FSI

LEI

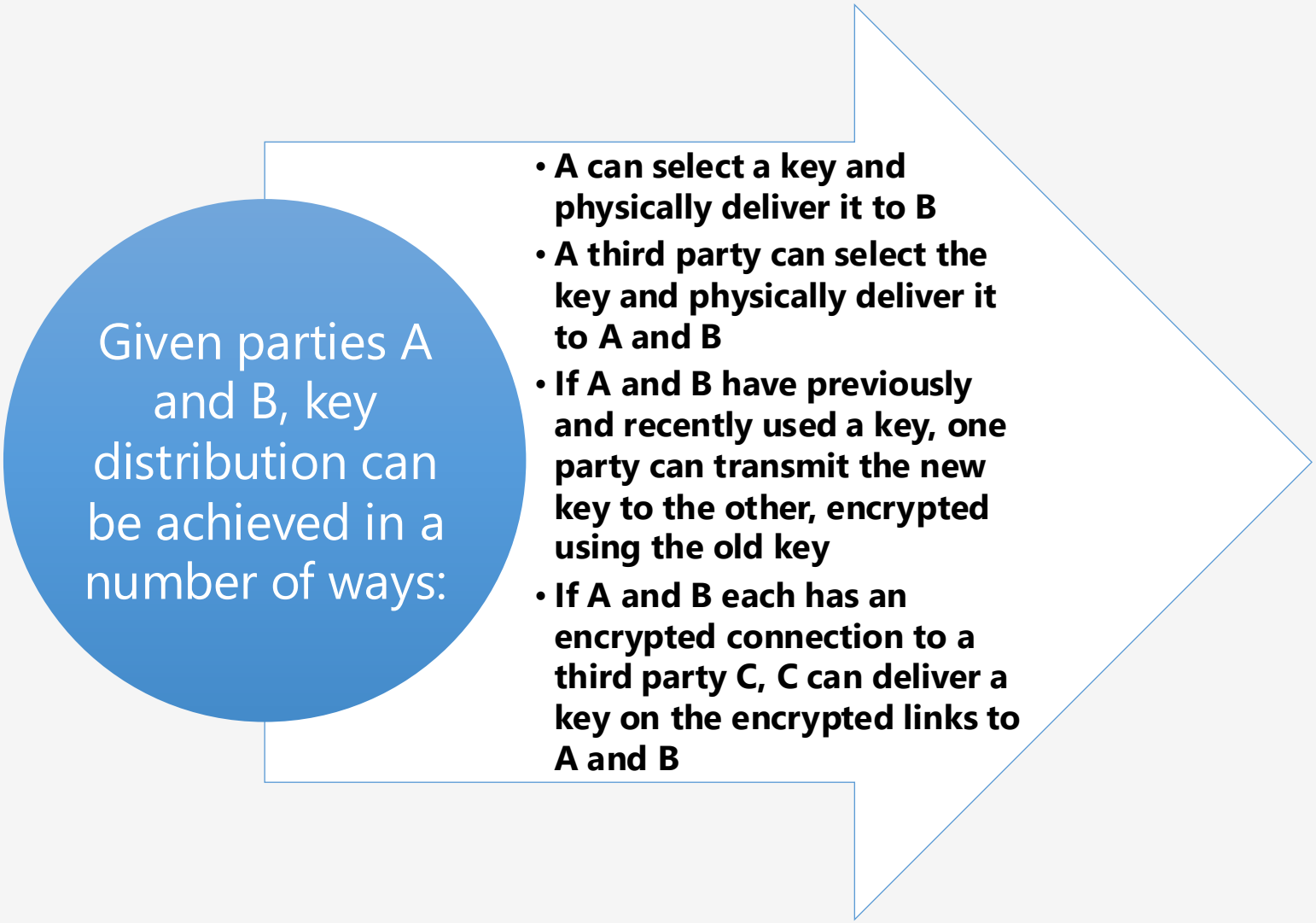
2025/2026

T6 – Key Management and User Authentication

Key Distribution Technique

- Term that refers to the means of delivering a key to two parties who wish to exchange data, without allowing others to see the key
- For symmetric encryption to work, the two parties to an exchange must share the same key, and that key must be protected from access by others
- Frequent key changes are desirable to limit the amount of data compromised if an attacker learns the key

Symmetric Key Distribution



Agreeing on keys for Arbitrary Connections Between Endpoints

- If end-to-end encryption is done at a network or IP level, a key is needed for each pair of hosts that wish to communicate
- If there are N hosts, the number of keys required is [N(N-1)]/2
- Example: a network with 1000 nodes would need to distribute as many as half million keys

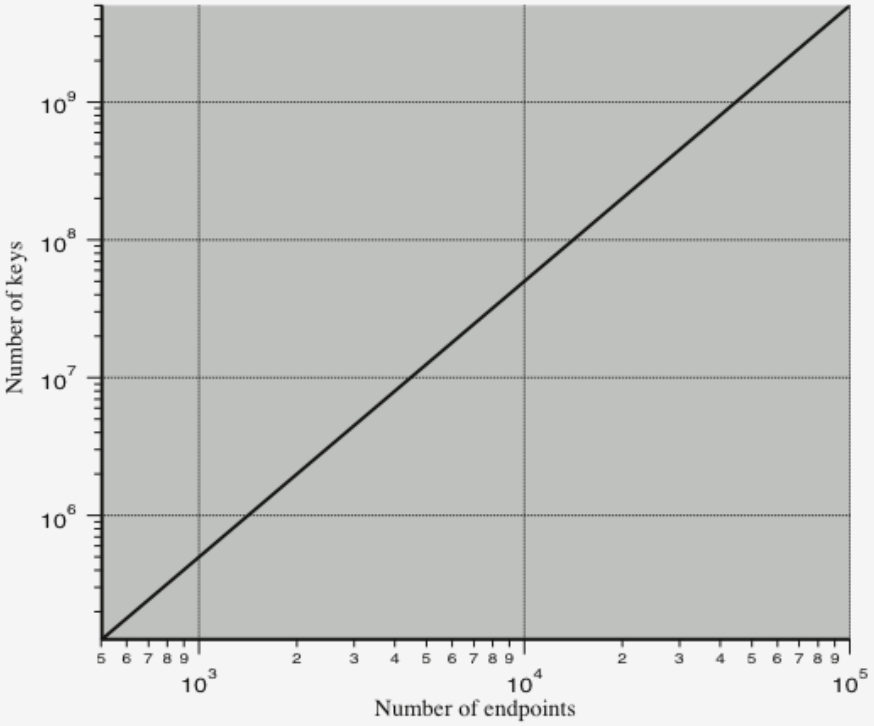


Figure 14.1 Number of Keys Required to Support Arbitrary Connections Between Endpoints

Key Hierarchy

- The use of a **KDC (Key Distribution Center)** is based on the use of a hierarchy of keys.
- Communications between end system use temporary (session) keys, e.g. for the duration of a logical session.
- Session keys are transmitted encrypted, using master keys shared by the KDC and end system or user.

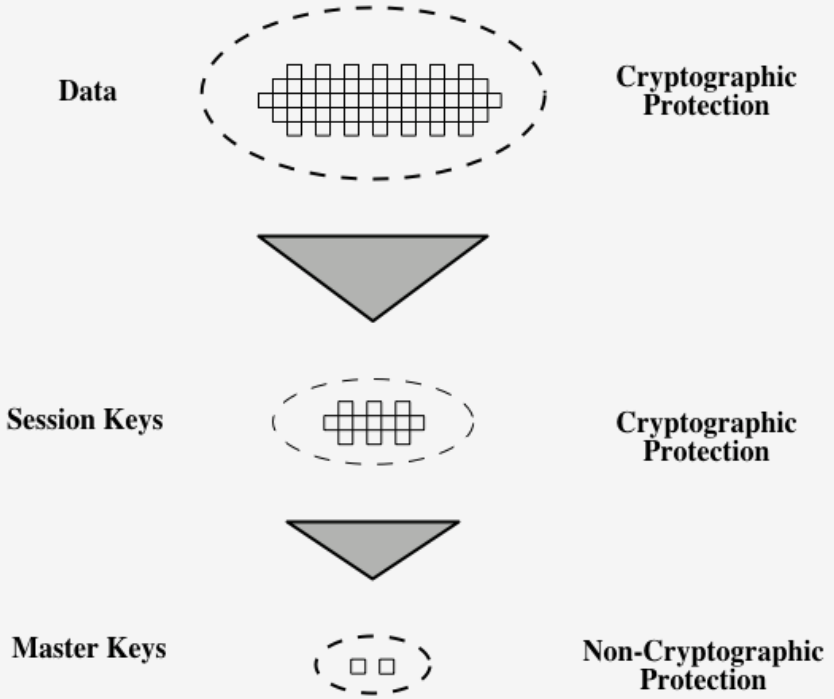
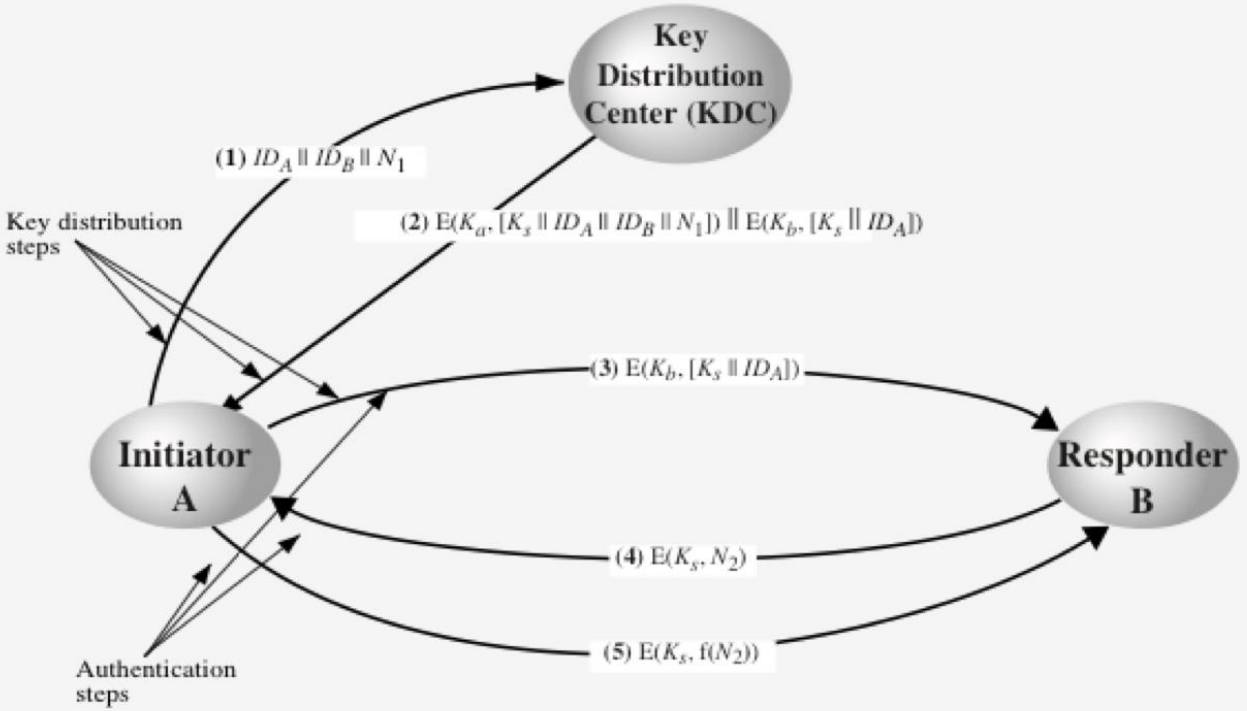


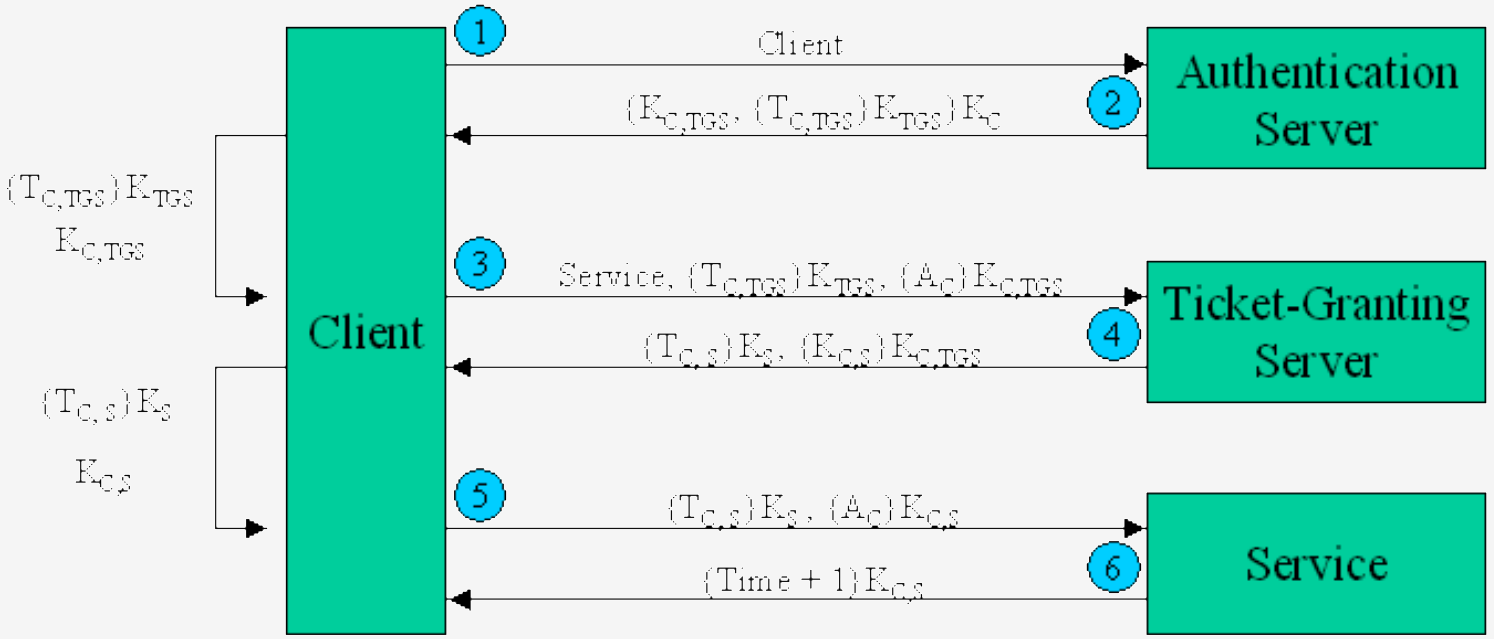
Figure 14.2 The Use of a Key Hierarchy

Key Distribution Scenario (using KDC)

- Assumes **each user shares a unique master key with KDC**: A and KDC share K_a, B and KDC share K_b
- Unique identifiers, or *nonces* (e.g. random numbers), are used to prevent replay attacks in steps 1 and 3



Key Distribution in Kerberos

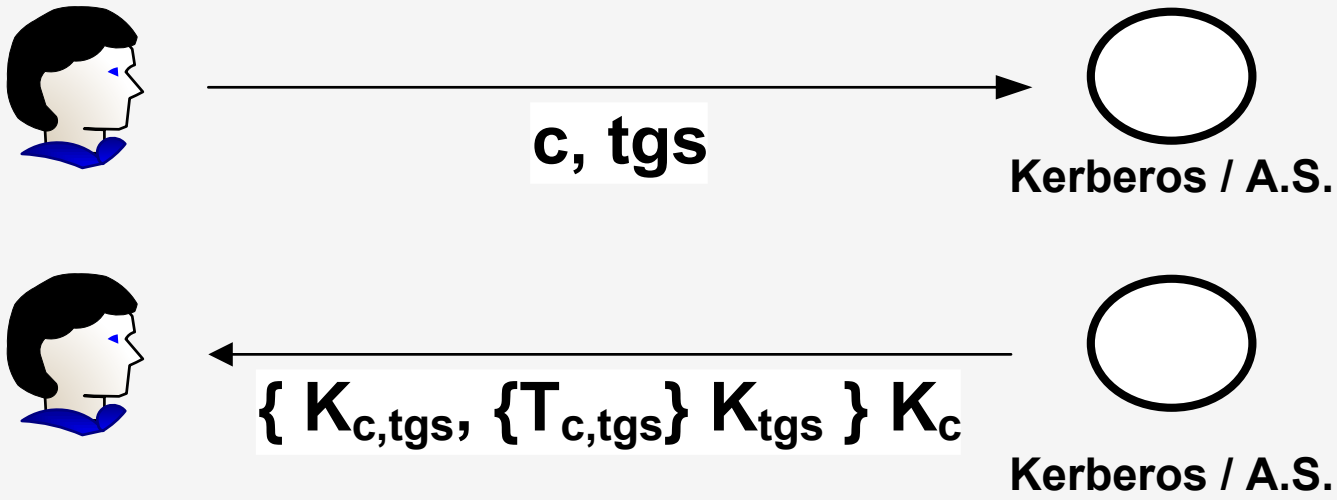


$T_{C,S}$ = {client, service, IP address, timestamp, lifetime, $K_{C,S}$}
= Ticket for Client to use Service

A_C = {Client, IP Address, Timestamp}
= Client Authenticator

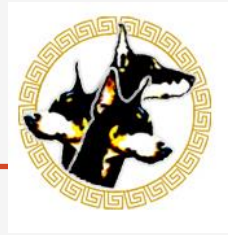
Kerberos authentication model

- Request for the initial ticket (or TGT, for the TGS):

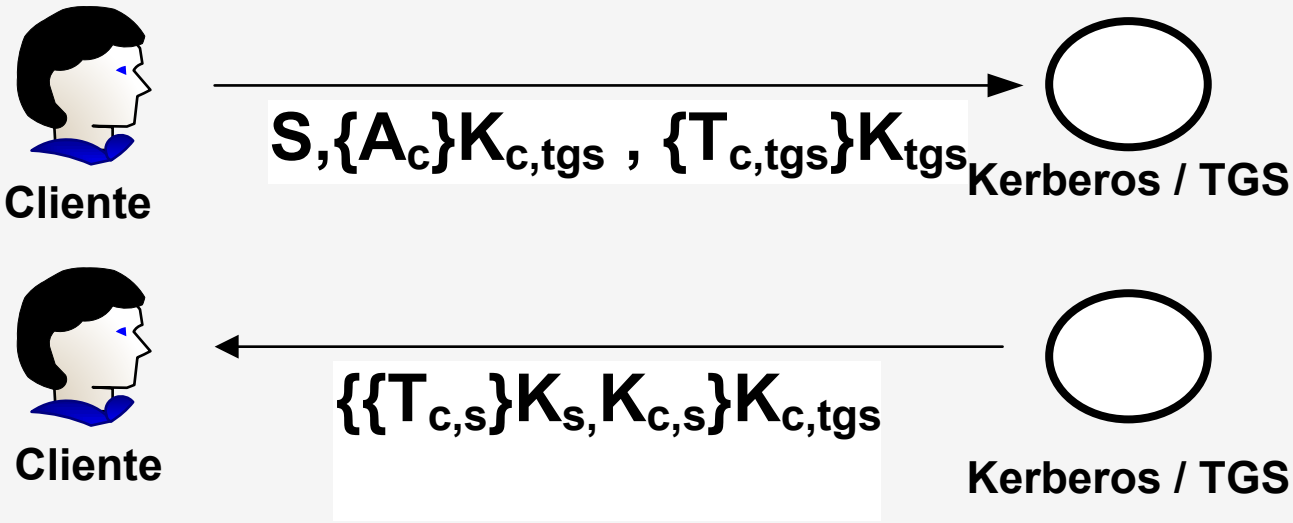


$$T_{c,tgs} = \{c,tgs,addr,timestamp,life,K_{c,tgs}\}$$

Kerberos authentication model



- Request to the TGS, for a ticket to a service:

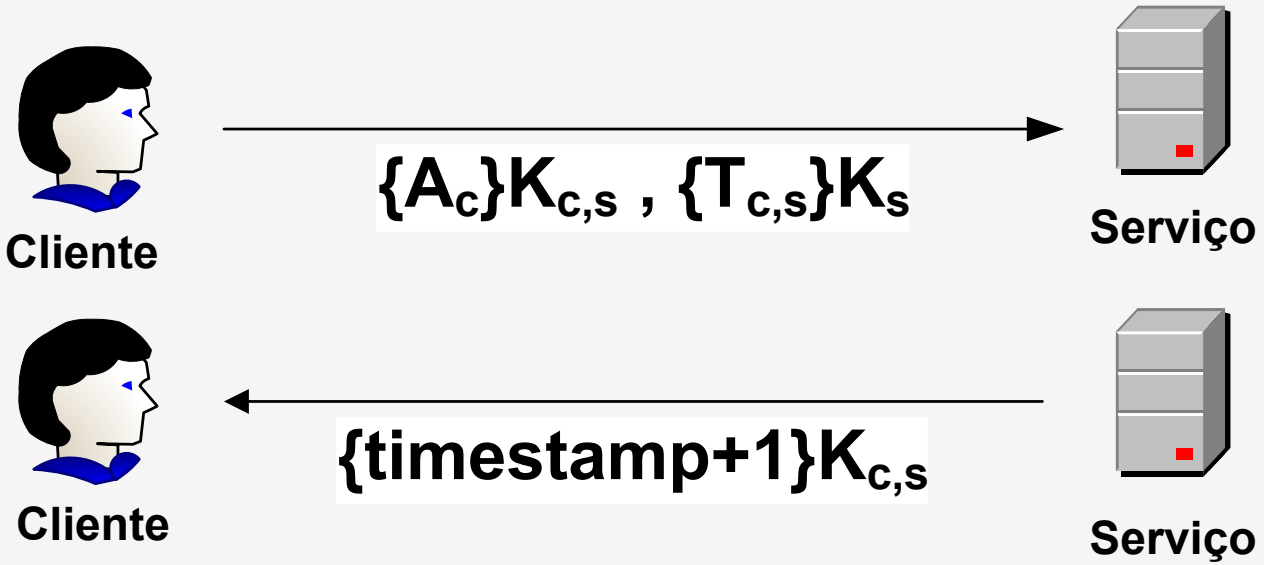


$$T_{c,tgs} = \{c,tgs,addr,timestamp,life,K_{c,tgs}\}$$

$$A_c = \{c,addr,time\}$$

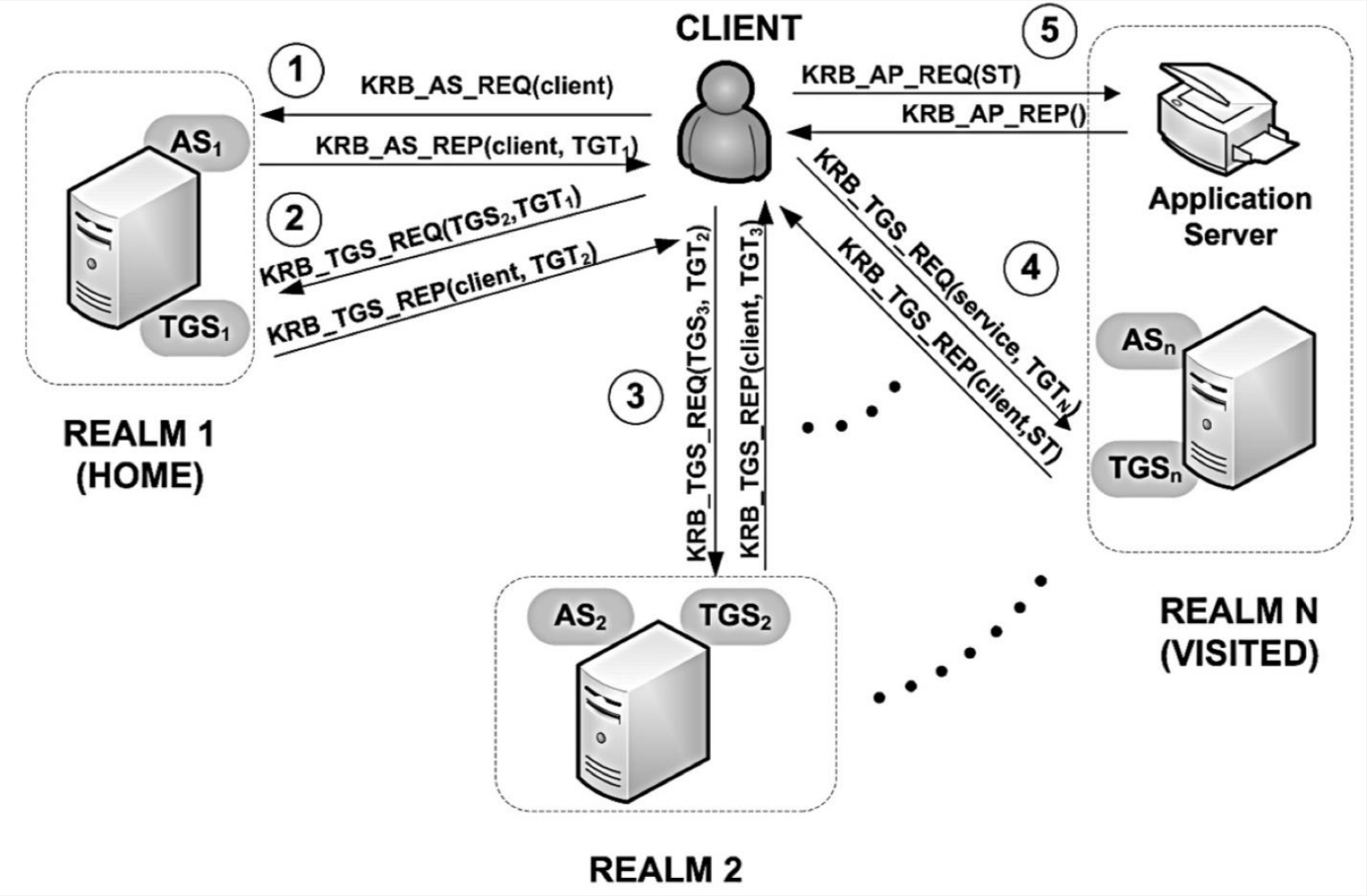
Kerberos authentication model

- Access to a service (with mutual authentication) :



$$T_{c,s} = \{c,s,addr,timestamp,life,K_{c,s}\}$$
$$A_c = \{c,addr,time\}$$

Kerberos cross-realm operation



Session Key Lifetime

For connection-oriented protocols one choice is to use the same session key for the length of time that the connection is open, using a new session key for each new session

A security manager must balance competing considerations:

- The more frequently session keys are exchanged, the more secure they are
- The distribution of session keys delays the start of any exchange and places a burden on network capacity

For a connectionless protocol there is no explicit connection initiation or termination, thus it is not obvious how often one needs to change the session key

Simple Use of Public-Key Encryption to Establish a Session Key

- One of the most important uses of a public-key cryptosystem is to encrypt secret keys for distribution
- Model is insecure against an adversary who can intercept messages (man-in-the-middle attack)

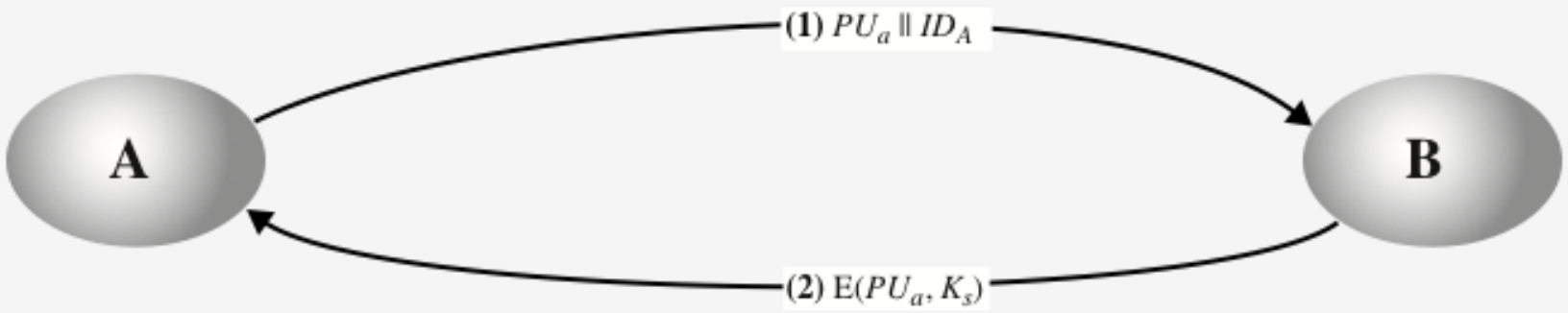


Figure 14.7 Simple Use of Public-Key Encryption to Establish a Session Key

(An obvious) Man-in-the-Middle Attack

- The result is that Alice and Bob both share K_s, and are unaware that K_s has already been revealed to Darth
- Only useful in an environment where the only threat is eavesdropping (traffic analysis)

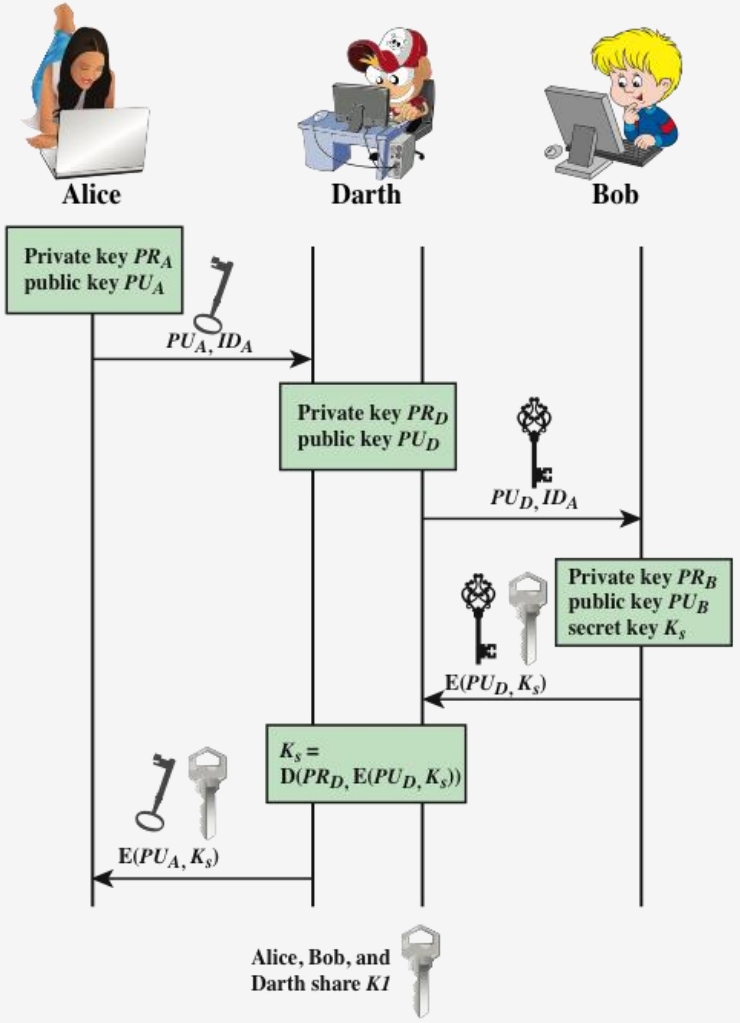


Figure 14.8 Another Man-in-the-Middle Attack

Secret Key distribution with Confidentiality and Authentication

- The model assumes that A and B already share public keys
- Ensures both confidentiality and authentication in the exchange of secret keys
- *Nonces* once again protect against replay attacks (used to identify transactions uniquely)

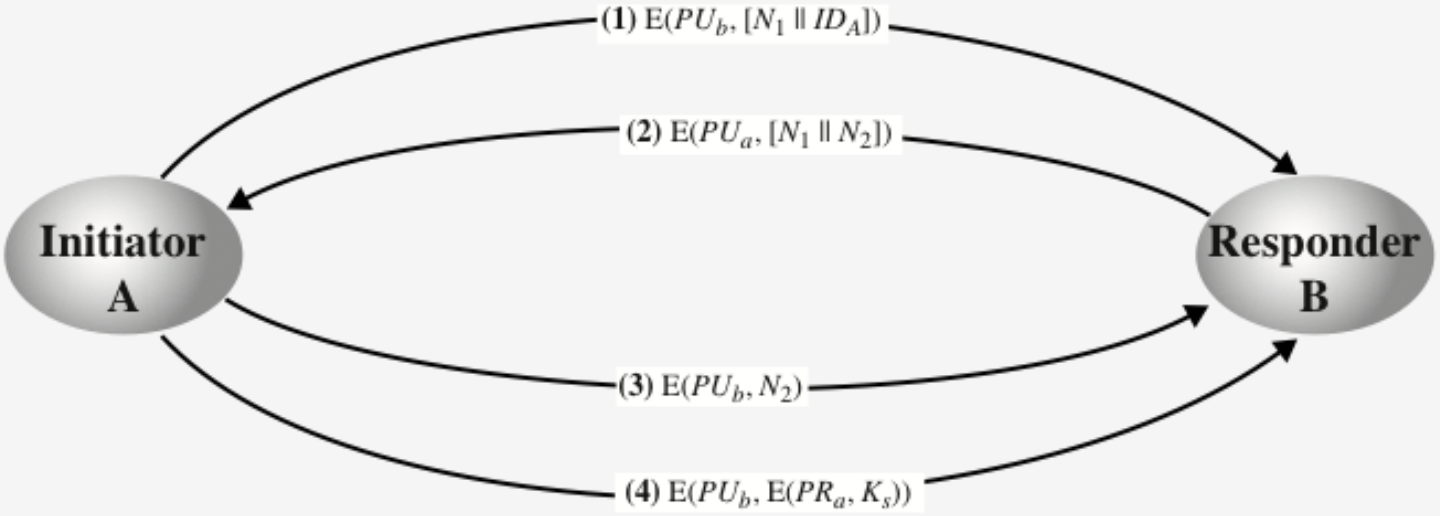
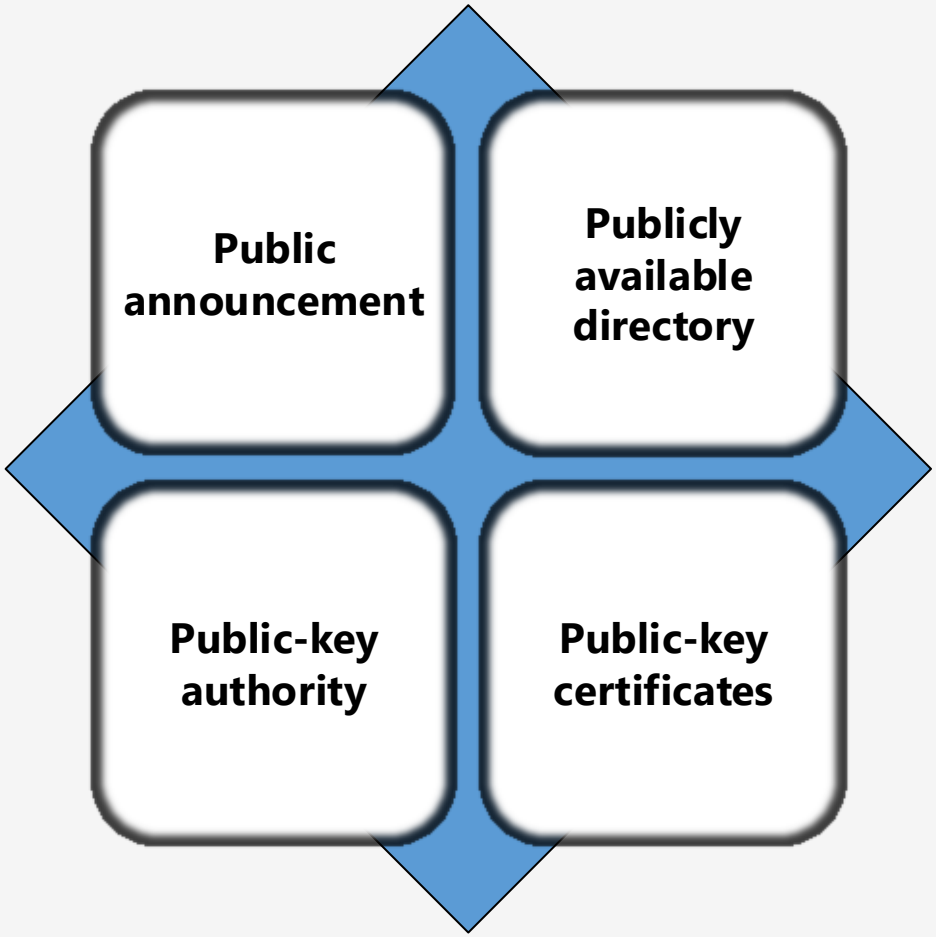


Figure 14.9 Public-Key Distribution of Secret Keys

Distribution of Public Keys

- Several techniques have been proposed for the distribution of public keys. Virtually all these proposals can be grouped into four general schemes



Uncontrolled Public Key Distribution (public announcement)

- Public announcement of public keys, for example as in PGP
- Announcements can be forged (that's why PGP requires manual validation and implements trust)

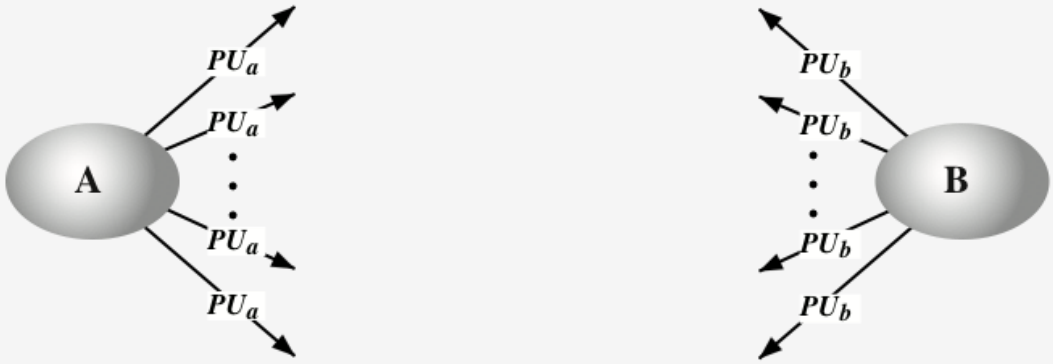


Figure 14.10 Uncontrolled Public Key Distribution

Public Key Publication

- Maintenance of Public Key Directory can be the responsibility of a trusted entity
- Registration can be in person using some form of authentication
- Vulnerable if adversary obtains or computes the private key of the directory authority

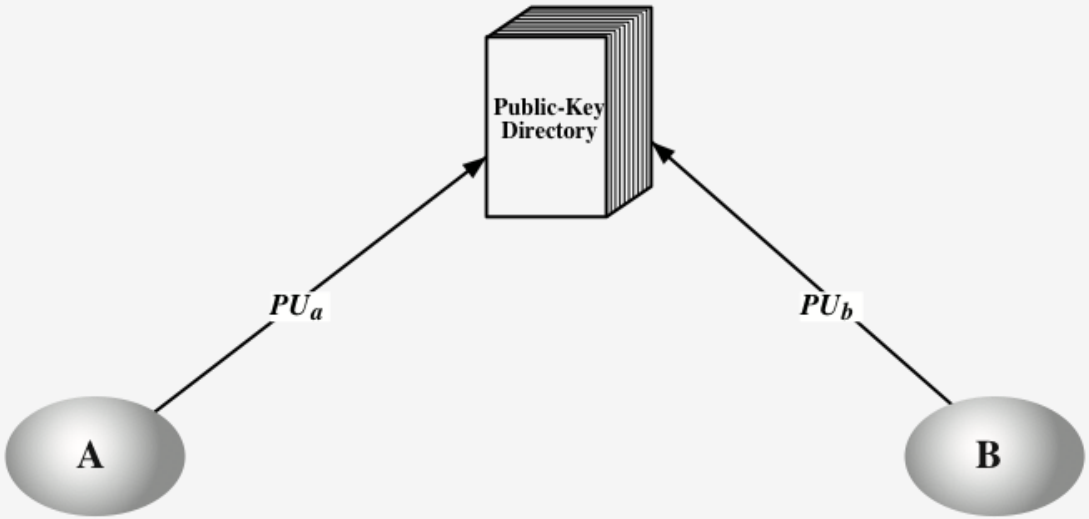
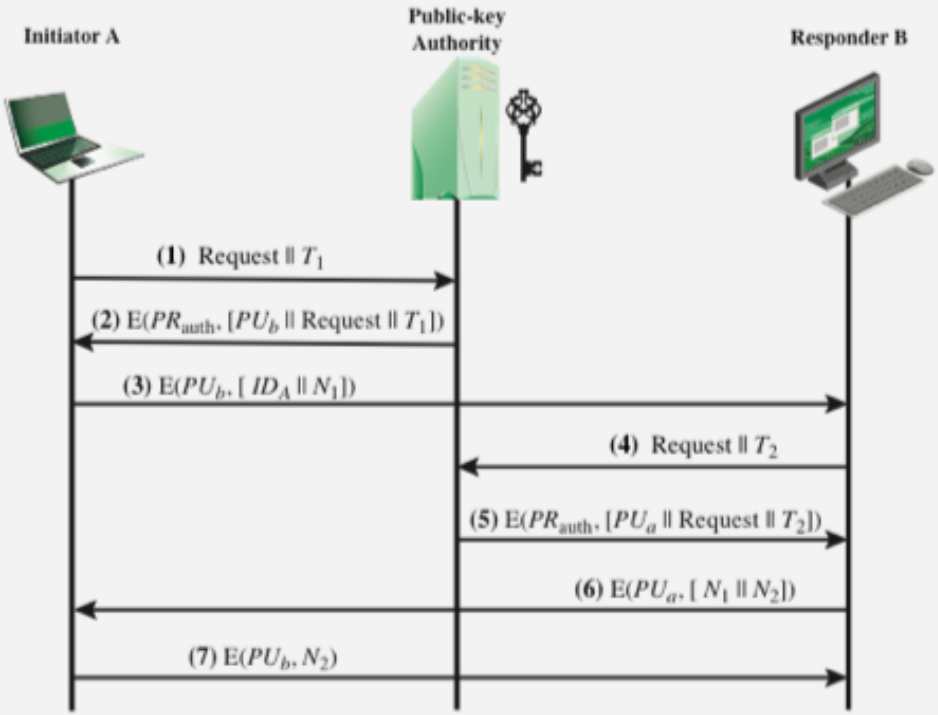


Figure 14.11 Public Key Publication

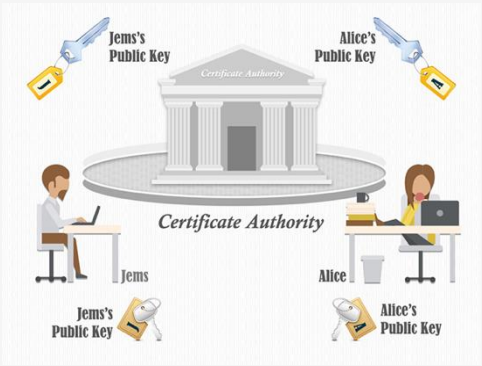
Public-Key Authority

- Implements stronger control over distribution of public keys from the directory
- Each participant known the public key of the authority
- *Nonce* values (or timestamps) are again used for securing against replay attacks (messages *1* and *2*, *4* and *5*)
- N1 assures A that B is the correspondent
- N2 assures B that the correspondent is A



Public-Key Certificates

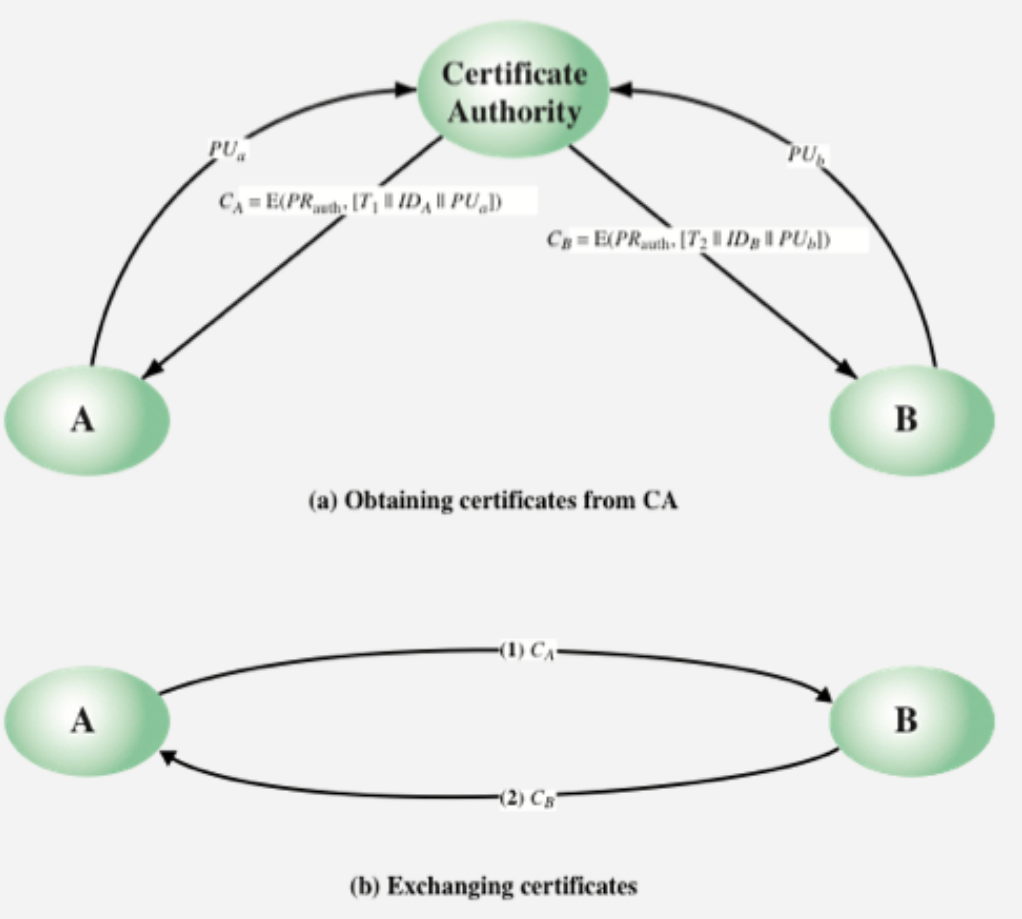
- Public-key Authorities can be a bottleneck and are vulnerable to tampering
- Certificates can be used to exchange Public Keys, without contacting a Public-Key Authority
- In essence, a certificate consists of a public key, an identifier of the key owner and the whole block signed by a **trusted third party** (typically a certificate authority)
- The validity of the user's public keys can be verified by way of the attached signature, and validate that the certificate was created by a trusted authority



Exchange of Public-key Certificates

Main principles:

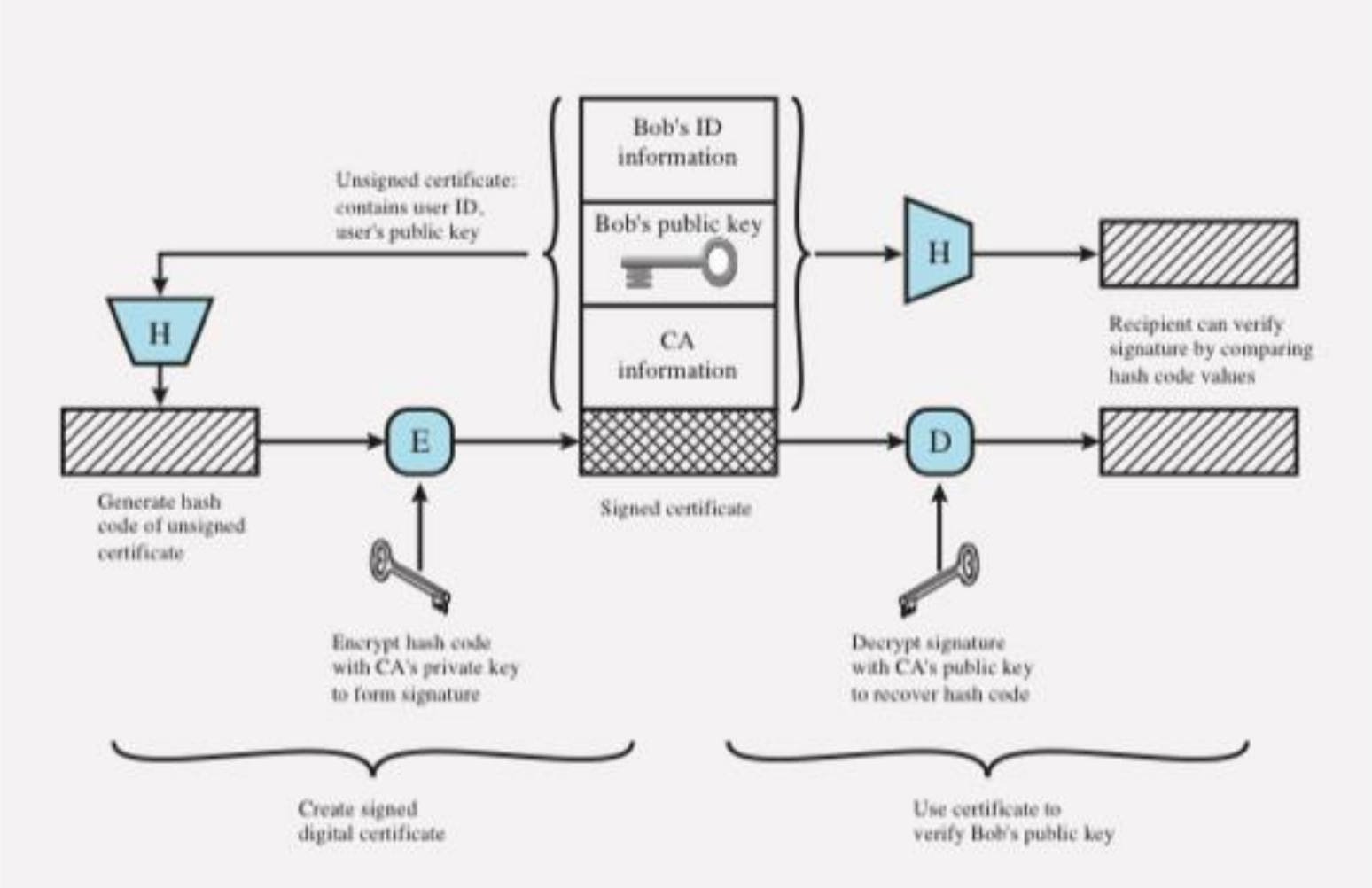
- Any participant can read a certificate to determine the name and public key of the certificate's owner
- Any participant can verify that the certificate originated from the CA, its time validity and revocation status
- Only the CA can create and update certificates
- X.509 standard is the accepted standard for formatting certificates



X.509 Certificates

- Part of the X.500 series of recommendations that define a directory service
  - ✓ The directory is, in effect, a server or distributed set of servers that maintains a database of information about users
- X.509 defines a framework for the provision of authentication services by the X.500 directory to its users
  - ✓ Was initially issued in 1988 with the latest revision in 2012
  - ✓ Based on the use of public-key cryptography and digital signatures
  - ✓ Does not dictate the use of a specific algorithm but recommends RSA
  - ✓ Does not dictate a specific hash algorithm
- Each certificate contains the public key of a user and is signed with the private key of a trusted certification authority
- X.509 defines alternative authentication protocols based on the use of public-key certificates

Public-Key Certificate Usage



X.509 Certificates

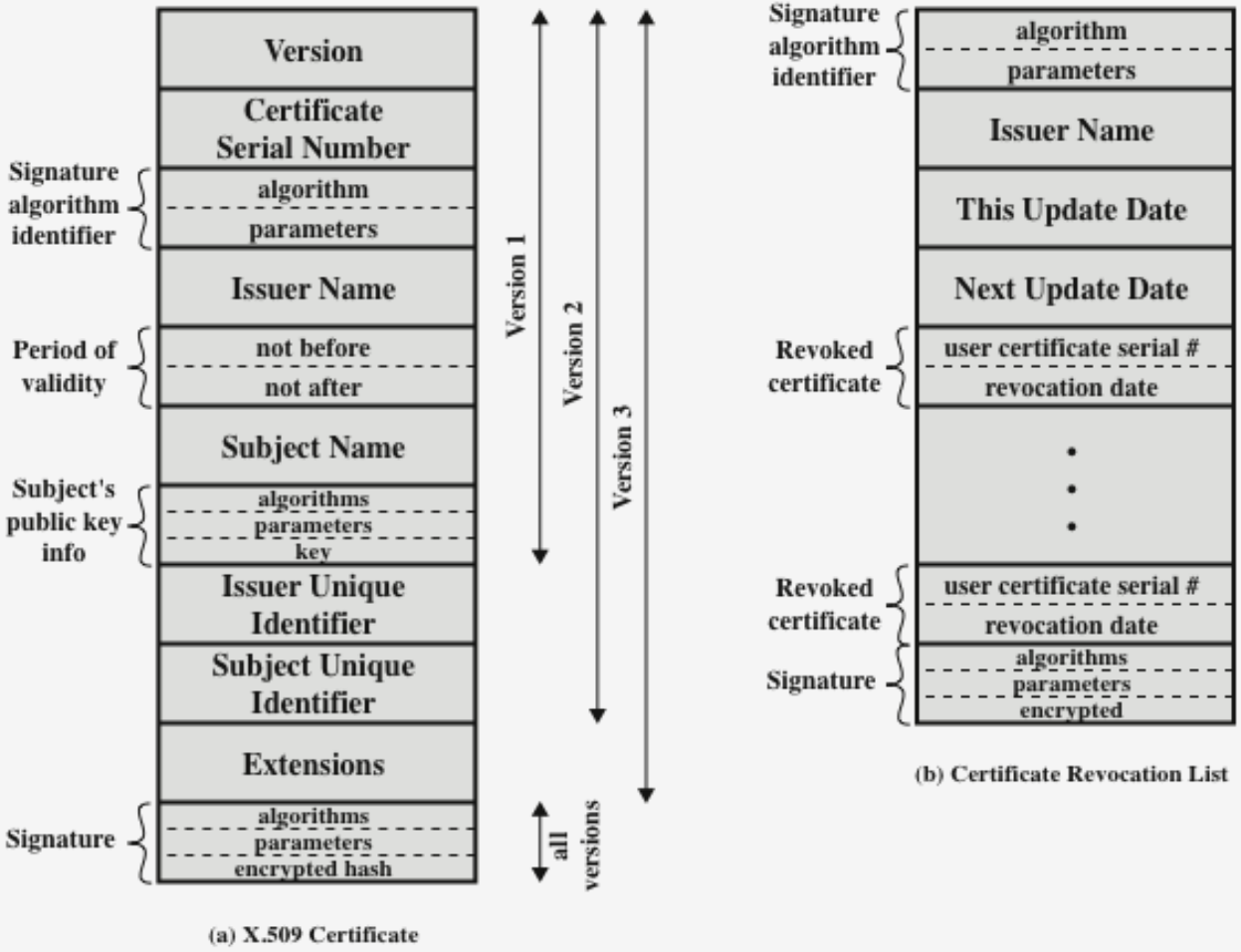


Figure 14.15 X.509 Formats

Obtaining a Certificate

- Because certificates are unforgeable, they can be placed in a directory without the need for the directory to make special efforts to protect them
- In addition, a user can transmit his or her certificate directly to other users
- Once B is in possession of A's certificate, B has confidence that messages it encrypts with A's public key will be secure from eavesdropping and that messages signed with A's private key are unforgeable
- Each participating user must have a copy of the CA's own public key to verify signatures, and public keys must be provided a secure way (e.g. OS and browser key stores)

User certificates generated by a CA have the following characteristics:

- Any user with access to the public key of the CA can verify the user public key that was certified
- No party other than the certification authority can modify the certificate without this being detected

Certificate chain strategy

- Suppose that **A has obtained a certificate from CA X_1**
- Suppose that **B has obtained a certificate from CA X_2**
- Without a relation between X_1 and X_2, A cannot verify B's certificate
- CAs X_1 and X_2 may securely exchange their own public keys
- The following procedure will **enable A to securely obtain B's public key**
  - A obtains from the directory the certificate of X_2 signed by X_1: $X_1 << X_2 >>$
  - A verified X_2's certificate with public key of X_1 and obtains X_2's public key
  - A obtains the certificate of B signed by X_2
  - A can now verify the signature and securely obtain X_2's public key
  - In the notation of X.509, this chain of certificates is expressed as:

    $$\mathbf{X_1 << X_2 >> X_2 << B >>}$$

  - A chain with N elements would be expressed as:

    $$\mathbf{X_1 << X_2 >> X_2 << X_3 >> .. X_N << B >>}$$

X.509 CA Hierarchy

- Any client validates the certificate if it known (and trusts) the CA's public key
- Pair of CAs can sign each other's certificates supporting a certification hierarchy
- Validation works as long as a certification path is possible
- For example, for User A to establish a certification path to B:

X <<W>> W <<V>> V <<Y>> Y<<Z>>

Z <<B>>

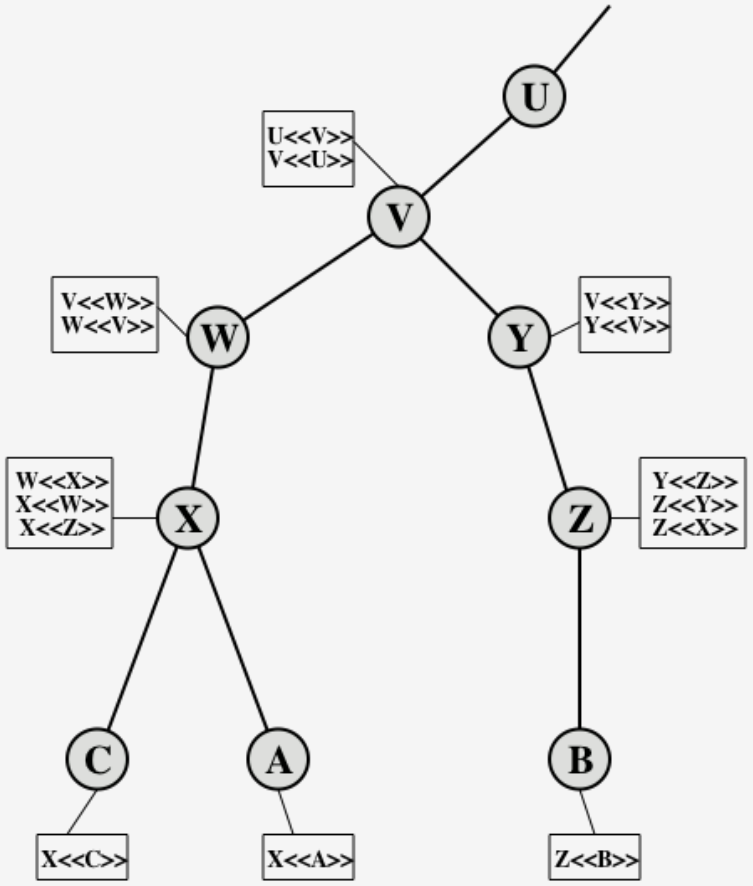
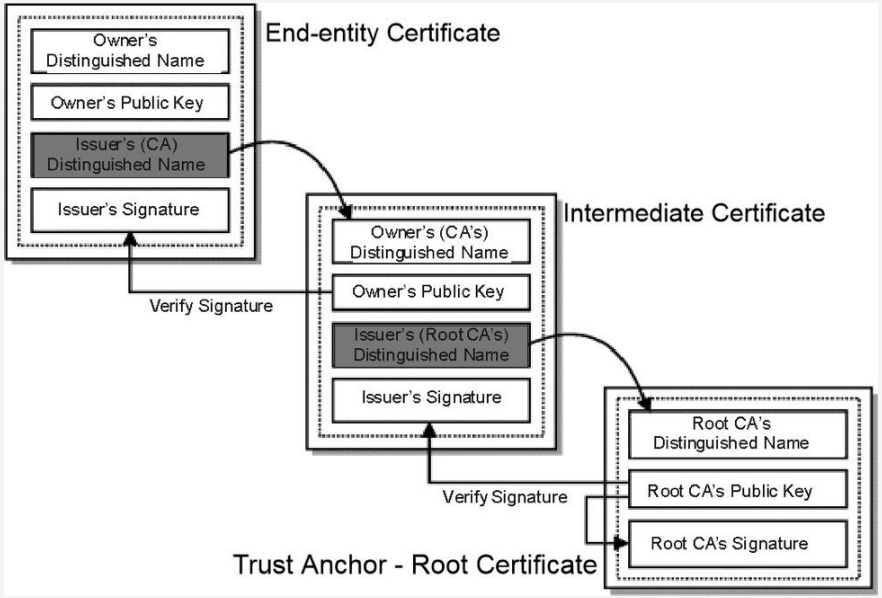


Figure 14.16 X.509 CA Hierarchy: a Hypothetical Example

Validation of a server certificate

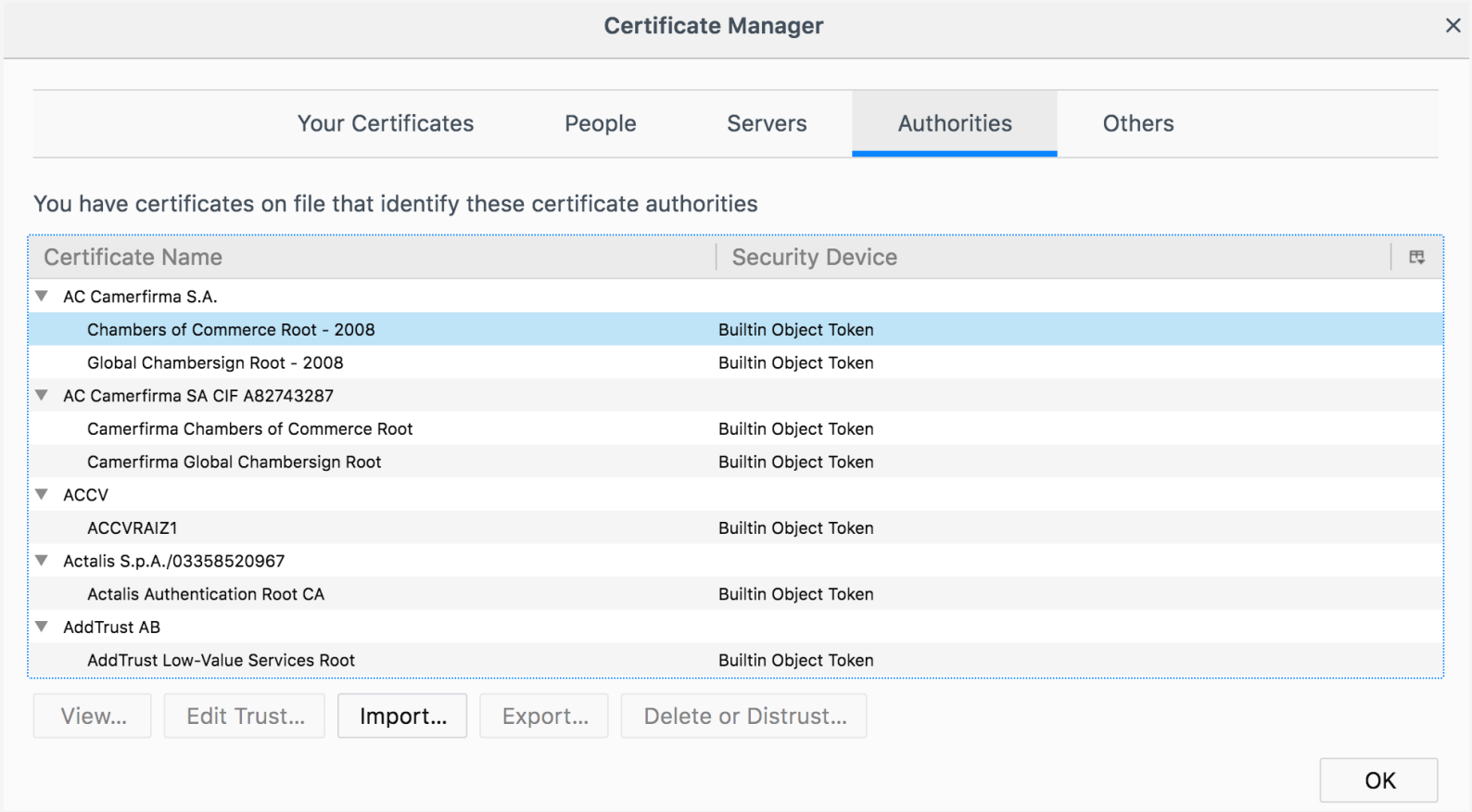
- A client (browser) may receive a single certificate or a certification chain from the server
- A certification chain includes the certificate of the server, intermediate CA certificates and the root CA certificate.
- The certificate chain allows the client to validate the certificate of the server, as long as it trusts the root CA
- Trusted (well-known) CAs are shipped with the browser or OS



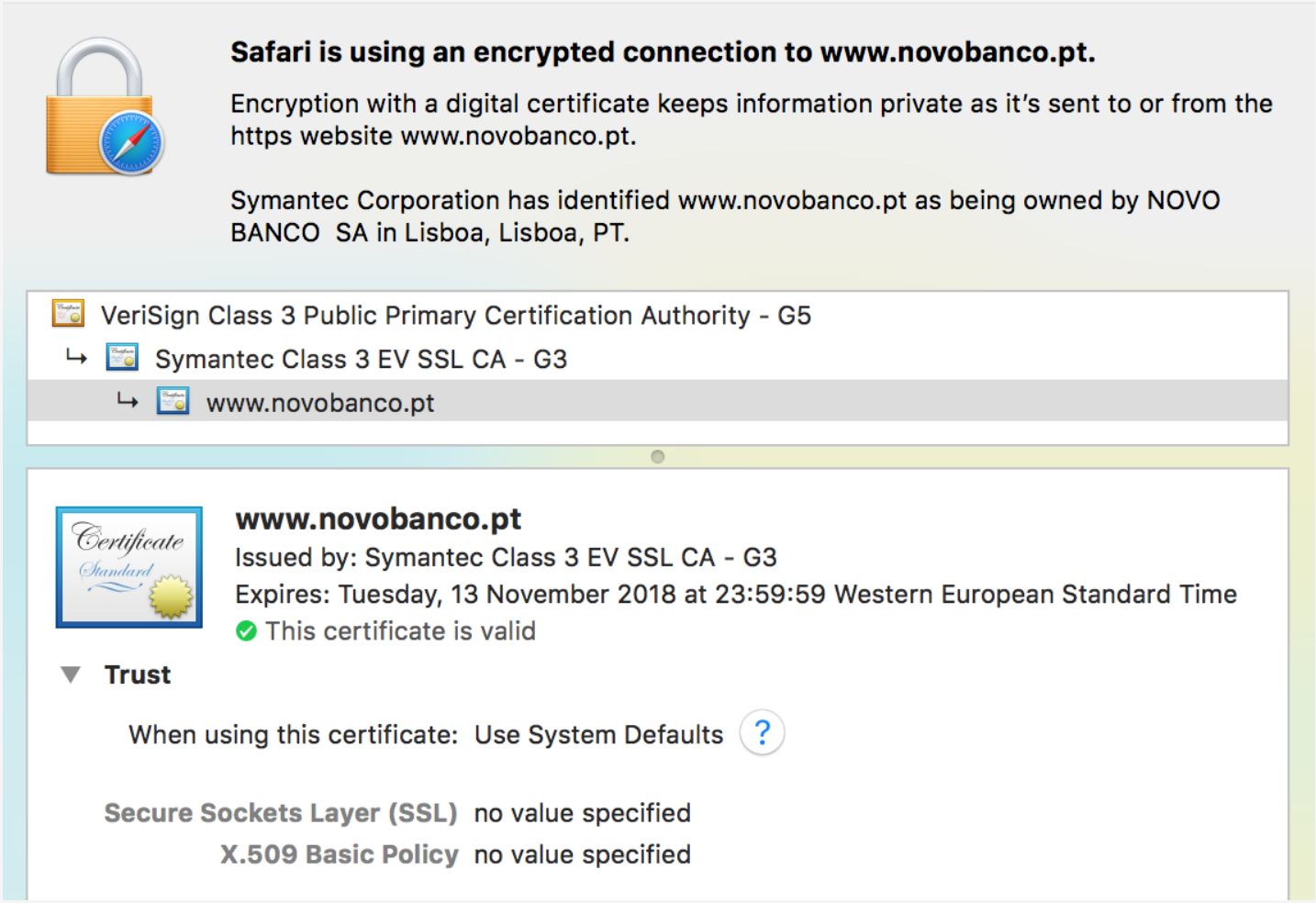

```
#   Server Certificate Chain:
#   Point So SLCertificateChainFile at a file containing the
#   concatenation of PEM encoded CA certificates which form the
#   certificate chain for the server certificate. Alternatively
#   the referenced file can be the same as SSLCertificateFile
#   when the CA certificates are directly appended to the server
#   certificate for convenience.
SSLCertificateChainFile /etc/pki/tls/certs/server-chain.crt
```

X.509 Certificate Validation

- Software client is shipped with list of well known Certification Authorities
- User can add new CAs or personal certificates, as required



X.509 Certificate Validation (hierarchy)

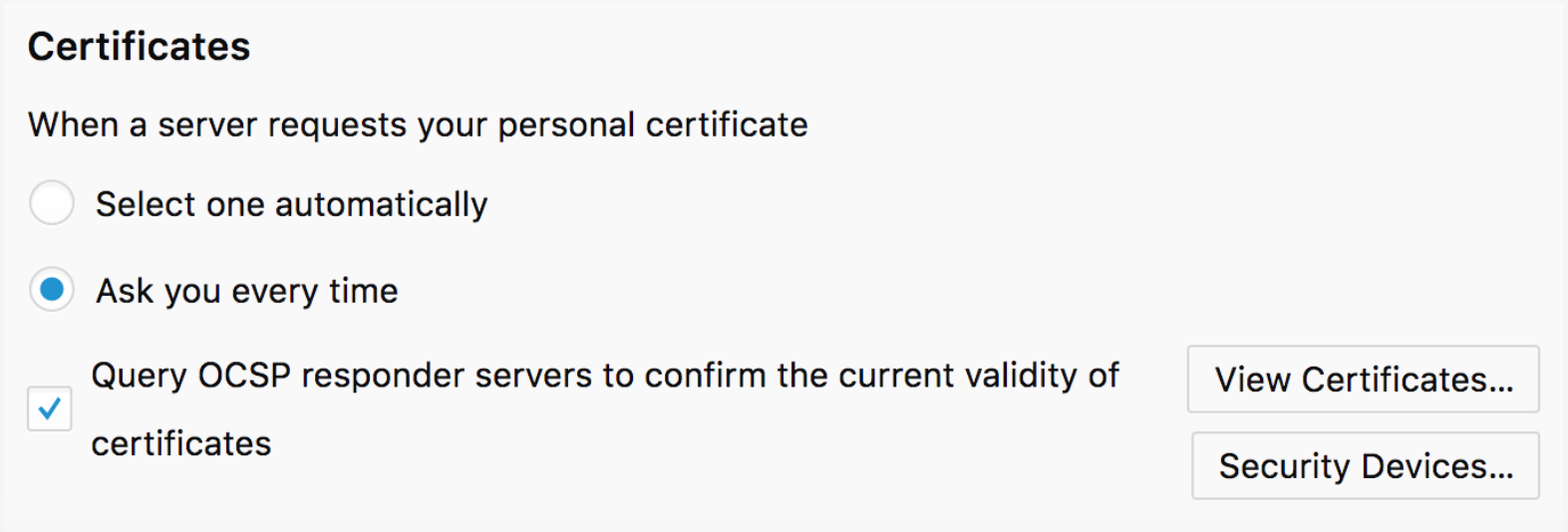


Certificate Revocation

- Each certificate includes a period of validity
  - ✓ Typically a new certificate is issued just before the expiration of the old one
- It may be desirable on occasion to revoke a certificate before it expires, for one of the following reasons:
  - ✓ The user's private key is assumed to be compromised
  - ✓ The user is no longer certified by this CA
  - ✓ The CA's certificate is assumed to be compromised
- Each CA must maintain a list consisting of all revoked but not expired certificates issued by that CA
  - ✓ These lists should be posted on the directory

OCSP (Online Certificate Status Protocol)

- An Internet protocol used for obtaining the revocation status of an X.509 digital certificate
- Alternative to CRL (Certificate Revocation Lists)
- OCSP messages are encoded in ASN.1 and communicated over HTTP (HTTPS)



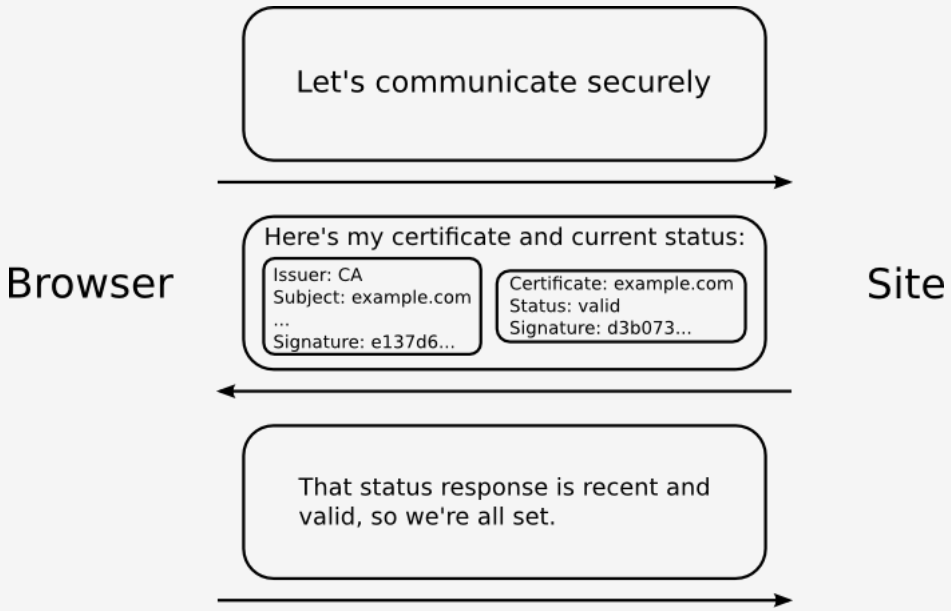
OCSP (Online Certificate Status Protocol)

The address of the OCSP server can be configured manually in the browser or included as part of the X.509 certificate by the CA

Extension	Certificate Authority Information Access (1.3.6.1.5.5.7.1.1)
Critical	NO
Method #1	Online Certificate Status Protocol (1.3.6.1.5.5.7.48.1)
URI	http://sr.symcd.com
Method #2	CA Issuers (1.3.6.1.5.5.7.48.2)
URI	http://sr.symcb.com/sr.crt

OCSP stapling

- An alternative approach to the Online Certificate Status Protocol (OCSP) for checking the revocation status of X.509 digital certificates
- The presenter of a certificate bears the resource cost involved in providing OCSP responses
- Server appends a time-stamped OCSP response signed by the CA to the initial TLS handshake, eliminating the need for clients to contact the CA



PKIX Architectural Model

- PKI (Public-Key Infrastructure) is the set of hardware, software, people, policies and procedures needed to create, manage, store, distribute, and revoke digital certificates.
- PKI allow for secure, convenient and efficient acquisition of public-keys (certificates).
- The IETF Public-Key Infrastructure X.509 (PKIX) working group defined a model for the deployment of a certificate-based architecture.

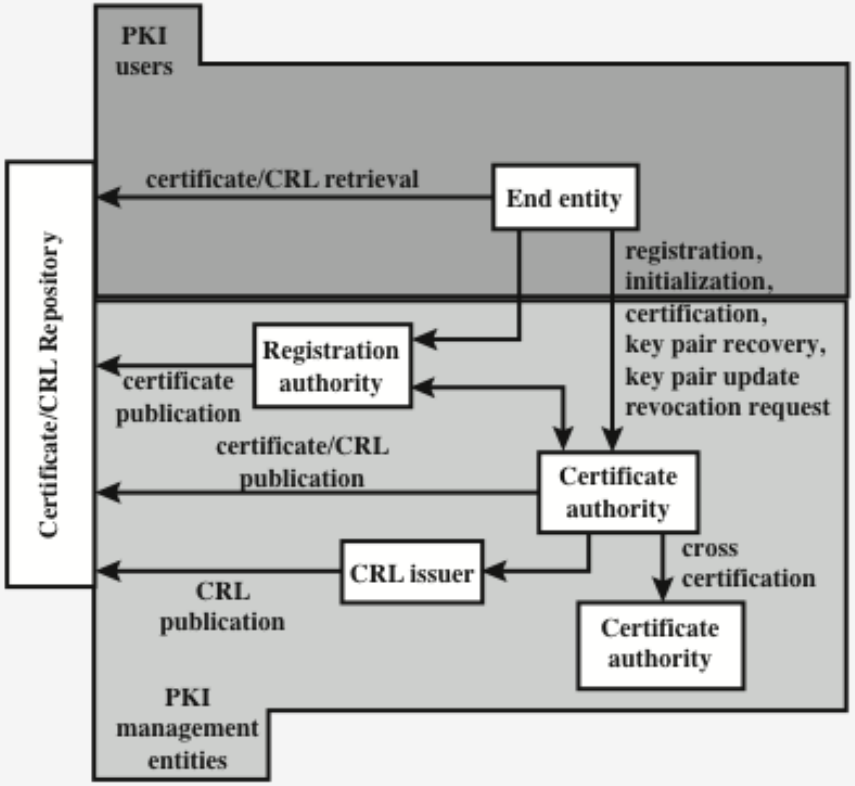


Figure 14.17 PKIX Architectural Model

PKIX Management Functions

PKIX identifies a number of management functions that potentially need to be supported by management protocols:

- ✓Registration
- ✓Initialization
- ✓Certification
- ✓Key pair recovery
- ✓Key pair update
- ✓Revocation request
- ✓Cross certification

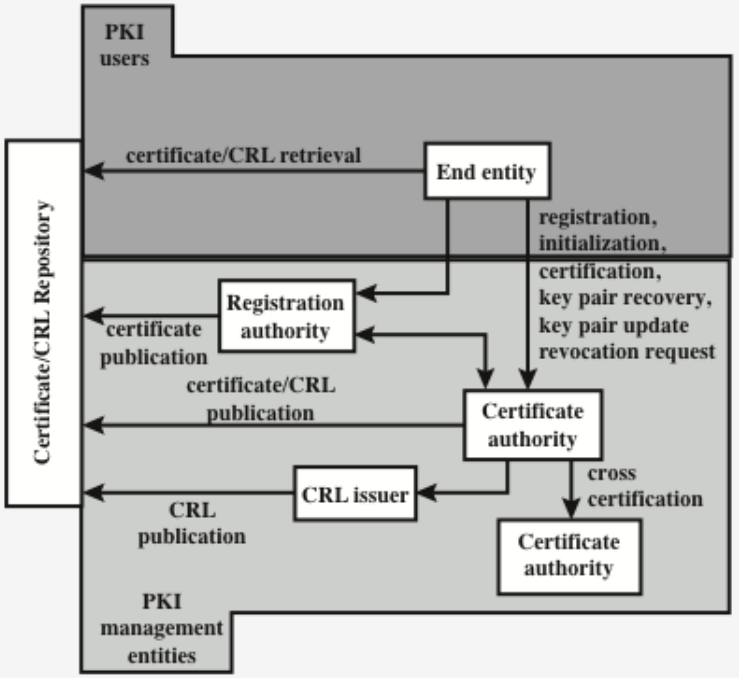


Figure 14.17 PKIX Architectural Model

Review Questions

What is a chain of certificates?

A chain of certificates consists of a sequence of certificates created by different certification authorities (CAs) in which each successive certificate is a certificate by one CA that certifies the public key of the next CA in the chain

How is an X.509 certificate revoked?

The owner of a public-key can issue a certificate revocation list that revokes one or more certificates. This list (the CRL) is signed by the CA and periodically updated.

What is the purpose of the OCSP protocol?

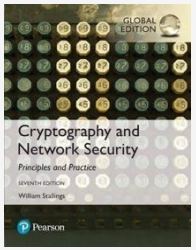
To facilitate the dissemination of revocation information in the context of a given PKI. Contrary to CRL, OCSP enabled the online verification of the validity status of a given certificate

Summary

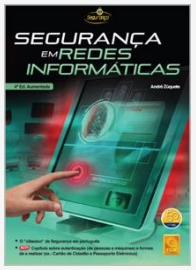
- Symmetric key distribution using symmetric encryption
  - ✓ Key distribution scenario
  - ✓ Hierarchical key control
  - ✓ Key distribution with Kerberos
  - ✓ Session key lifetime
  - ✓ Transparent key control scheme
  - ✓ Decentralized key control
- Symmetric key distribution using asymmetric encryption
  - ✓ Simple secret key distribution
  - ✓ Secret key distribution with confidentiality and authentication
- Distribution of public keys
  - ✓ Public announcement of public keys
  - ✓ Publicly available directory
  - ✓ Public-key authority
  - ✓ Public-key certificates
- Public-key infrastructures (PKIX)
  - ✓ PKIX management functions
  - ✓ PKIX management protocols

Bibliography

Cryptography and network security, Stallings, Pearson, 2017, Chapter 14: Key Management and Distribution



Segurança em Redes Informáticas, Capítulo 2: Criptografia, Capítulo 3: Gestão de Chaves Públicas



Applied Cryptography, Bruce Schneier, Chapter 2: Protocol Building Blocks, Chapter 3: Basic Protocols