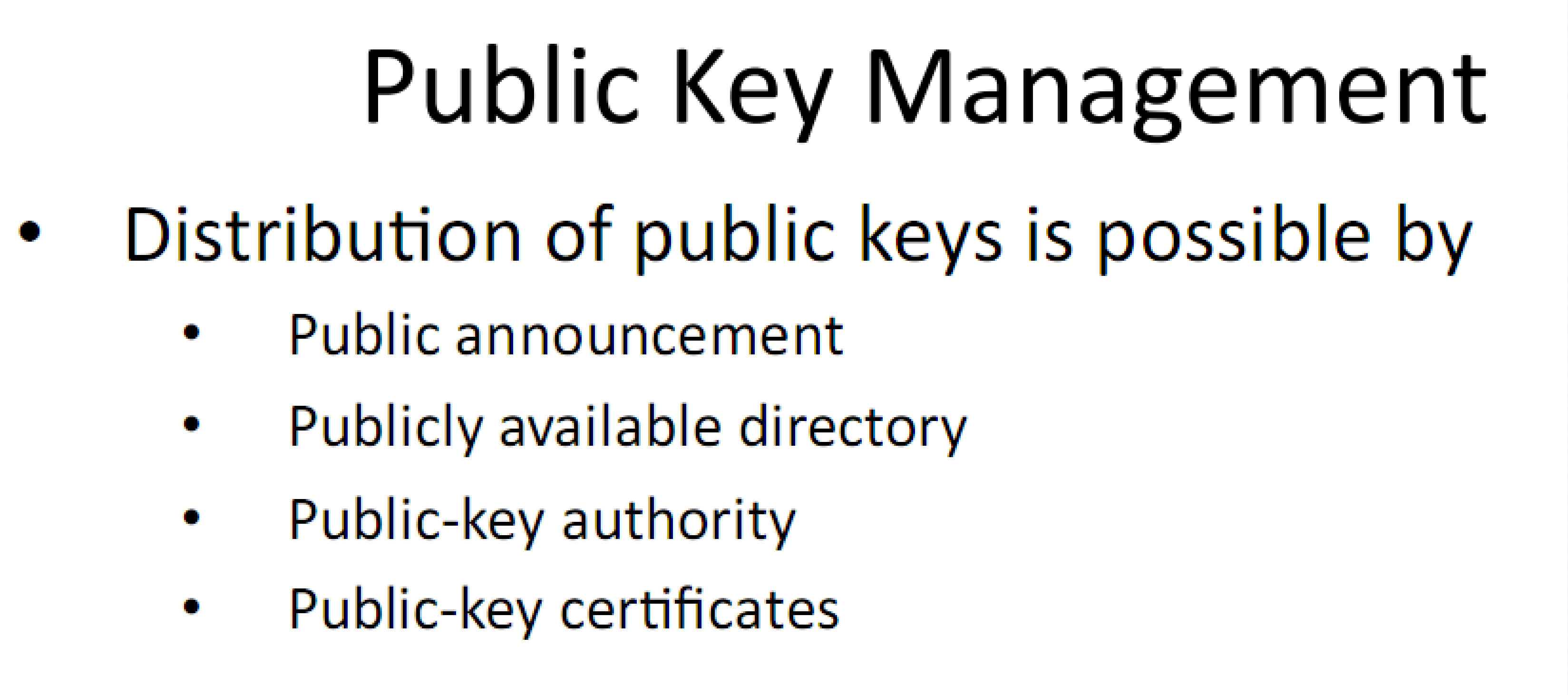
Connection Closure

- 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

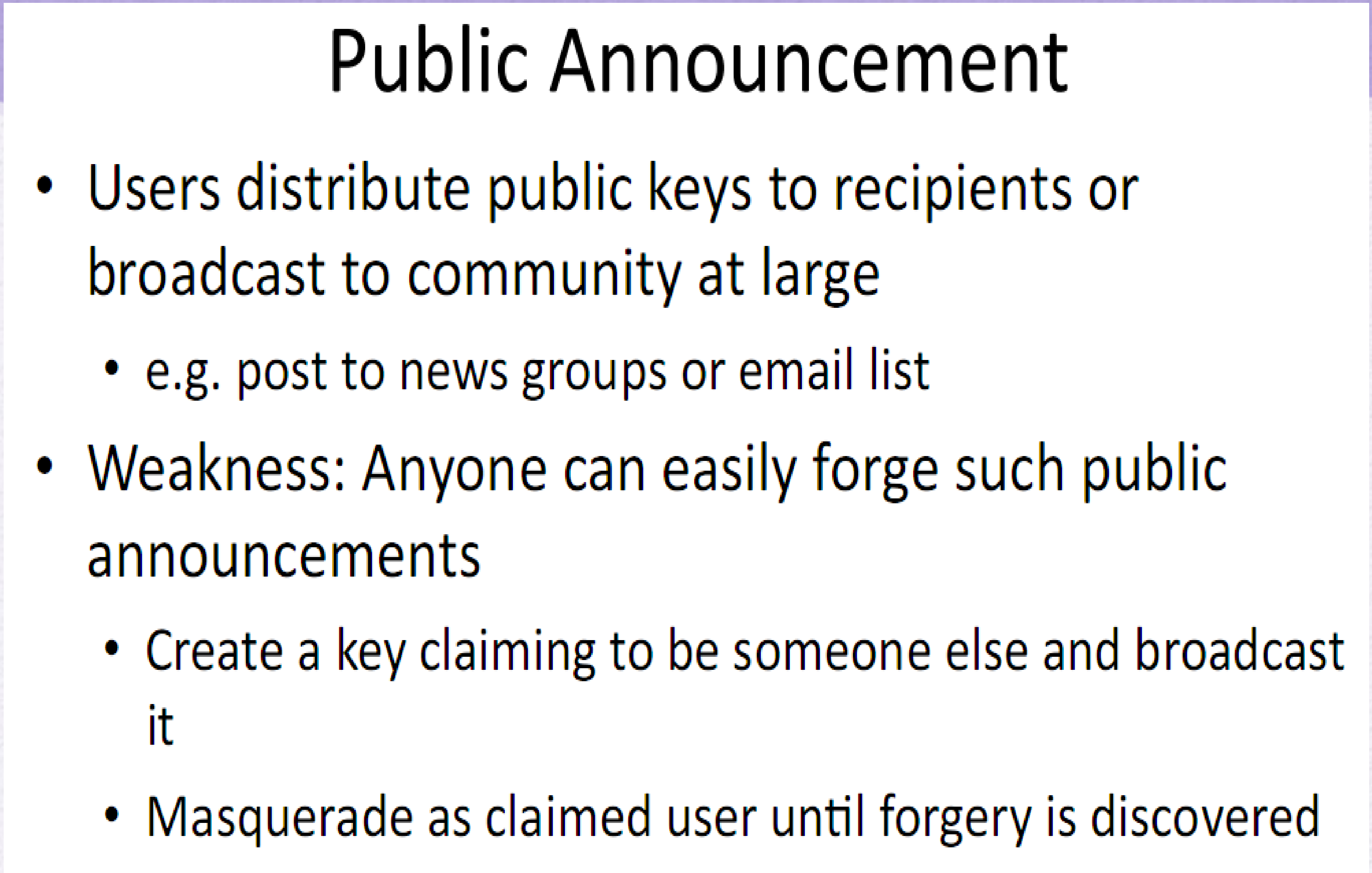
- TLS implementations must initiate an exchange of closure alerts before closing a connection
  - A TLS implementation may, after sending a closure alert, close the connection without waiting for the peer to send its closure alert, generating an "incomplete close"

- 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



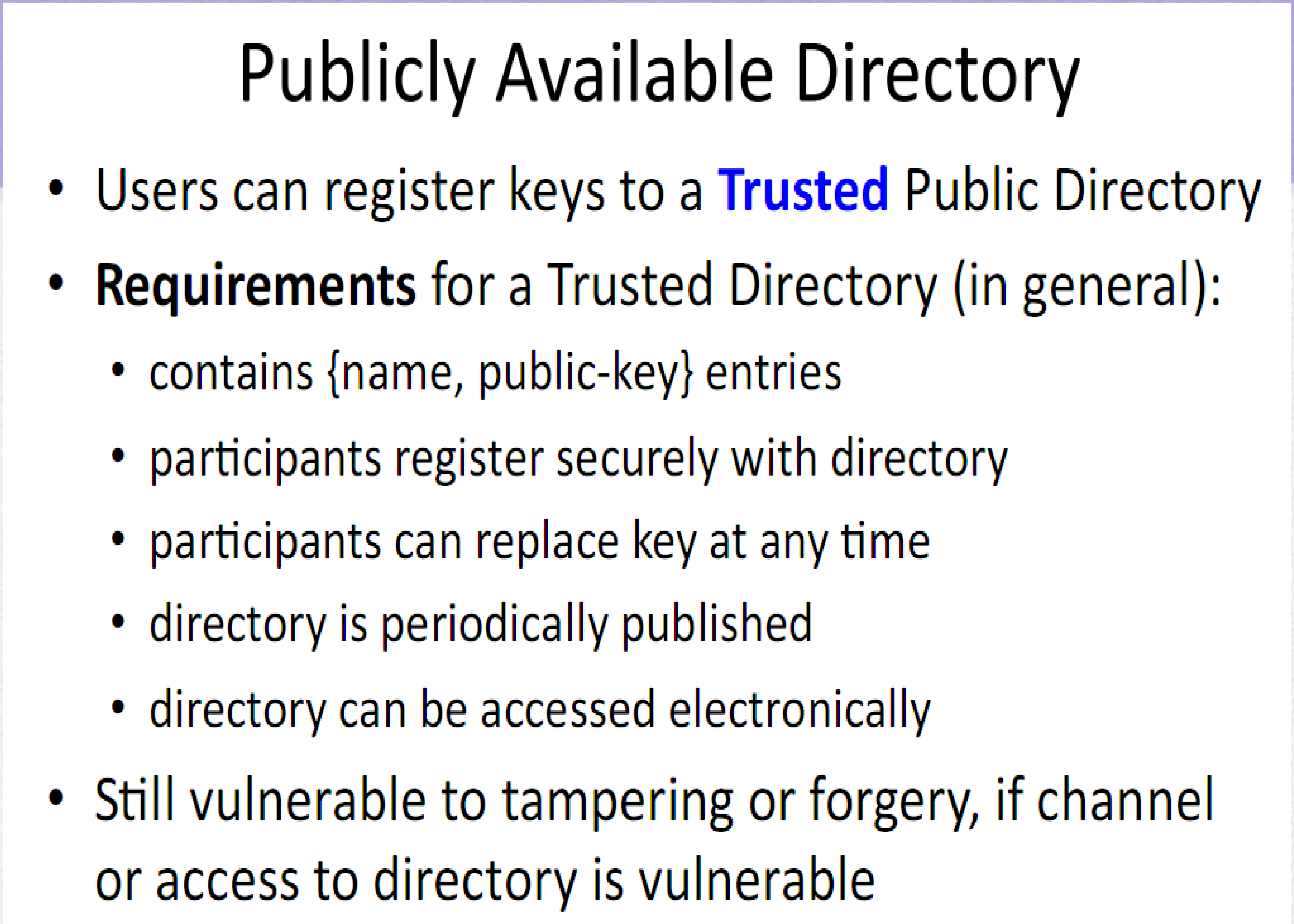
Public Key Management

- Distribution of public keys is possible by
  - Public announcement
  - Publicly available directory
  - Public-key authority
  - Public-key certificates



Public Announcement

- Users distribute public keys to recipients or broadcast to community at large
  - e.g. post to news groups or email list
- Weakness: Anyone can easily forge such public announcements
  - Create a key claiming to be someone else and broadcast it
  - Masquerade as claimed user until forgery is discovered

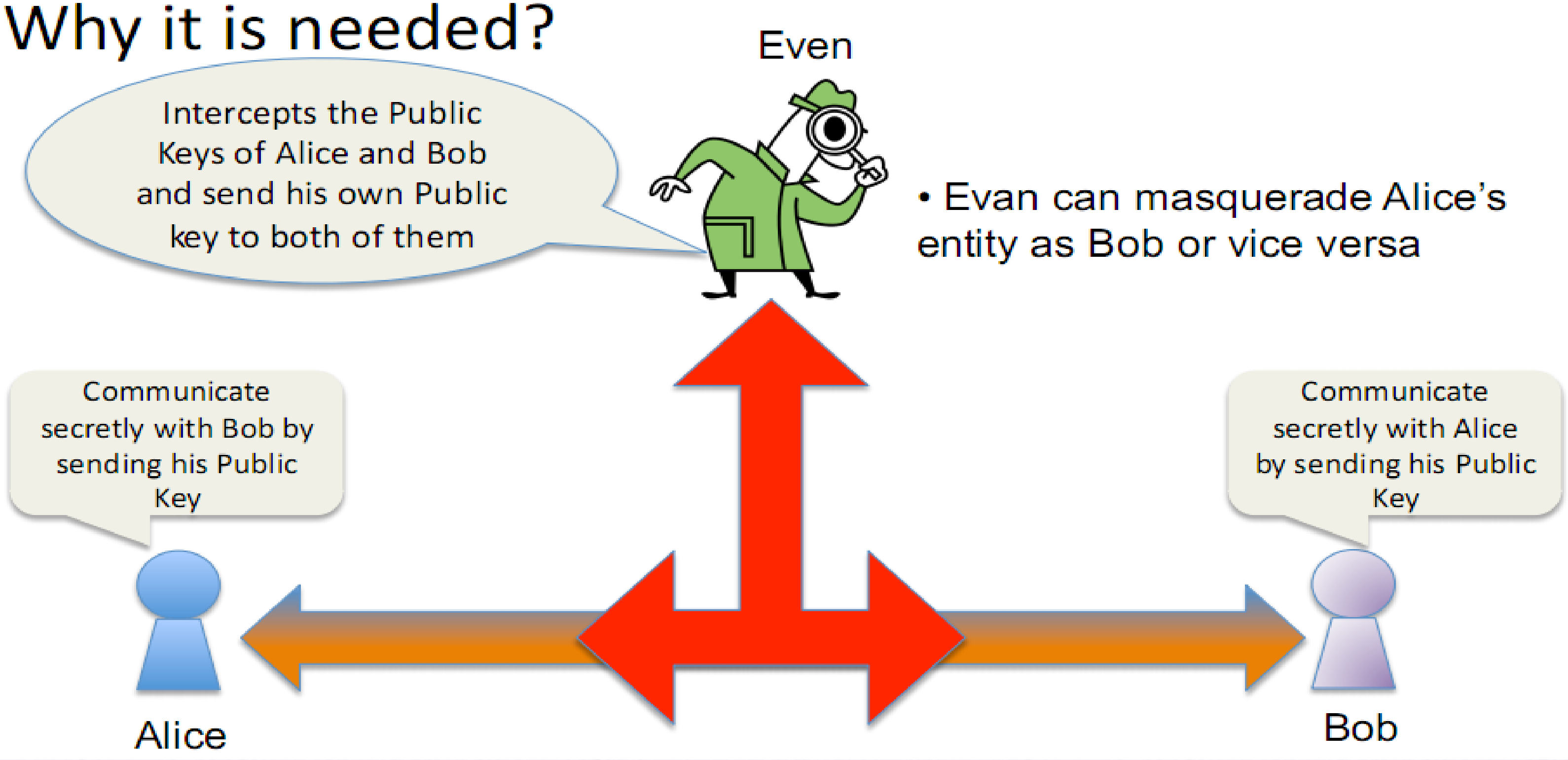


Publicly Available Directory

- Users can register keys to a **Trusted** Public Directory
- **Requirements** for a Trusted Directory (in general):
  - contains {name, public-key} entries
  - participants register securely with directory
  - participants can replace key at any time
  - directory is periodically published
  - directory can be accessed electronically
- Still vulnerable to tampering or forgery, if channel or access to directory is vulnerable

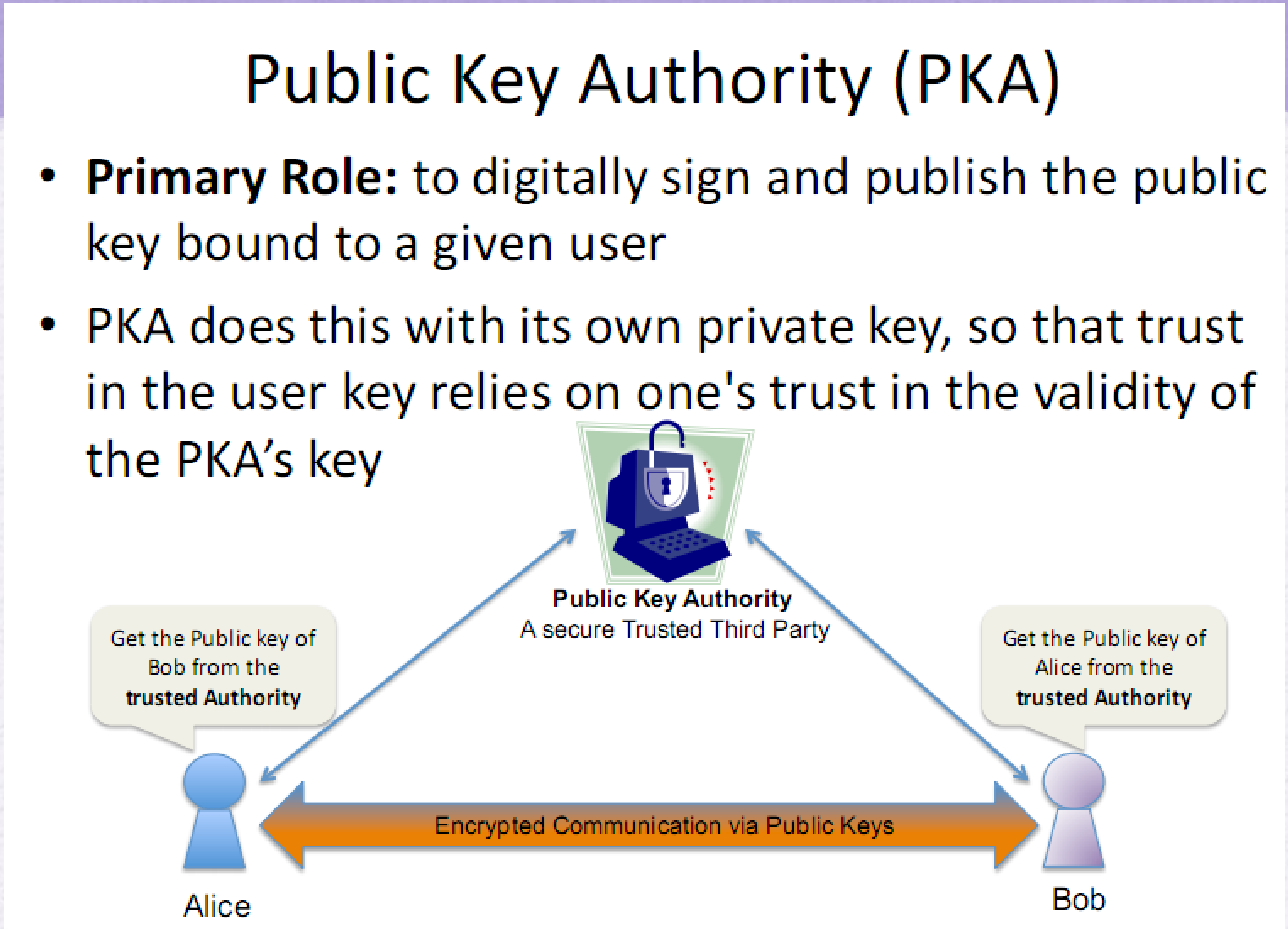
Public Key Authority (PKA)

- A well-trusted Authority is required to distribute keys from the directory
- Why it is needed?



Even
Intercepts the Public Keys of Alice and Bob and send his own Public key to both of them
Communicate secretly with Bob by sending his Public Key
Communicate secretly with Alice by sending his Public Key
Alice
Bob

- Evan can masquerade Alice's entity as Bob or vice versa

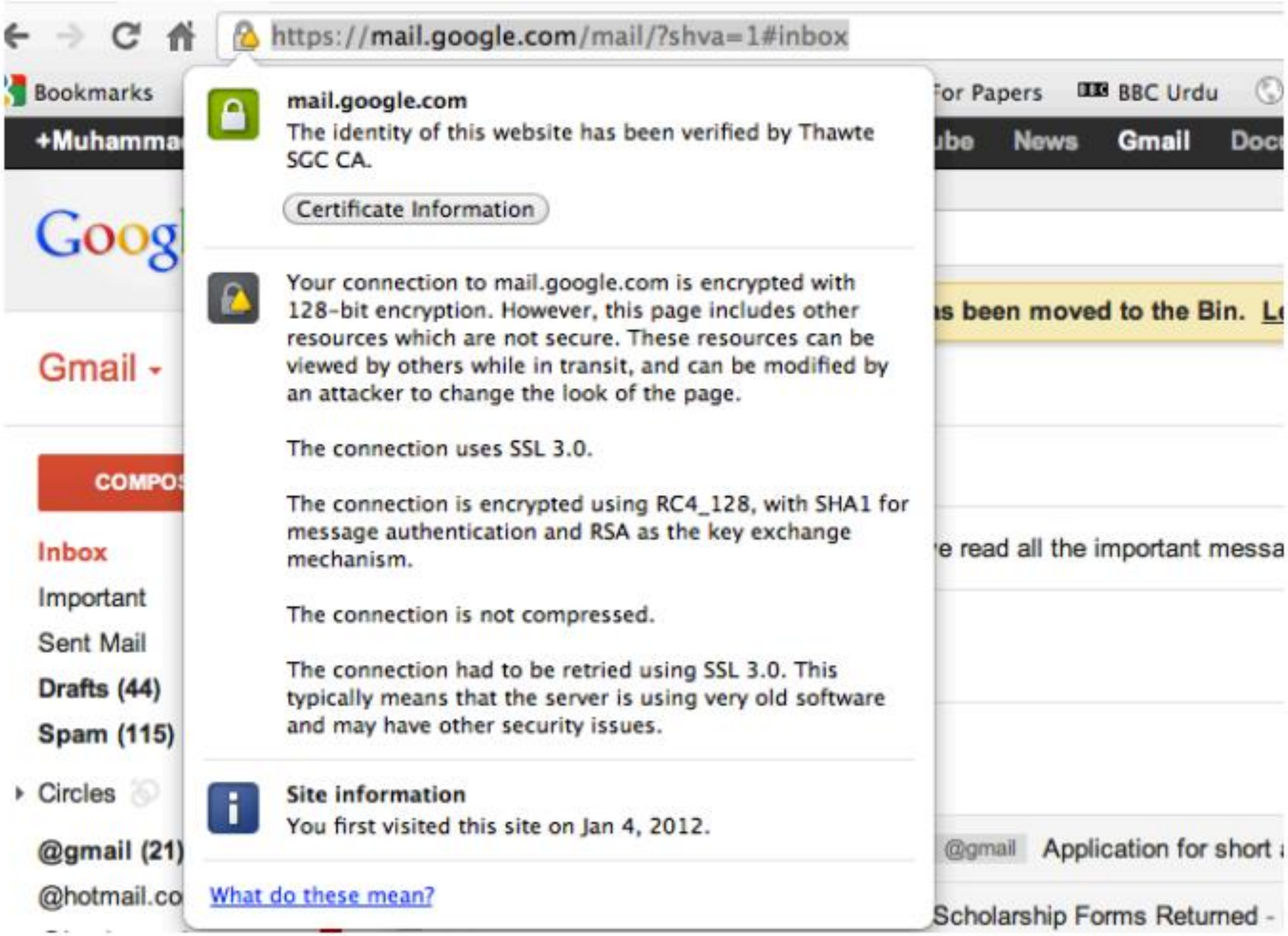


Public Key Authority (PKA)

- **Primary Role:** to digitally sign and publish the public key bound to a given user
- PKA does this with its own private key, so that trust in the user key relies on one's trust in the validity of the PKA's key

Public Key Authority
A secure Trusted Third Party
Get the Public key of Bob from the **trusted Authority**
Get the Public key of Alice from the **trusted Authority**
Encrypted Communication via Public Keys
Alice
Bob

Public Key Certificates



mail.google.com

The identity of this website has been verified by Thawte SGC CA.

Certificate Information

Your connection to mail.google.com is encrypted with 128-bit encryption. However, this page includes other resources which are not secure. These resources can be viewed by others while in transit, and can be modified by an attacker to change the look of the page.

The connection uses SSL 3.0.

The connection is encrypted using RC4_128, with SHA1 for message authentication and RSA as the key exchange mechanism.

The connection is not compressed.

The connection had to be retried using SSL 3.0. This typically means that the server is using very old software and may have other security issues.

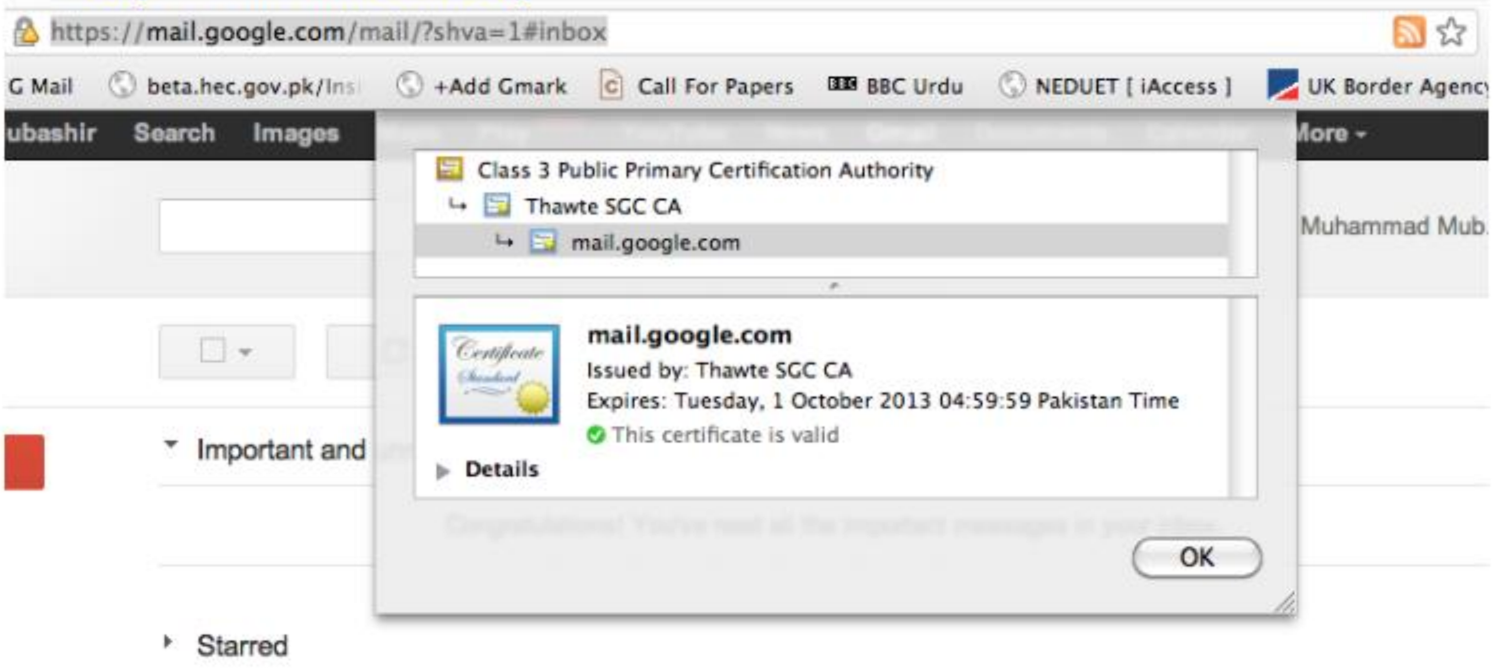
Site information

You first visited this site on Jan 4, 2012.

What do these mean?

Public Key Certificates

- In cryptography, a public key certificate (also known as a digital certificate or identity certificate) is an electronic document which uses a digital signature to bind a public key with an identity

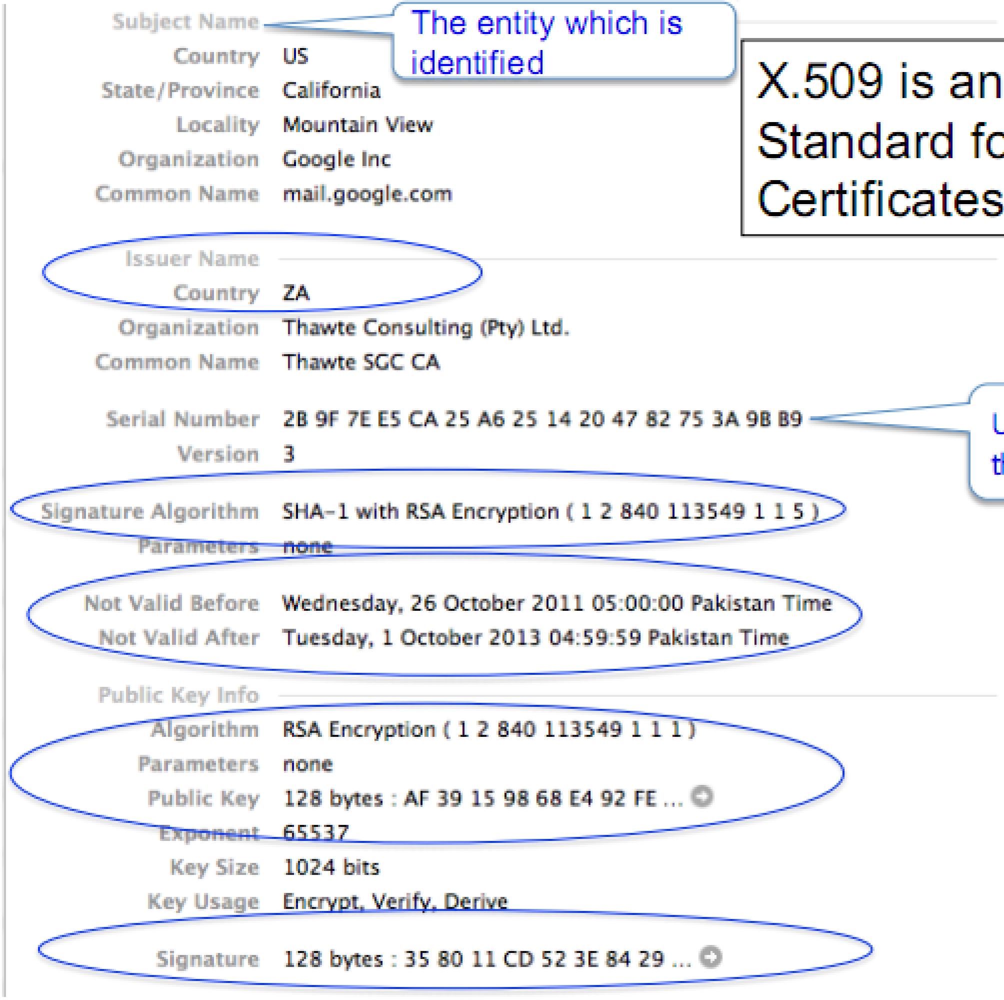


https://mail.google.com/mail/?shva=1#inbox

Class 3 Public Primary Certification Authority
↳ Thawte SGC CA
↳ mail.google.com

mail.google.com
Issued by: Thawte SGC CA
Expires: Tuesday, 1 October 2013 04:59:59 Pakistan Time
This certificate is valid

X.509 Public Key Certificate

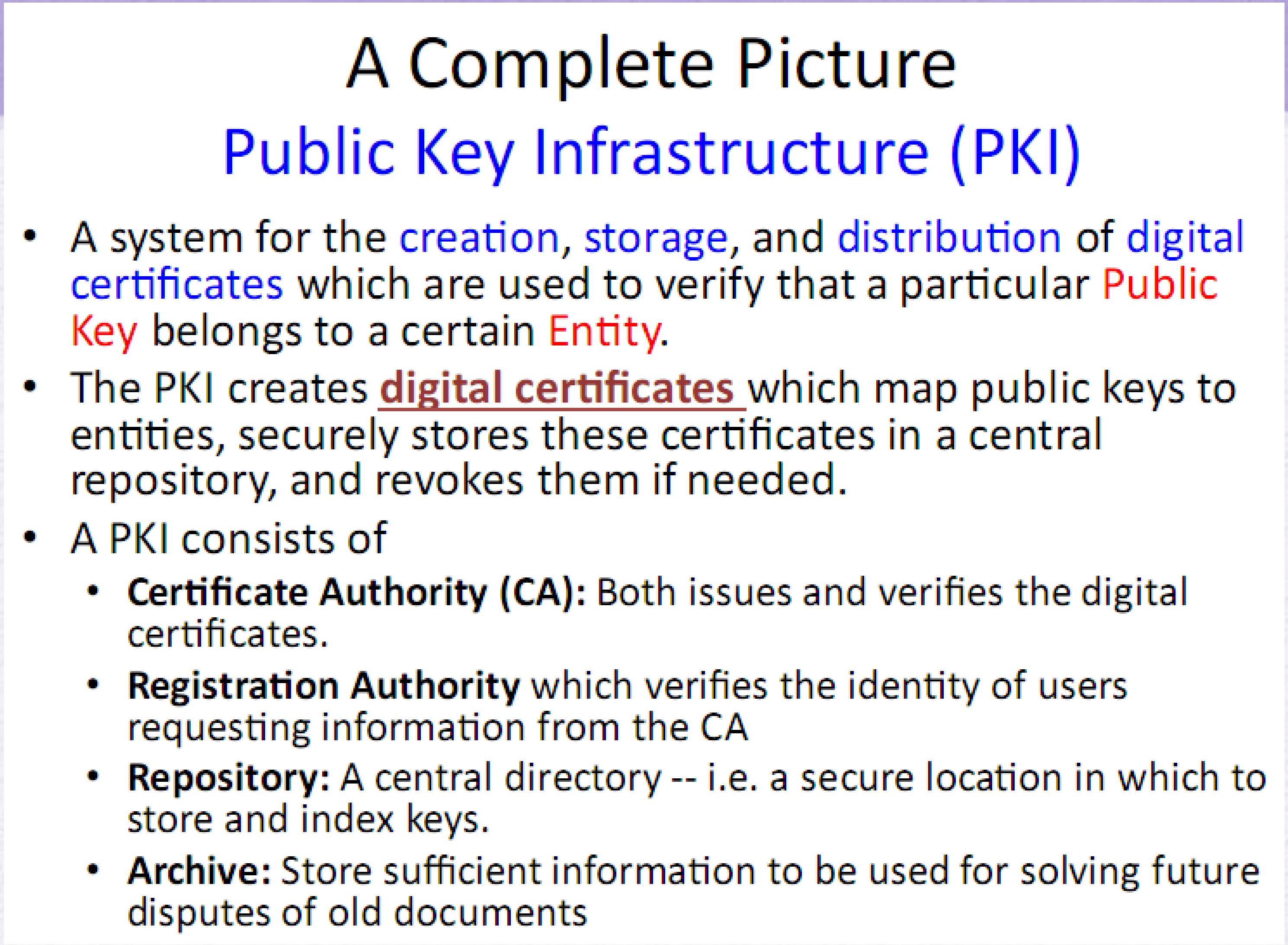


Subject Name	
Country	US
State/Province	California
Locality	Mountain View
Organization	Google Inc
Common Name	mail.google.com
Issuer Name	
Country	ZA
Organization	Thawte Consulting (Pty) Ltd.
Common Name	Thawte SGC CA
Serial Number	2B 9F 7E E5 CA 25 A6 25 14 20 47 82 75 3A 9B B9
Version	3
Signature Algorithm	SHA-1 with RSA Encryption (1 2 840 113549 1 1 5)
Parameters	none
Not Valid Before	Wednesday, 26 October 2011 05:00:00 Pakistan Time
Not Valid After	Tuesday, 1 October 2013 04:59:59 Pakistan Time
Public Key Info	
Algorithm	RSA Encryption (1 2 840 113549 1 1 1)
Parameters	none
Public Key	128 bytes : AF 39 15 98 68 E4 92 FE ...
Exponent	65537
Key Size	1024 bits
Key Usage	Encrypt, Verify, Derive
Signature	128 bytes : 35 80 11 CD 52 3E 84 29 ...

The entity which is identified

Used to uniquely identify the certificate

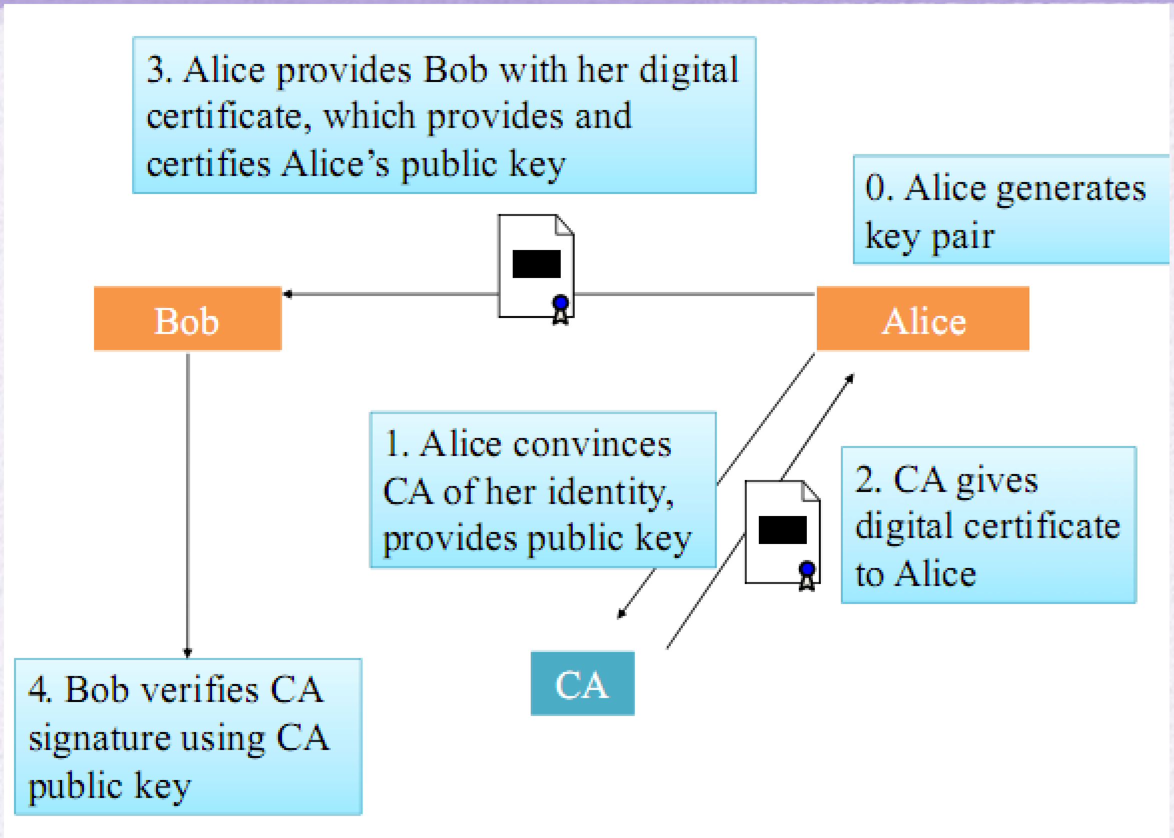
X.509 is an ITU-T Standard for Public Key Certificates



A Complete Picture

Public Key Infrastructure (PKI)

- A system for the creation, storage, and distribution of digital certificates which are used to verify that a particular Public Key belongs to a certain Entity.
- The PKI creates **digital certificates** which map public keys to entities, securely stores these certificates in a central repository, and revokes them if needed.
- A PKI consists of
  - **Certificate Authority (CA):** Both issues and verifies the digital certificates.
  - **Registration Authority** which verifies the identity of users requesting information from the CA
  - **Repository:** A central directory -- i.e. a secure location in which to store and index keys.
  - **Archive:** Store sufficient information to be used for solving future disputes of old documents



3. Alice provides Bob with her digital certificate, which provides and certifies Alice’s public key

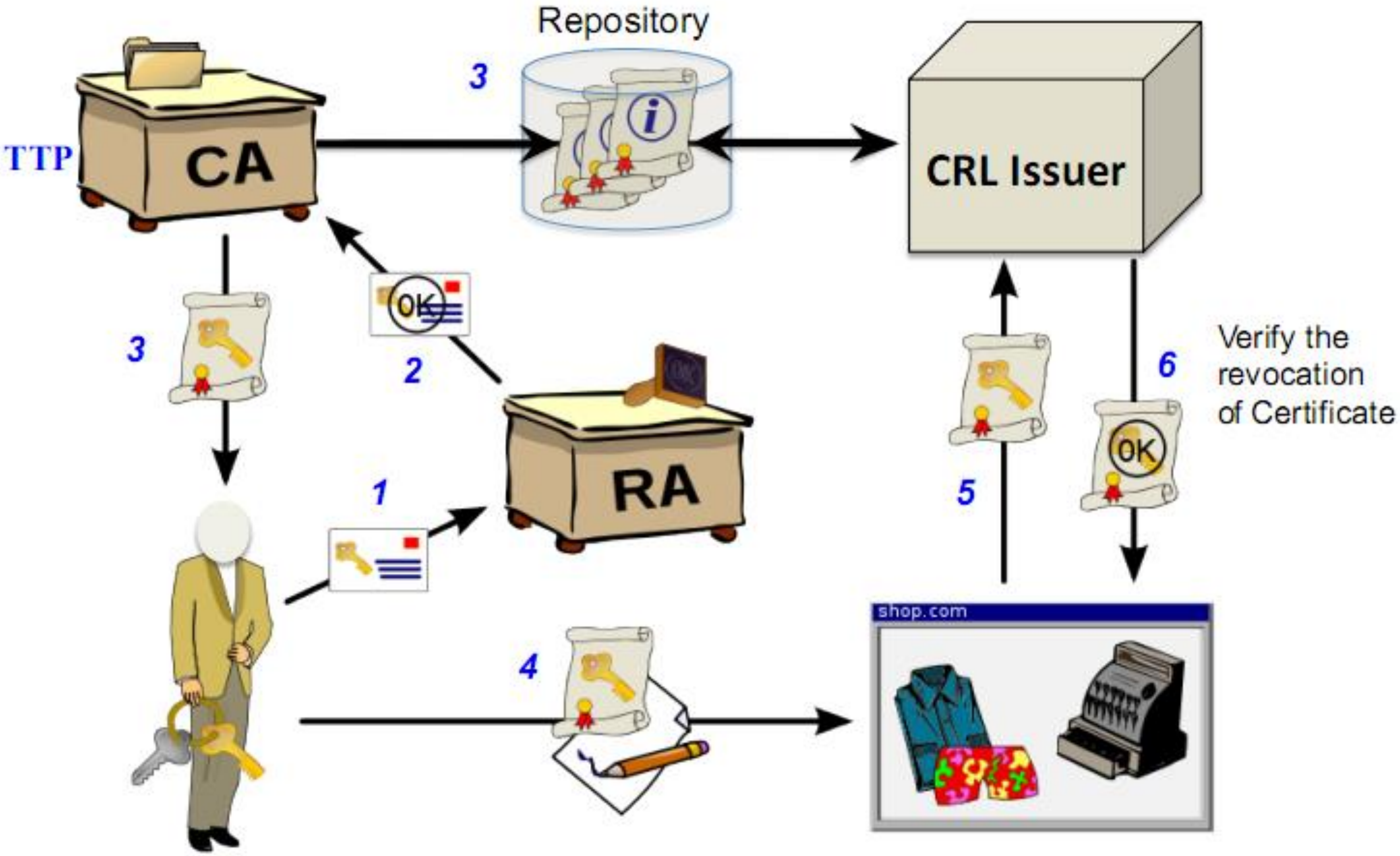
0. Alice generates key pair

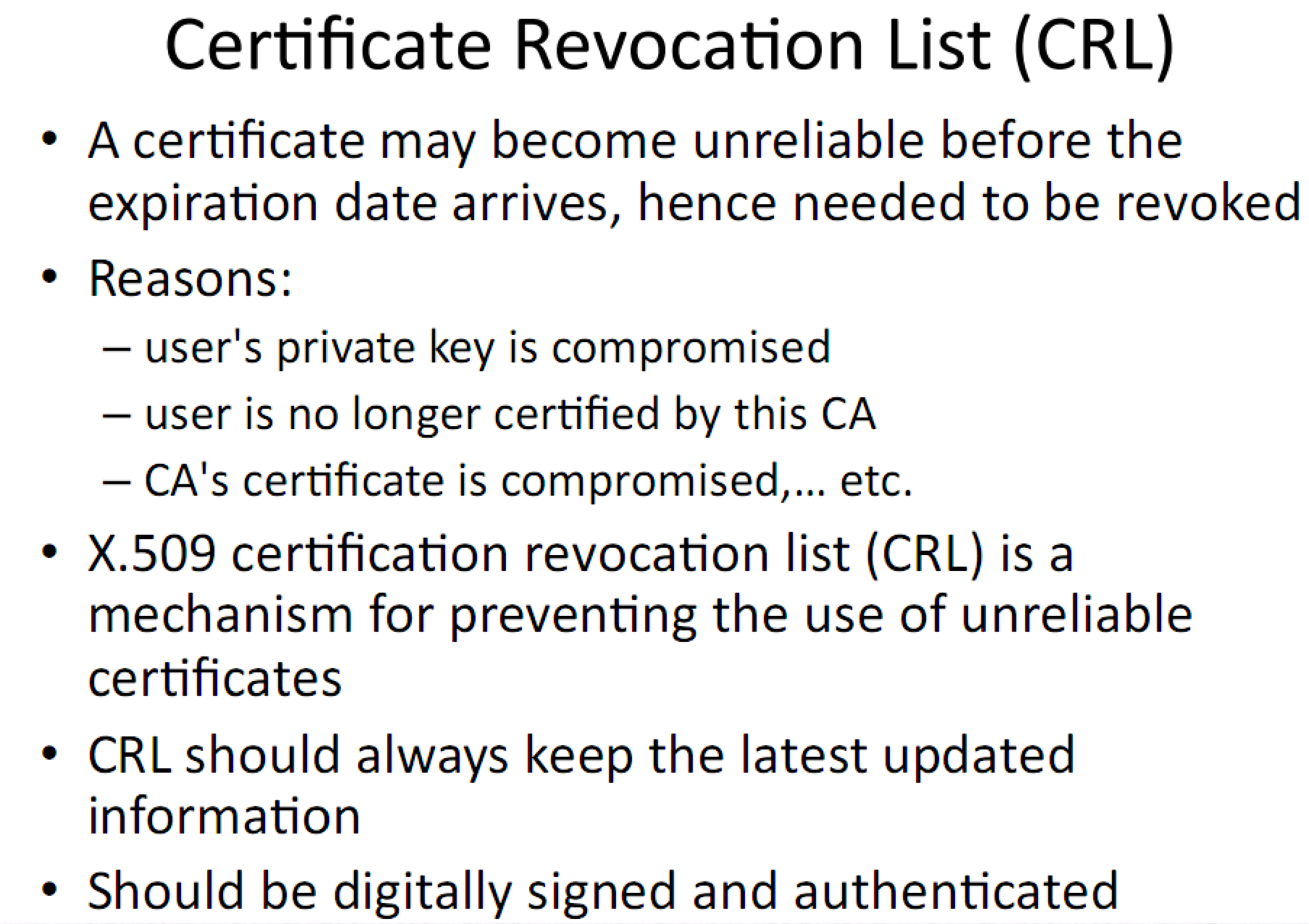
1. Alice convinces CA of her identity, provides public key

2. CA gives digital certificate to Alice

4. Bob verifies CA signature using CA public key

Public Key Infrastructure (PKI)





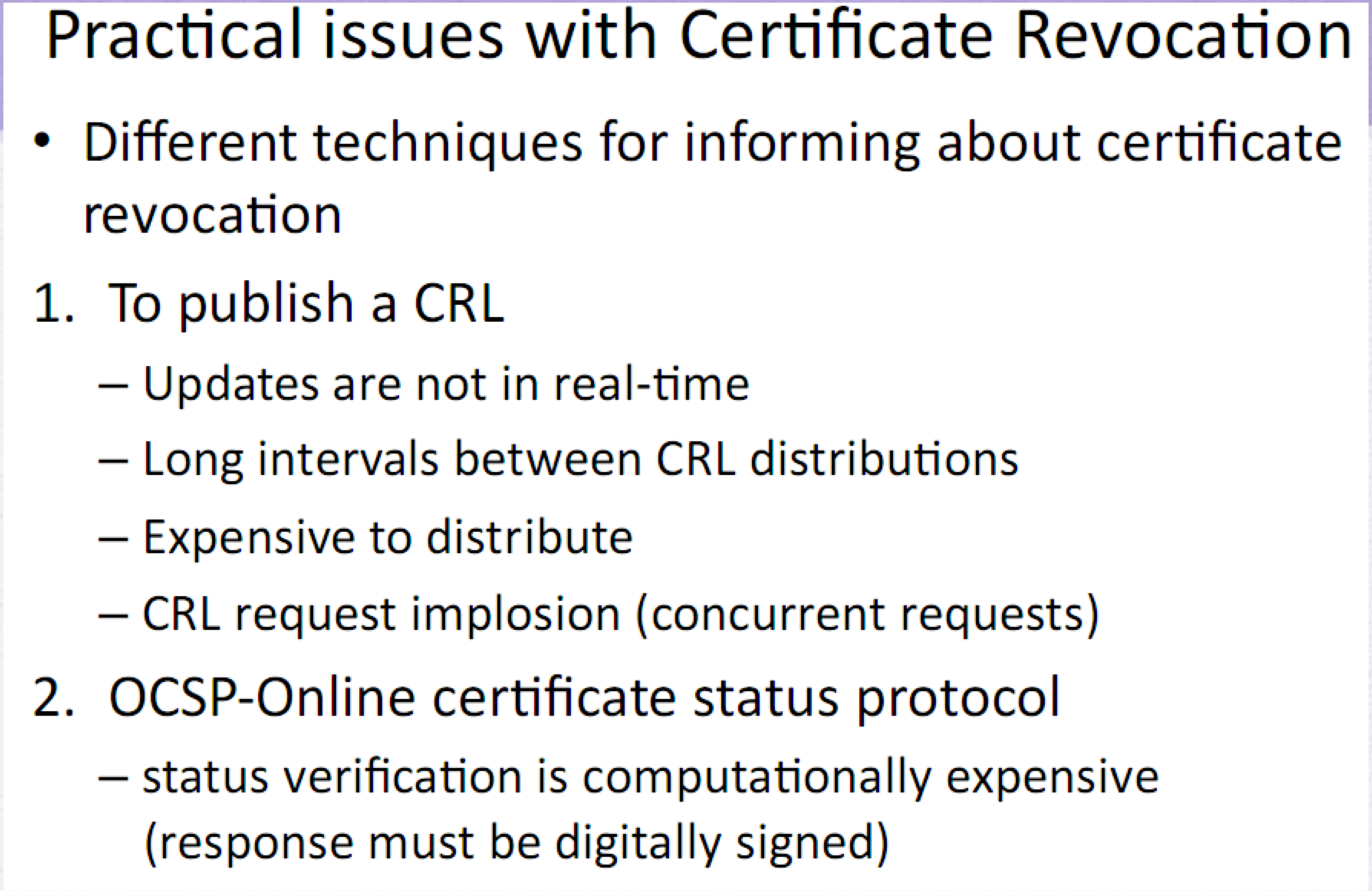
Certificate Revocation List (CRL)

- A certificate may become unreliable before the expiration date arrives, hence needed to be revoked
- Reasons:
  - user's private key is compromised
  - user is no longer certified by this CA
  - CA's certificate is compromised,… etc.
- X.509 certification revocation list (CRL) is a mechanism for preventing the use of unreliable certificates
- CRL should always keep the latest updated information
- Should be digitally signed and authenticated

X.509 CRL Format

The CRL contains the following main fields with certain extensions:

- **Version:** describes the syntax of the CRL
- **Signature:** contains the algorithm identifier for the digital signature algorithm used by the CRL issuer to sign the CRL
- **Issuer:** contains the X.500 distinguished name of the CRL issuer
- **This update:** indicates the issue date of this CRL
- **Next update:** indicates the date by which the next CRL will be issued
- **Revoked certificates:** lists the revoked certificates (contains the certificate serial number, time of revocation, and optional CRL entry extensions)



Practical issues with Certificate Revocation

- Different techniques for informing about certificate revocation

1. To publish a CRL
   - Updates are not in real-time
   - Long intervals between CRL distributions
   - Expensive to distribute
   - CRL request implosion (concurrent requests)
2. OCSP-Online certificate status protocol
   - status verification is computationally expensive (response must be digitally signed)