Ceng 471 Cryptography

Asst. Prof. Dr. Serap ŞAHİN

Key Distribution and Management

Ref: Lecture slides by Lawrie Brown, "Cryptography and Network Security", Ch.14*, 5th Ed. By William Stalling*

Key Management and Distribution

No Singhalese, whether man or woman, would venture out of the house without a bunch of keys in his hand, for without such a talisman he would fear that some devil might take advantage of his weak state to slip into his body.

—The Golden Bough, Sir James George Frazer

Key Management and Distribution

- Topics of cryptographic key management / key distribution are complex
  - cryptography, protocol, & management issues
- Symmetric schemes require both parties to share a common secret key
- Public key schemes require parties to acquire valid public keys
- have concerns with doing both

Key Distribution

- Symmetric schemes require both parties to share a common secret key
- Issue is how to securely distribute this key
- Whilst protecting it from others
- Frequent key changes can be desirable
- Often secure system failure due to a break in the key distribution scheme

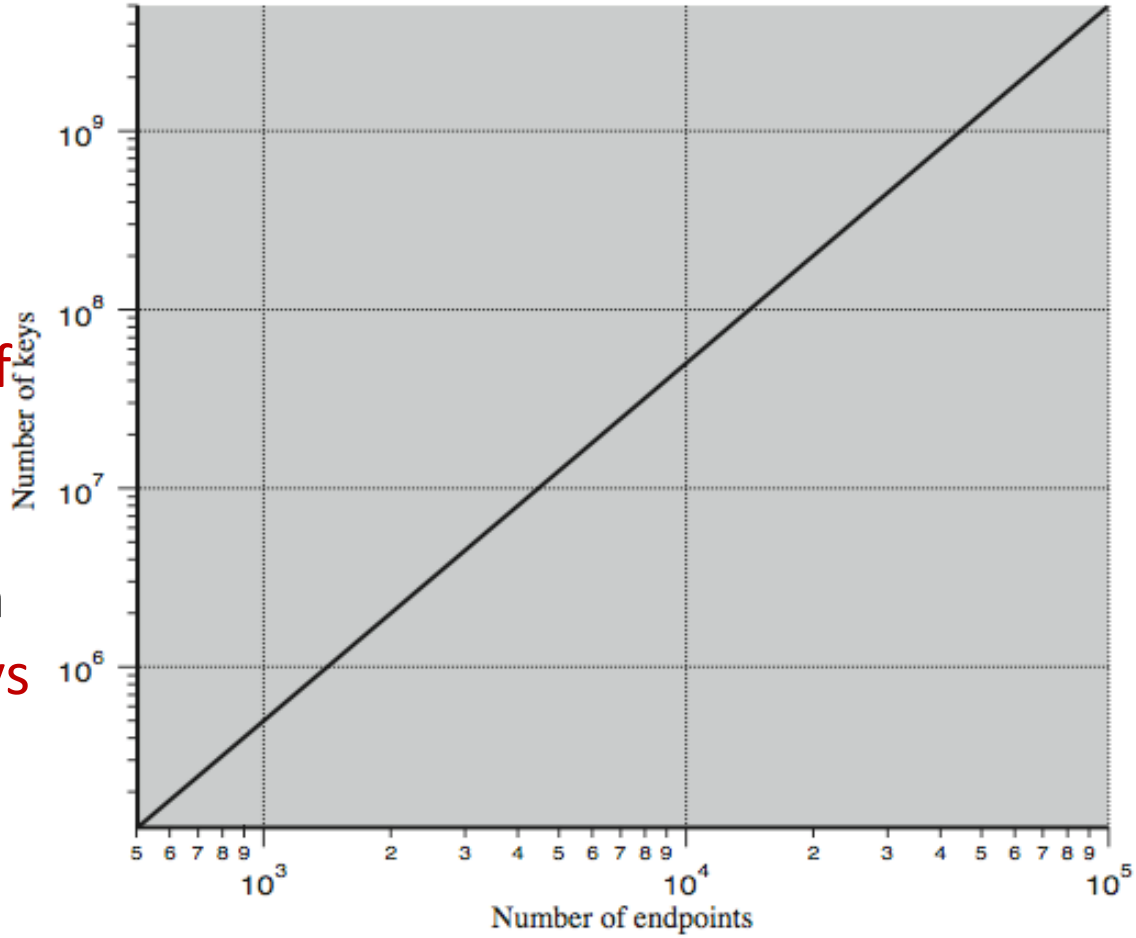
Key Distribution

- given parties A and B have various **key distribution** alternatives:
  1. A can select key and physically deliver to B
  2. third party can select & deliver key to A & B
  3. if A & B have communicated previously can use previous key to encrypt a new key
  4. if A & B have secure communications with a third party C, C can relay key between A & B
     - The third party is called as “trusted intermediary”.

Key Distribution Task

If end-to-end encryption is done at a network or IP level, then a key is needed for each pair of hosts on the network that wish to communicate. Thus, if there are *N* hosts, the number of required keys is *[N(N – 1)]/2*.

A network using node-level encryption with 1000 nodes would conceivably need to distribute as many as half a million keys. If that same network supported 10,000 applications, then as many as 50 million keys may be required for application-level encryption.



Key Hierarchy

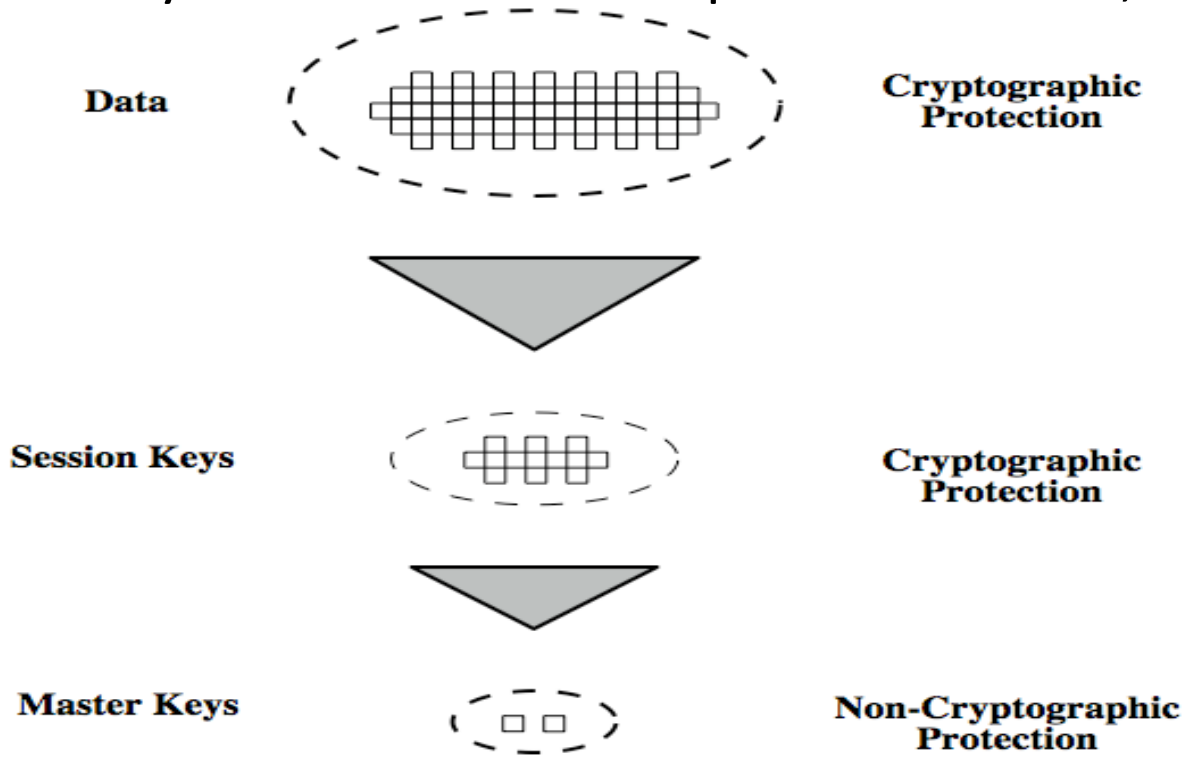
- A key distribution center is responsible for distributing keys to pairs of users; hosts, processes, applications.
- Each user must share a unique key with the key distribution center for purposes of key distribution.
- The use of a key distribution center is based on the use of a hierarchy of keys.
- At a minimum, two levels of keys are used:
  - a session key, used for the duration of a logical connection; and
  - a master key shared by the key distribution center and an end system or user and used to encrypt the session key.

Key Hierarchy

- typically have a hierarchy of keys
- session key
  - temporary key
  - used for encryption of data between users
  - for one logical session then discarded
- master key
  - used to encrypt session keys
  - shared by user & key distribution center

Key Hierarchy

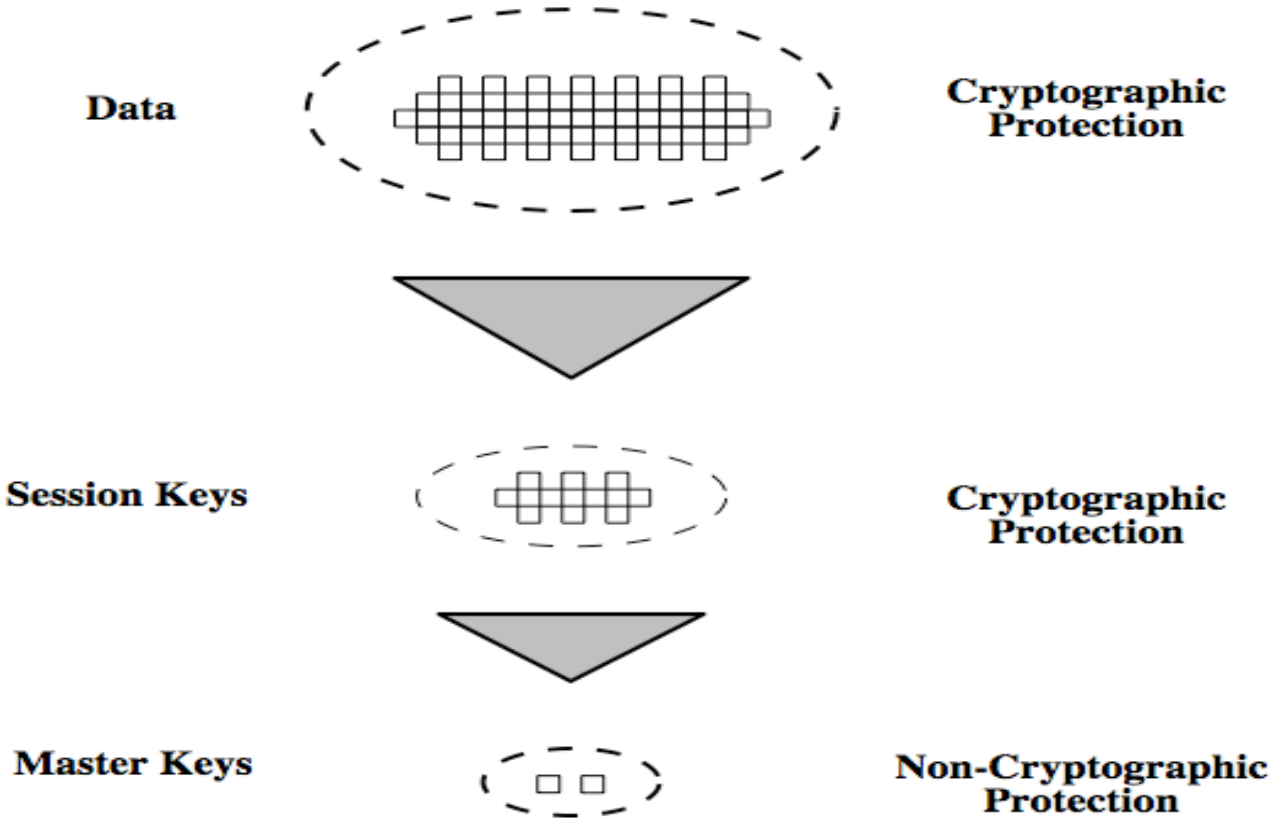
- The use of a key distribution center is based on the use of a hierarchy of key.
- Communication between end systems is encrypted using a temporary key, often referred to as a **session key**.
- Typically, the session key is used for the duration of a logical connection, such as a frame relay connection or transport connection, and then discarded.



Key Hierarchy

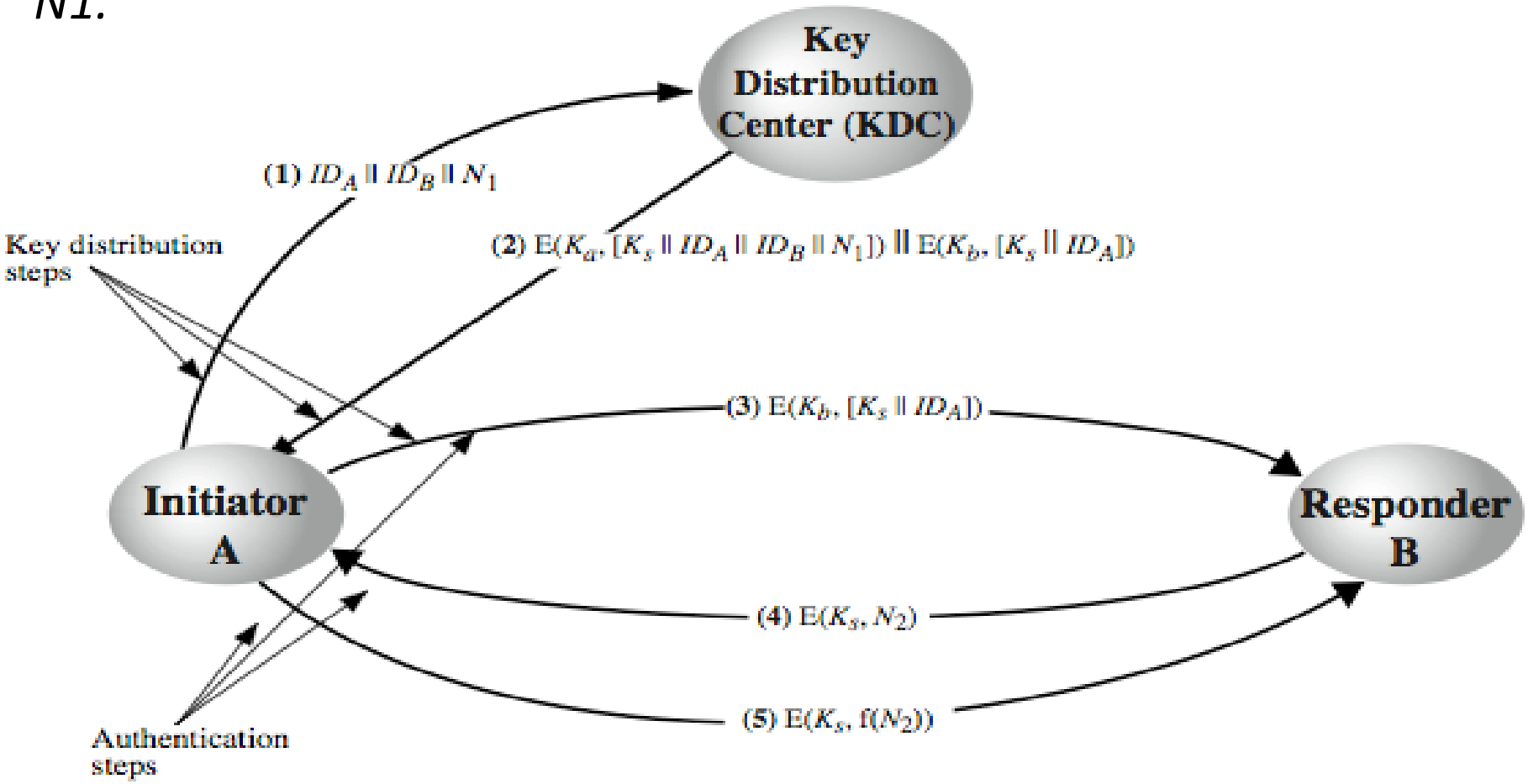
Session keys are transmitted in encrypted form, using a **master key** that is shared by the key distribution center and an end system or user.

Only *N* master keys are required, one for each entity. Thus, master keys can be distributed in some non-cryptographic way, such as physical delivery.



Key Distribution Scenario

A requests from the KDC a session key to protect a logical connection to B. The message includes the identity of A and B and a unique *nonce N1.*



Key Distribution Issues

- Hierarchies of KDC's required for large networks, but must trust each other
  - For communication among entities within the same local domain, the local KDC is responsible for key distribution.
  - If two entities in different domains desire a shared key, then the corresponding local KDCs can communicate through a (hierarchy of) global KDC(s)
- To balance security & effort, a new session key should be used for each new connection-oriented session.
  - For a connectionless protocol, a new session key is used for a certain fixed period only or for a certain number of transactions.

Key Distribution Issues

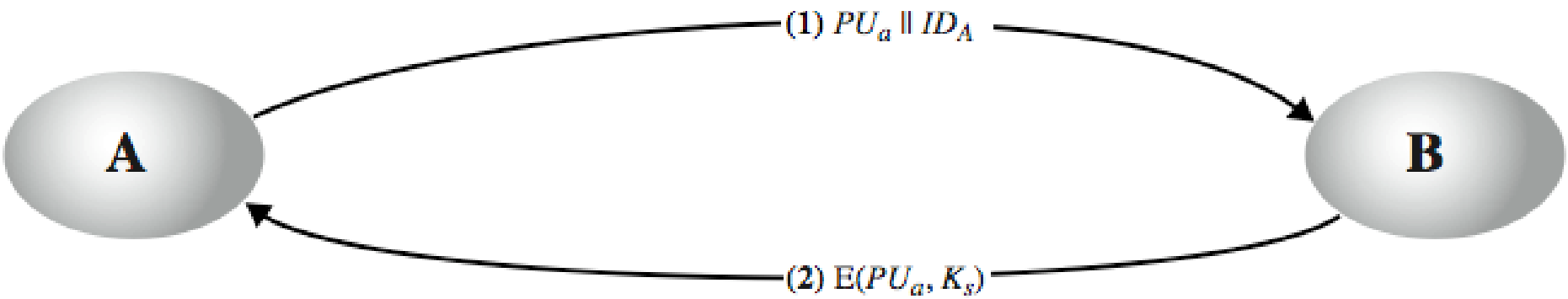
- Session key lifetimes should be limited for greater security
- Use of automatic key distribution on behalf of users, but must trust system
  - An automated key distribution approach provides the flexibility and dynamic characteristics needed to allow a number of terminal users to access a number of hosts and
  - For the hosts to exchange data with each other, provided they trust the system to act on their behalf.
- This requirement can be avoided if key distribution is fully decentralized
- Controlling key usage

Symmetric Key Distribution Using Public Keys

- public key cryptosystems are inefficient
  - so almost never use for direct data encryption
  - rather use to encrypt secret keys for distribution

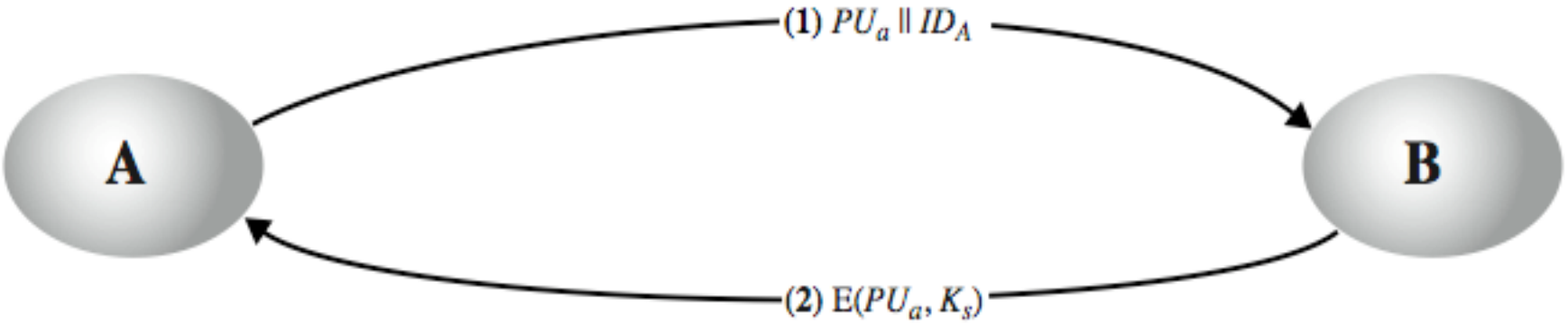
Simple Secret Key Distribution

- Merkle proposed this very simple scheme
  - Allows secure communications
  - This is an attractive protocol. No keys exist before the start of the communication and none exist after the completion of communication.
  - Thus, the risk of compromise of the keys is minimal. At the same time, the communication is secure from eavesdropping.



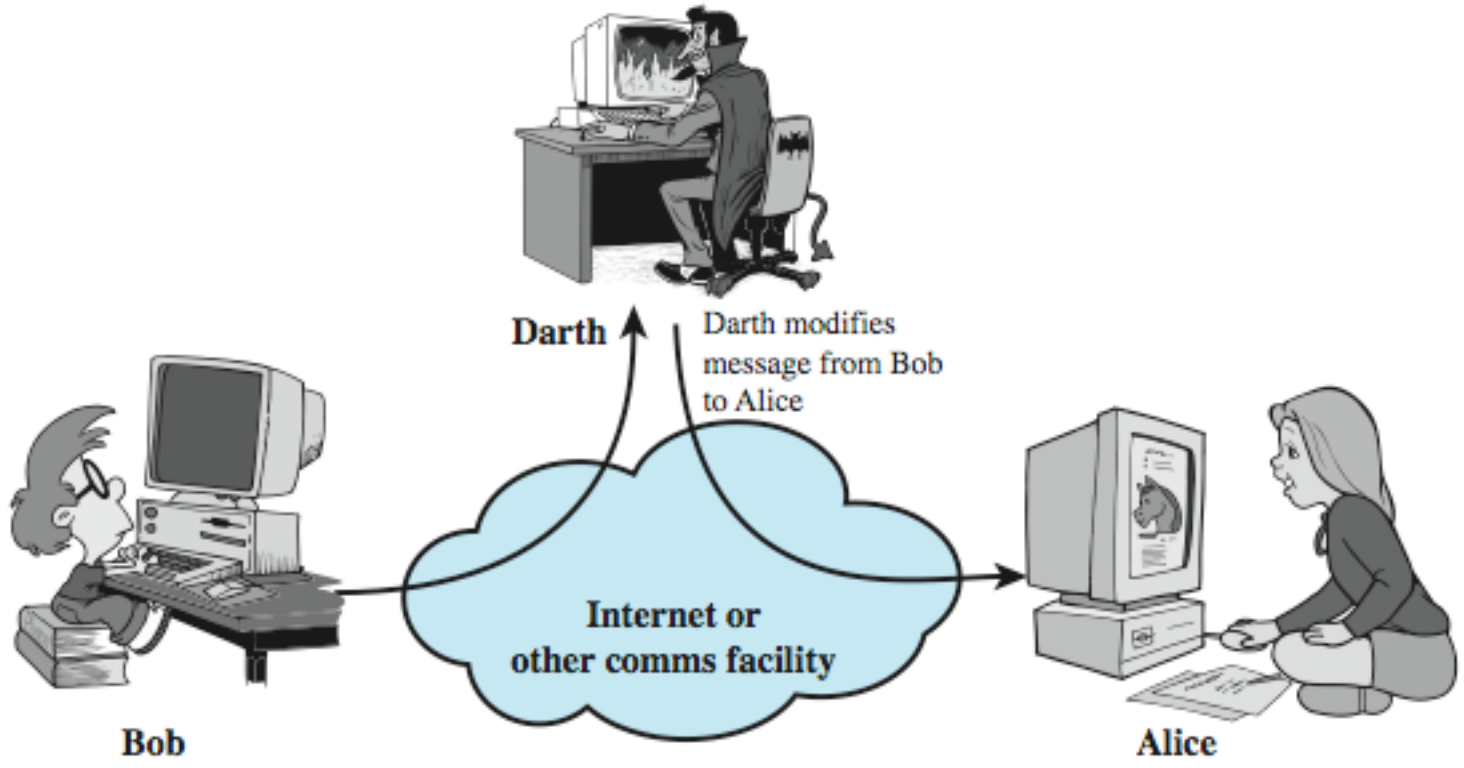
Simple Secret Key Distribution

1. A generates a public/private key pair {*PUa, PRa}* and transmits a message to B consisting of PUa and an identifier of A, ID_A.
2. B generates a secret key, Ks, and transmits it to A, encrypted with A's public key.
3. A computes D(PRa, E(PUa, Ks)) to recover the secret key. Because only A can decrypt the message, only A and B will know the identity of Ks.
4. A discards PUa and PRa and B discards PUa.



Man-in-the-Middle Attack

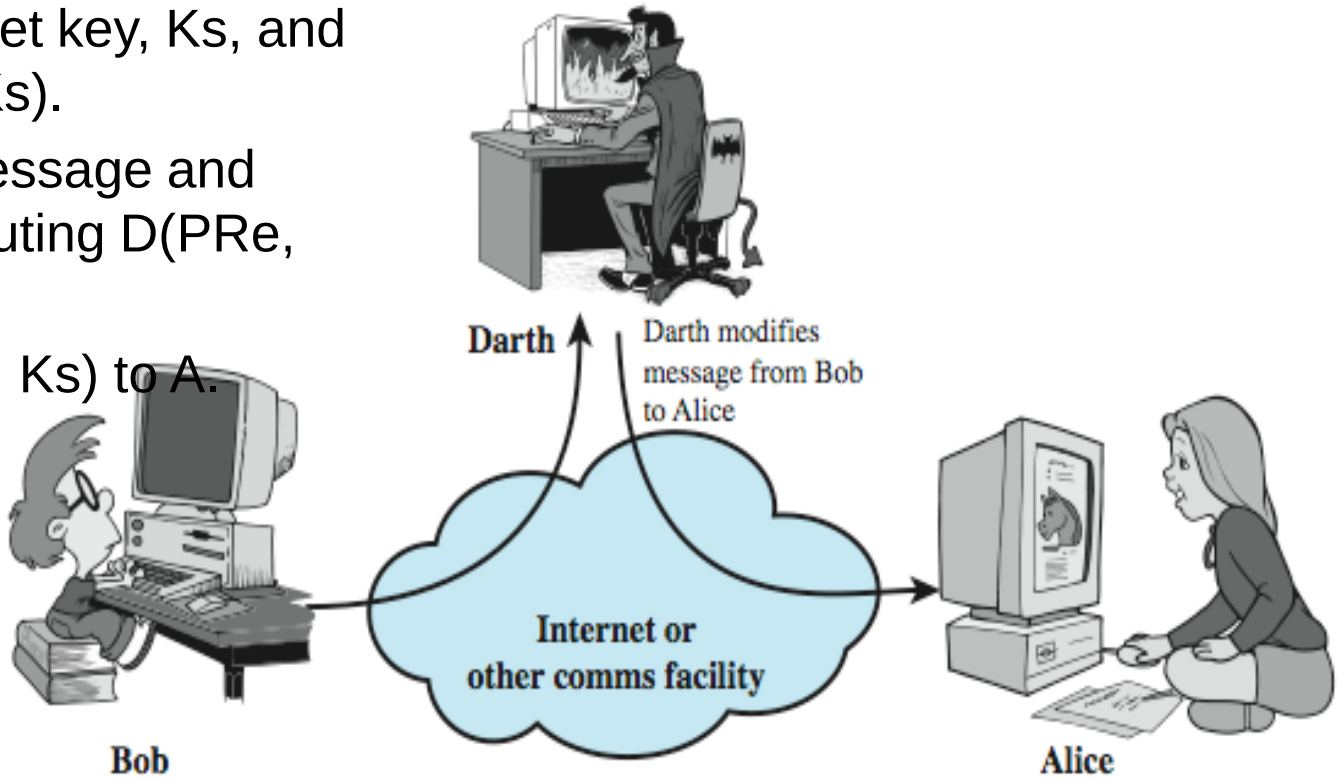
- this very simple scheme is vulnerable to an active man-in-the-middle attack



Man-in-the-Middle Attack

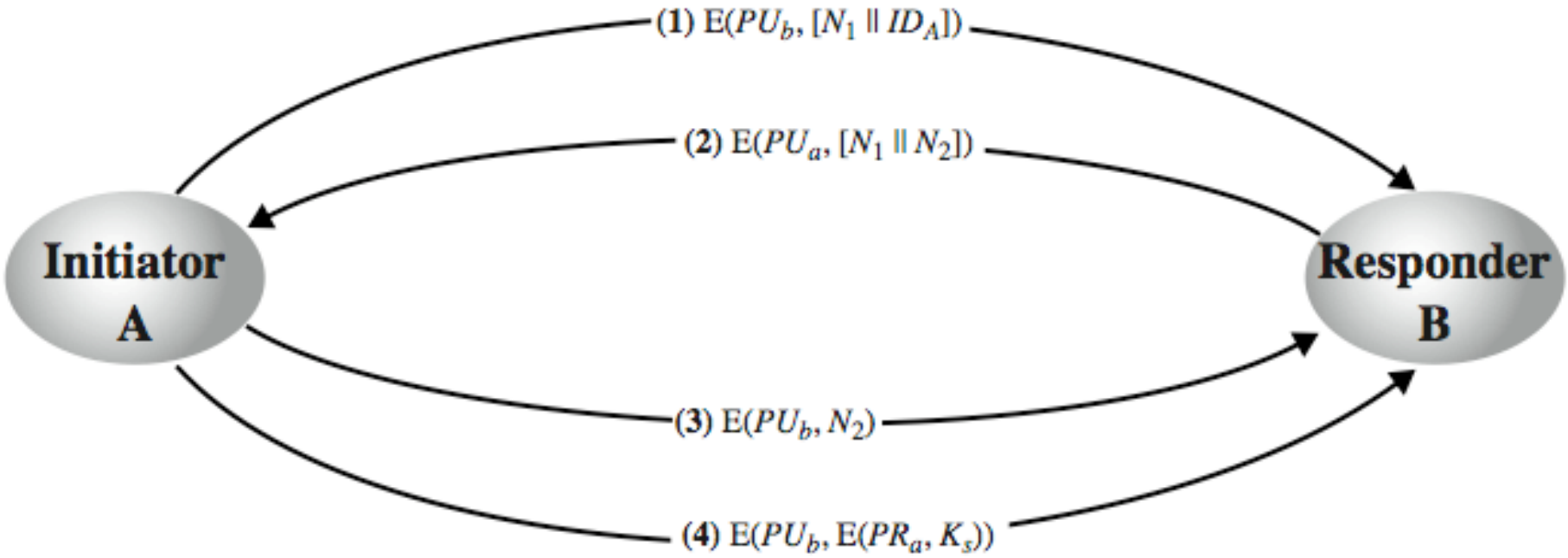
- E, has control of the intervening communication channel, then E can compromise the communication in the following fashion without being detected:

1. A generates a public/private key pair {PUa, PRa} and transmits a message intended for B consisting of PUa and an identifier of A, IDA.
2. E intercepts the message, creates its own public/private key pair {PUe, PRe} and transmits PUe || IDA to B.
3. B generates a secret key, Ks, and transmits E(PUe, Ks).
4. E intercepts the message and learns Ks by computing D(PRe, E(PUe, Ks)).
5. E transmits E(PUa, Ks) to A.



Secret Key Distribution with Confidentiality and Authentication

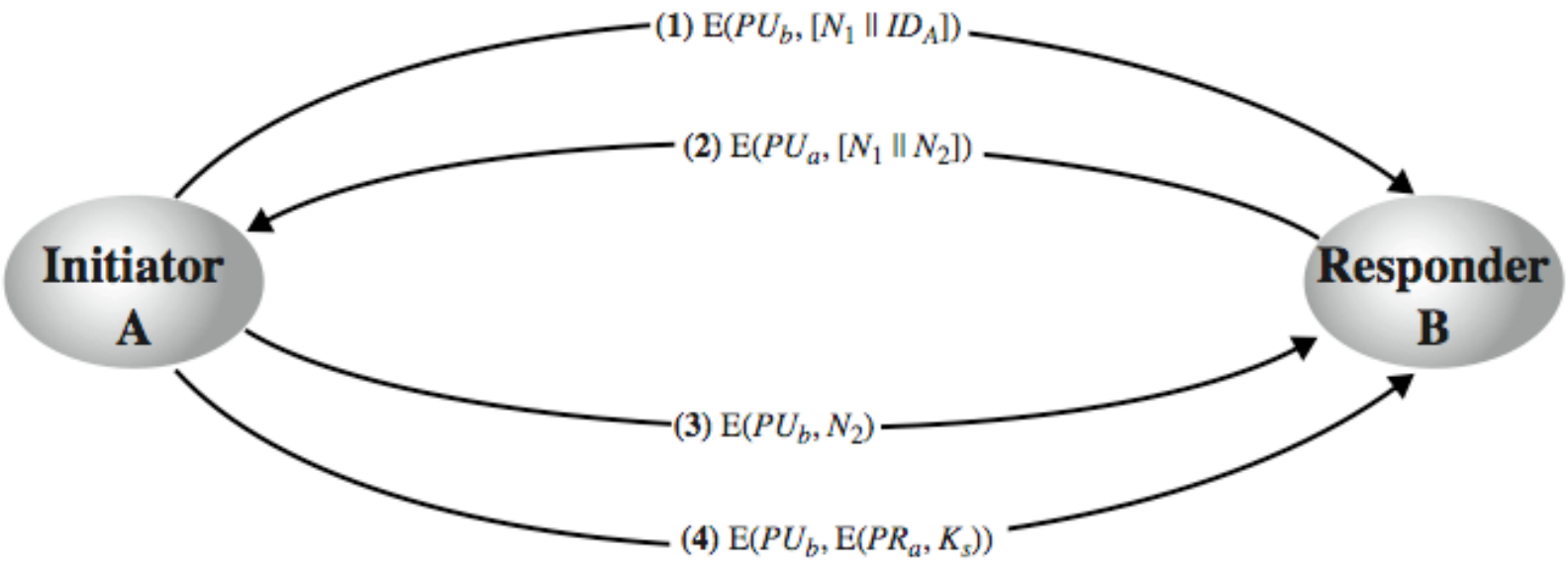
1. A uses B's public key to encrypt a message to B containing an identifier of A (ID_A) and a nonce (N_1), which is used to identify this transaction uniquely.
2. B sends a message to A encrypted with PUa and containing A's nonce (N_1) as well as a new nonce generated by B (N_2). Because only B could have decrypted message (1), the presence of N_1 in message (2) assures A that the correspondent is B.



Secret Key Distribution with Confidentiality and Authentication

3. A returns N_2, encrypted using B's public key, to assure B that its correspondent is A key.
4. A selects a secret key Ks and sends M = E(PUb, E(PRa, Ks)) to B. Encryption with B's public key ensures that only B can read it; encryption with A's private key ensures that only A could have sent it.
5. B computes D(PUa, D(PRb, M)) to recover the secret key.

The result is that this scheme ensures both confidentiality and authentication in the exchange of a secret key.



Hybrid Key Distribution

A hybrid approach in use on IBM mainframes and a key distribution center (KDC) that shares a secret master key with each user and distributes secret session keys encrypted with the master key.

- Retain use of private-key KDC
- Shares secret master key with each user
- Distributes session key using master key
- Public-key used to distribute master keys
  - Especially useful with widely distributed users
- Rationale
  - Performance; a single KDC serves a widely distributed set of users
  - Backward compatibility

Distribution of Public Keys

- Several techniques have been proposed for the distribution of public keys, which can be considered as using one of:
  1. Public announcement
  2. Publicly available directory
  3. Public-key authority
  4. Public-key certificates

1. Public Announcement

- Users distribute public keys to recipients or broadcast to community at large
  - Eg. append PGP keys to email messages or post to news groups or email list
- Major weakness is forgery
  - Anyone can create a key claiming to be someone else and broadcast it
  - Until forgery is discovered can masquerade as claimed user

2. Publicly Available Directory

- Can obtain greater security by registering keys with a public directory
- Directory must be trusted with properties:
  - 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
- Maintenance and distribution of the public directory would have to be the responsibility of some trusted entity or organization.
- Still vulnerable to tampering or forgery

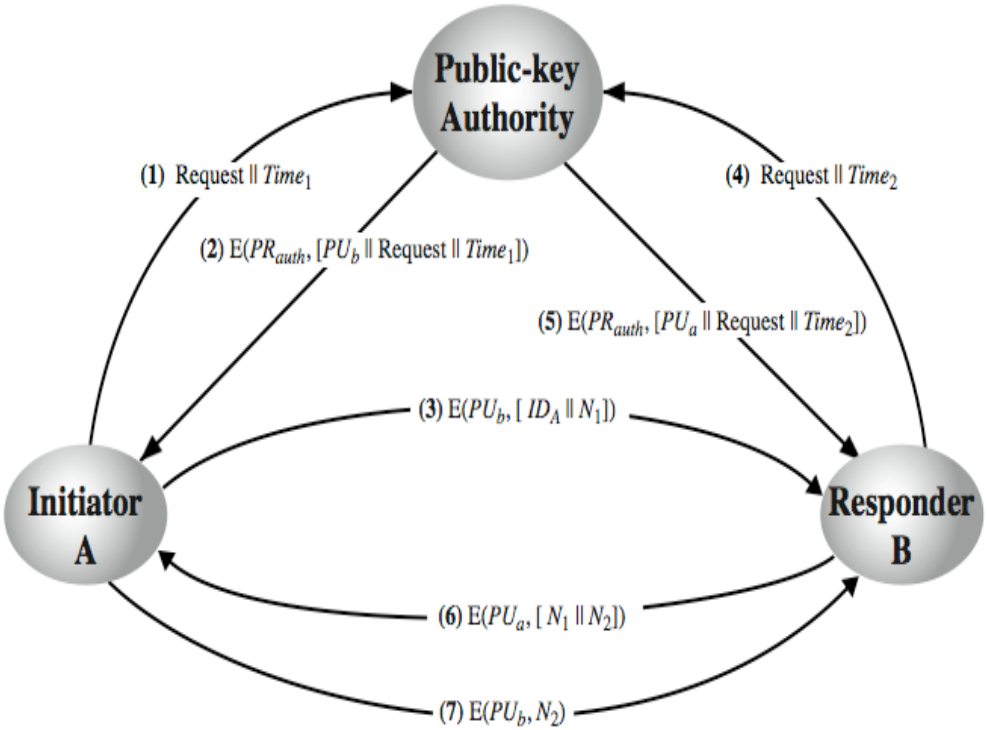
3. Public-Key Authority

- Stronger security for public-key distribution can be achieved by **providing tighter control over the distribution of public keys from the directory**.
- **It requires users to know the public key for the directory**, and that they interact with directory in real-time to obtain any desired public key securely.
- Has properties of directory
- Users interact with directory to obtain any desired public key securely
  - does require real-time access to directory when keys are needed
  - As before, the directory of names and public keys maintained by the authority is **vulnerable to tampering**

3. Public-Key Authority

- The **public-key authority** could be somewhat of a bottleneck in the system, for a user must appeal to the authority for a public key for every other user that it wishes to contact.

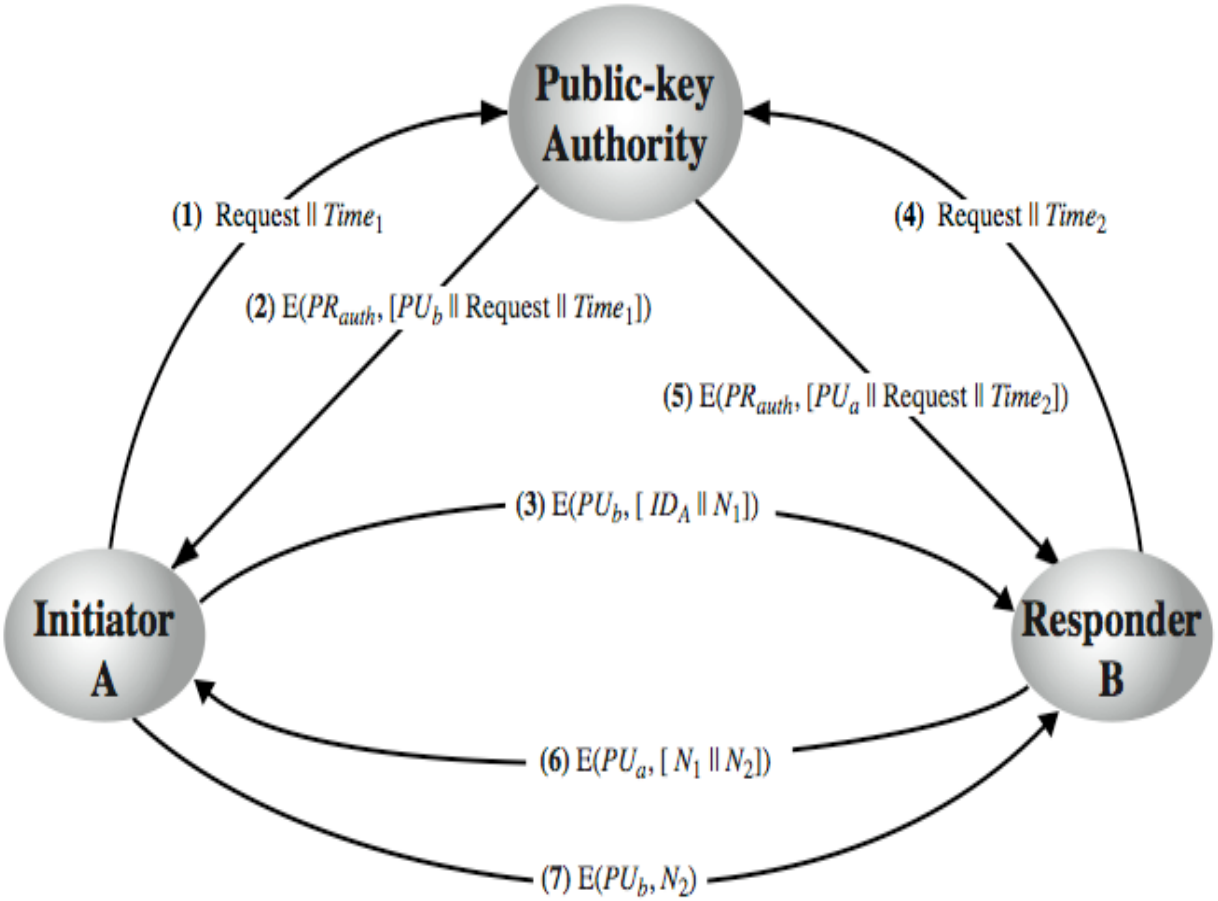
"Public-Key Authority" illustrates a typical protocol interaction. As before, the scenario assumes that a central authority maintains a dynamic directory of public keys of all participants. In addition, each participant reliably knows a public key for the authority, with only the authority knowing the corresponding private key.



3. Public-Key Authority

The initial four messages need be used only infrequently because both A and B can save the other's public key for future use, a technique known as **caching**.

Periodically, a user should request fresh copies of the public keys of its correspondents to ensure currency.



4. Public-Key Certificates

- First suggested by Kohnfelder, which can be used to exchange keys without contacting a public-key authority, in a way that is as reliable as if the keys were obtained directly from a public-key authority.
- A certificate binds an identity to public key, with all contents signed by a trusted Public-Key or Certificate Authority (CA).
- A user can present his or her public key to the authority in a secure manner, and obtain a certificate. The user can then publish the certificate.
- Anyone needing this user's public key can obtain the certificate and verify that it is valid by way of the attached trusted signature.
- A participant can also convey its key information to another by transmitting its certificate. Other participants can verify that the certificate was created by the authority, provided they know its public key.

4. Public-Key Certificates

- Certificates allow key exchange without real-time access to public-key authority
- A certificate binds **identity** to **public key**
  - usually with other info such as period of validity, rights of use etc
- With all contents **signed** by a trusted Public-Key or Certificate Authority (CA)
- Can be verified by anyone who knows the public-key authorities public-key

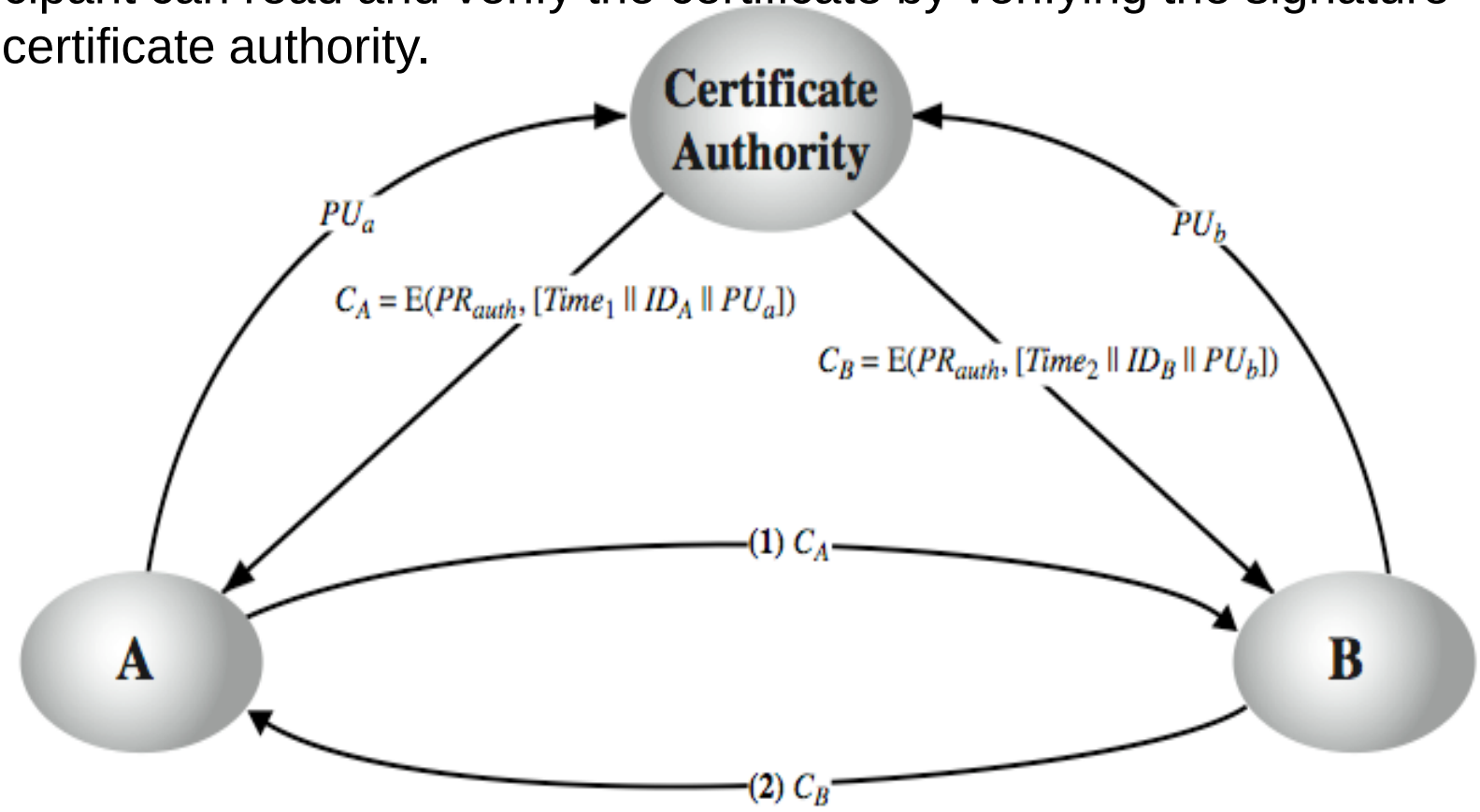
4. Public-Key Certificates

- One scheme has become universally accepted for formatting public-key certificates:
- The X.509 standard.
  - X.509 certificates are used in most network security applications, including IP security, secure sockets layer (SSL), secure electronic transactions (SET), and S/MIME. Will discuss it in much more detail later.

4. Public-Key Certificates

Each participant applies to the certificate authority, supplying a public key and requesting a certificate.
Application must be in person or by some form of secure authenticated communication.

Any participant can read and verify the certificate by verifying the signature from the certificate authority.



4. Public-Key Certificates

The **timestamp** counters the following scenario.

1. A's private key is learned by an adversary.
2. A generates a new private/public key pair and applies to the certificate authority for a new certificate. Meanwhile, the adversary replays the old certificate to B.
3. If B then encrypts messages using the compromised old public key, the adversary can read those messages.

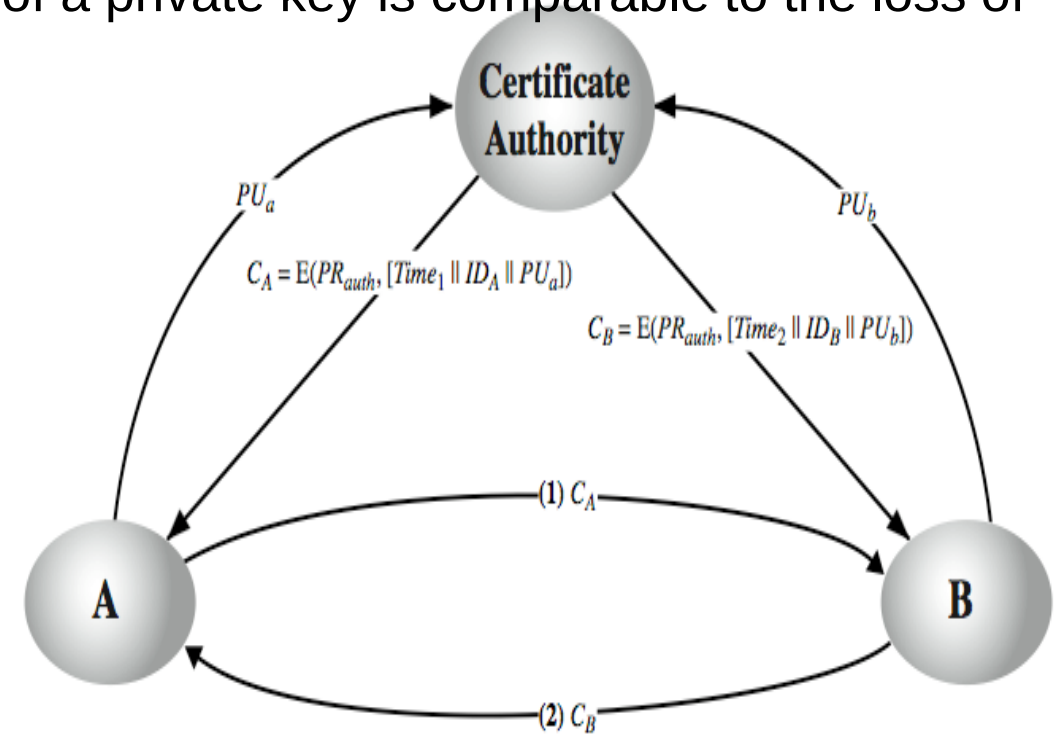
In this context, the compromise of a private key is comparable to the loss of a credit card.

The owner cancels the credit card number but is at risk until all possible communicants are aware that the old credit card is obsolete.

Thus, the timestamp serves as something like an expiration date.

If a certificate is sufficiently old, it is assumed to be expired.

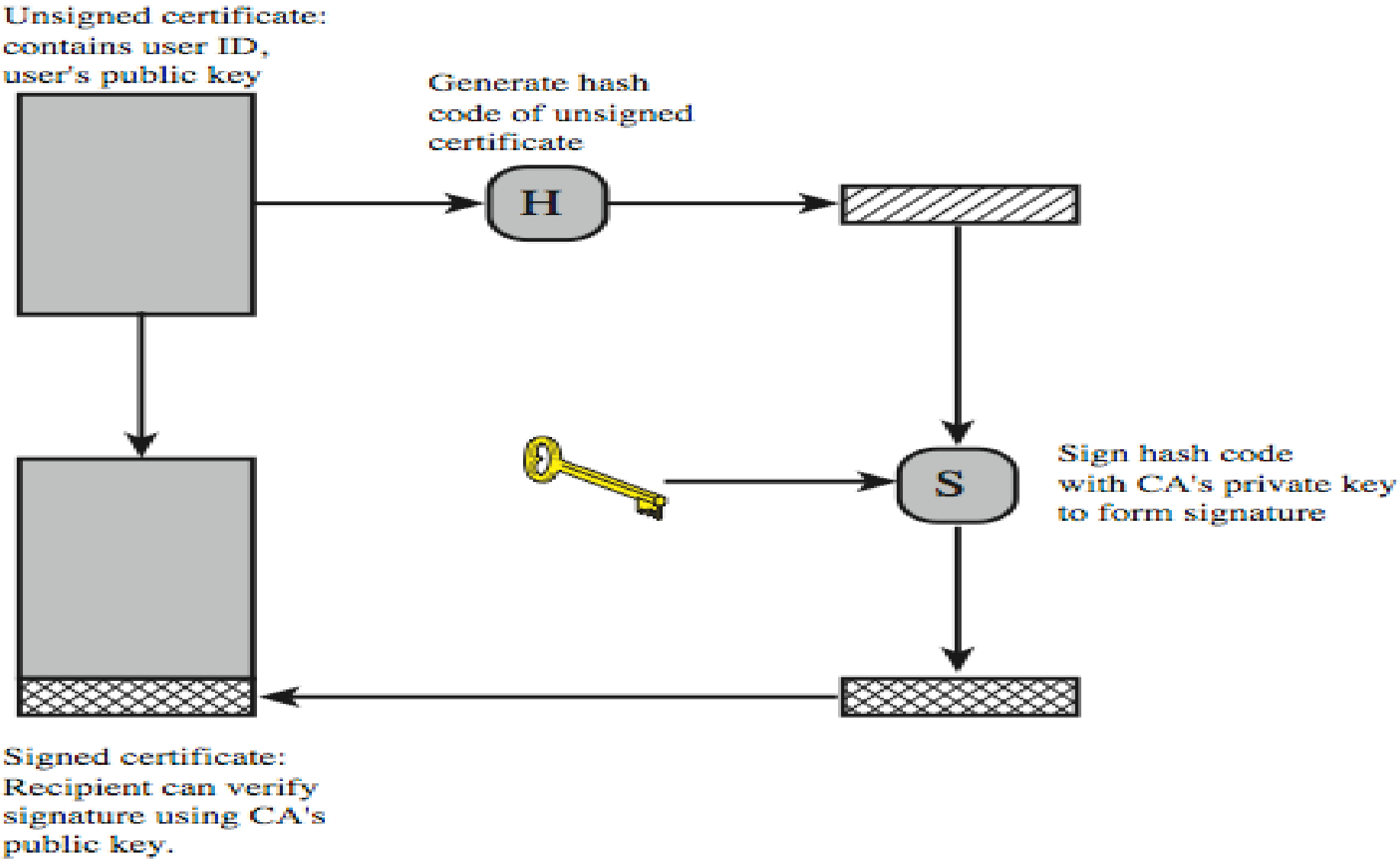
One scheme has become universally accepted for formatting public-key certificates: the X.509 standard.



X.509 Authentication Service

- Part of CCITT X.500 directory service standards
  - distributed servers maintaining user info database
- X509 defines framework for authentication services by the X500 directory to its users
  - directory may store public-key certificates
  - with public key of user signed by certification authority
- X.509 defines alternative authentication protocols based on the use of public-key certificates
- Uses public-key crypto & digital signatures
  - algorithms not standardized, but RSA recommended
- X.509 certificates are widely used
  - X.509 was initially issued in 1988.
  - The standard was subsequently revised to address some of the security concerns; a revised recommendation was issued in 1993. A third version was issued in 1995 and revised in 2000

X.509 Certificate Use



X.509 Certificates

- The heart of the X.509 scheme is the public-key certificate associated with each user.
- There are 3 versions, with successively more info in the certificate –
  - must be v2 if either unique identifier field exists,
  - must be v3 if any extensions are used.
- These user certificates are assumed to be created by some trusted certification authority (CA) and placed in the directory by the CA or by the user.
- The directory server itself is not responsible for the creation of public keys or for the certification function; it merely provides an easily accessible location for users to obtain certificates.

X.509 Certificates

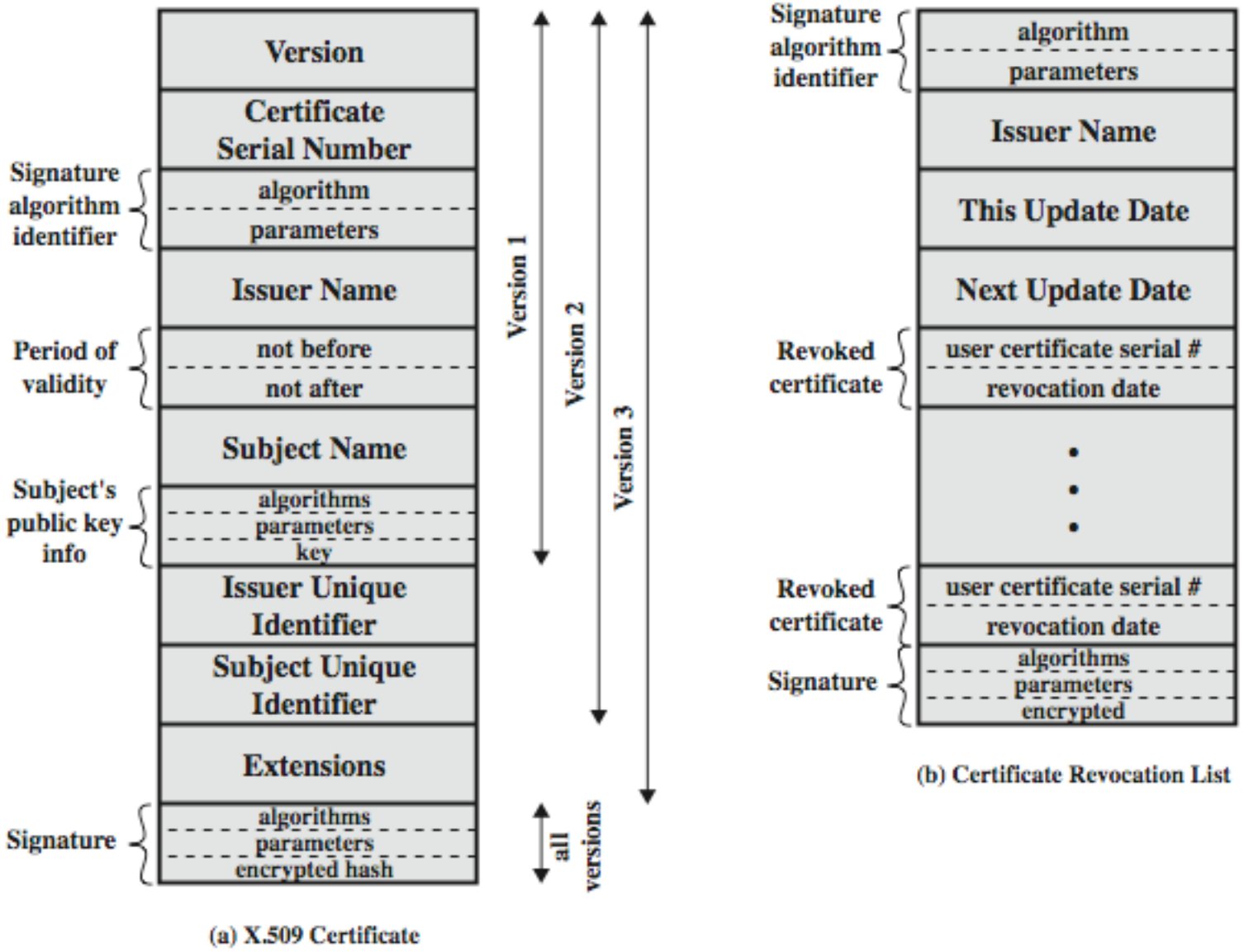
- Issued by a Certification Authority (CA), containing:
  - version V (1, 2, or 3)
  - serial number SN (unique within CA) identifying certificate
  - signature algorithm identifier AI
  - issuer X.500 name (CA)
  - period of validity TA (from - to dates)
  - subject X.500 name A (name of owner)
  - subject public-key info Ap (algorithm, parameters, key)
  - issuer unique identifier (v2+)
  - subject unique identifier (v2+)
  - extension fields (v3)
  - signature (of hash of all fields in certificate)
- Notation CA<<A>> denotes certificate for A signed by CA

X.509 Certificates

- The standard uses the notation for a certificate of:
  - CA<<A>> where the CA signs the certificate for user A with its private key.
  - In more detail

    CA<<A>> = CA {V, SN, AI, CA, UCA, A, UA, Ap, TA}.
  - If the corresponding public key is known to a user, then that user can verify that a certificate signed by the CA is valid.

X.509 Certificates



(a) X.509 Certificate

(b) Certificate Revocation List

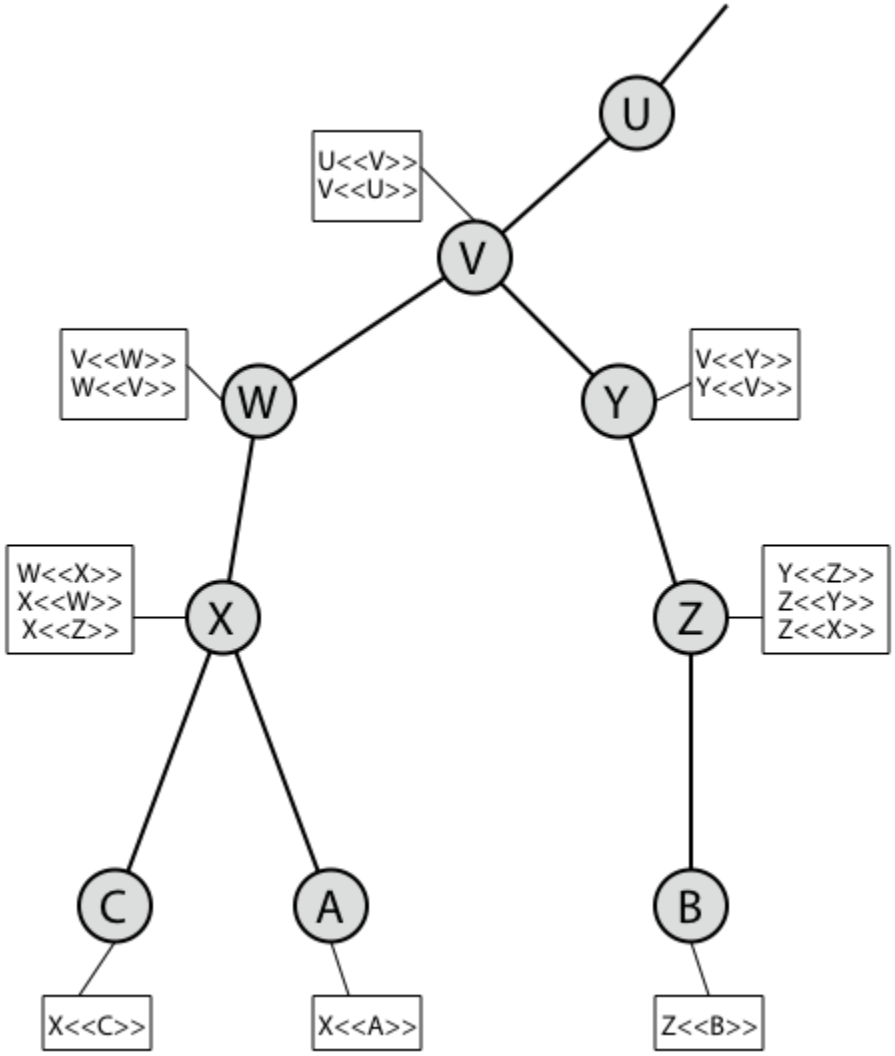
Obtaining a Certificate

- Any user with access to CA can get any certificate from it
- Only the CA can modify a certificate
- Because cannot be forged, certificates can be placed in a public directory

CA Hierarchy

- If both users share a common CA then they are assumed to know its public key
- Otherwise CA's must form a hierarchy
- Use certificates linking members of hierarchy to validate other CA's
  - each CA has certificates for clients (forward) and parent (backward)
- Each client trusts parents certificates
- Enable verification of any certificate from one CA by users of all other CAs in hierarchy

CA Hierarchy Use



The directory entry for each CA includes two types of certificates:

Forward certificates: Certificates of X generated by other CAs, and

Reverse certificates: Certificates generated by X that are the certificates of other CAs.

In this example, we can track chains of certificates as follows:

A acquires B certificate using chain:
X<<W>>W<<V>>V<<Y>>Y<<Z>>Z<<B>>

B acquires A certificate using chain:
Z<<Y>>Y<<V>>V<<W>>W<<X>>X<<A>>

Certificate Revocation

- Certificates have a period of validity
- May need to revoke before expiry, eg:
  1. user's private key is compromised
  2. user is no longer certified by this CA
  3. CA's certificate is compromised
- CA's maintain list of revoked certificates
  - the Certificate Revocation List (CRL)
- Users should check certificates with CA's CRL

X.509 Version 3

- Has been recognised that additional information is needed in a certificate
  - email/URL, policy details, usage constraints
- Rather than explicitly naming new fields defined a general extension method
- Extensions consist of:
  - extension identifier
  - criticality indicator; indicates whether an extension can be safely ignored or not (in which case if unknown the certificate is invalid)
  - extension value

Certificate Extensions

The certificate extensions fall into three main categories:

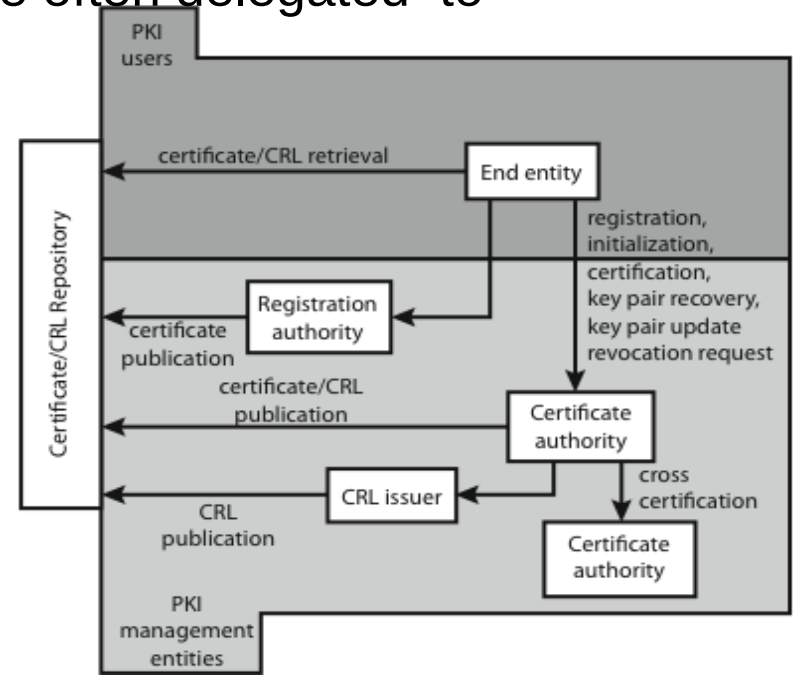
- Key and policy information
  - convey info about subject & issuer keys, plus indicators of certificate policy
  - A certificate policy is a named set of rules that **indicates the applicability of a certificate to a particular community and/or class of application with common security requirements**.
- Certificate subject and issuer attributes
  - support alternative names, in alternative formats for certificate subject and/or issuer; eg. postal address, email address, or picture image
- Certificate path constraints
  - allow constraints on use of certificates by other CA's

Public Key Infrastructure

RFC 2822 (Internet Security Glossary) defines public-key infrastructure (PKI) as the set of hardware, software, people, policies, and procedures needed to create, manage, store, distribute, and revoke digital certificates based on asymmetric cryptography.

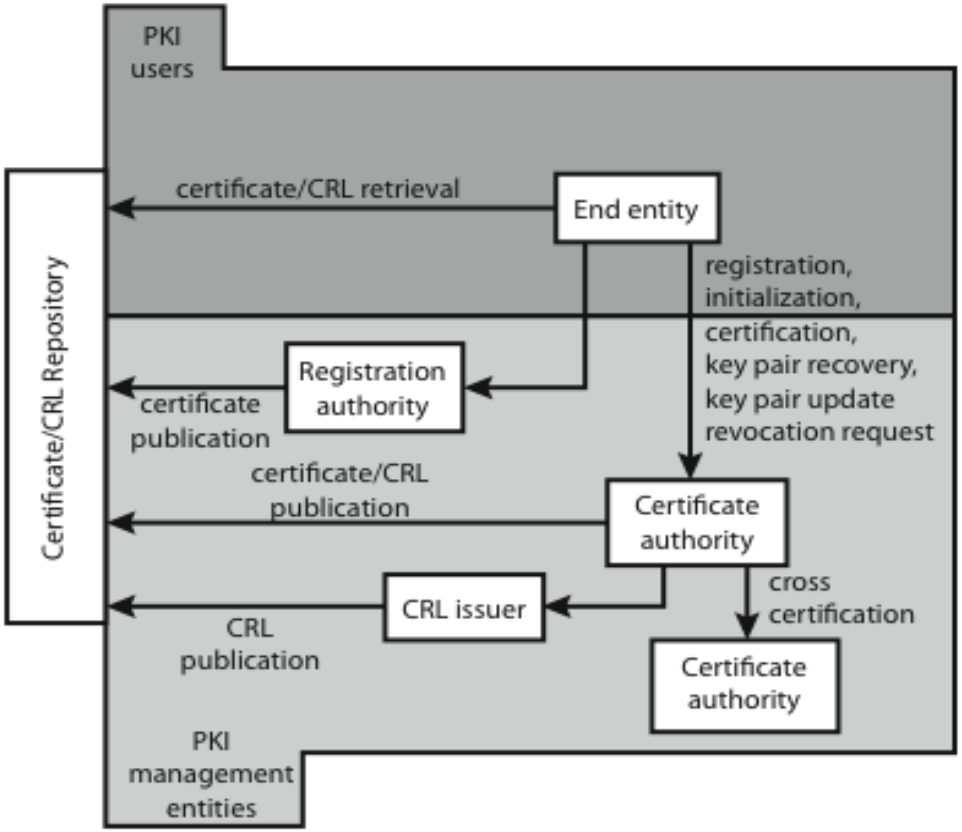
•**End entity**: A generic term used to denote end users, devices or any other entity that can be identified in the subject field of a public key certificate. End entities can consume and/or support PKI-related services.

• **Certification authority** (CA): The issuer of certificates and (usually) certificate revocation lists (CRLs). It may also support a variety of **Administrative Functions**, although these are often delegated to Registration Authorities.



Public Key Infrastructure

- **Registration authority (RA):** An optional component that can assume a number of administrative functions from the CA. The RA is often associated with the End Entity registration process, but can assist in a number of other areas as well.
- **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 so that they can be retrieved by End Entities.



Public Key Infrastructure

The IETF Public Key Infrastructure X.509 (PKIX) working group has setup a formal (and generic) model based on X.509 that is suitable for deploying a certificate-based architecture on the Internet. (RFC 5280)

PKIX Management

- **Functions**:
  - **Registration**; whereby a user first makes itself known to a CA, prior to issue of a certificate(s) for that user. It usually involves some off-line or online procedure for mutual authentication.
  - **Initialization**; to install key materials that have the appropriate relationship with keys stored elsewhere in the infrastructure.
  - **Certification**; process where a CA issues a certificate for a user's public key, and returns it to the user's client system and/or posts it in a repository.
  - **Key Pair Recovery**; a mechanism to recover the necessary decryption keys when normal access to the keying material is no longer possible.

PKIX Management

- **Functions**:
  - **Key pair update;** key pairs need to be updated and new certificates issued.
  - **Revocation request**; when authorized person advises need for certificate revocation, e.g. private key compromise, affiliation change, name change.
  - **Cross Certification**; when two CAs exchange information used in establishing a cross-certificate, issued by one CA to another CA that contains a CA signature key used for issuing certificates.
  - **Protocols:** The PKIX working group has defines two alternative management protocols between PKIX entities. RFC 2510 defines the certificate management protocols (CMP), which is a flexible protocol able to accommodate a variety of technical, operational, and business models. RFC 2797 defines certificate management messages over Cryptographic Message Syntax CMS (RFC 2630) called CMC.

Summary

- Have considered:
  - Symmetric key distribution using symmetric encryption
  - Symmetric key distribution using public-key encryption
  - Distribution of public keys
    - announcement, directory, authoritery, CA
  - X.509 authentication and certificates
  - Public key infrastructure (PKIX)