Key Management - Asymmetric Cryptography

19CSE311 Computer Security

Jevitha KP

Department of CSE

Public Key Distribution

- In asymmetric-key cryptography, people do not need to know a **symmetric shared key.**

- If Alice wants to send a message to Bob, she only needs to know **Bob's public key,** which is open to the public and available to everyone.

- If Bob needs to send a message to Alice, he only needs to know **Alice's public key**, which is also known to everyone.

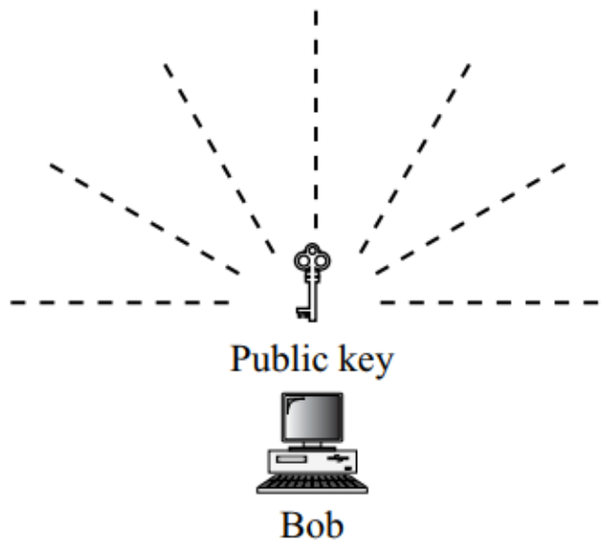
- In public-key cryptography, **private key is kept secret and public key is distributed.**

Public Key Distribution

- Public keys, like secret keys, need to be distributed to be useful.
- Techniques :
    - Public Announcement
    - Trusted Center
    - Controlled Trusted Center
    - **Certification Authority**
    - **X.509**

Public Announcement

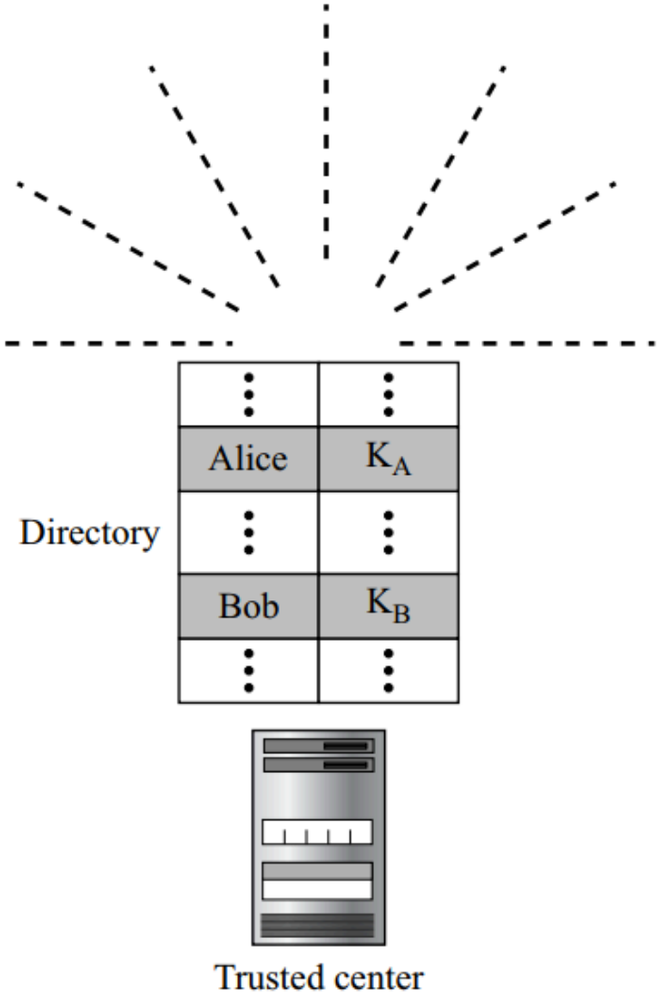
- The naive approach is to announce public keys publicly.
- Bob can put his public key on his website or announce it in a local or national newspaper.
- When Alice needs to send a confidential message to Bob, she can obtain Bob's public key from his site or from the newspaper, or even send a message to ask for it.



Public Announcement

- This approach, however, is not secure; it is subject to forgery.
- For example, Eve could make such a public announcement. Before Bob can react, damage could be done.
- Eve could also **sign a document** with a corresponding **forged private key** and make everyone believe it was signed by Bob.
- The approach is also vulnerable if Alice directly requests Bob's public key.
- Eve can **intercept Bob's response** and **substitute her own forged public key** for Bob's public key

Trusted Center



Trusted Center

- A more secure approach is to have a trusted center retain a directory of public keys.

- The directory, like the one used in a telephone system, is dynamically updated.

- Each user can **select a private and public key**, keep the private key, and deliver the public key for insertion into the directory.

- The center requires that each user register in the center and **prove his or her identity.**

- The **directory** can be **publicly advertised by the trusted center.**

- The center can also **respond to any inquiry about a public key.**

Controlled Trusted Center

- A higher level of security can be achieved if there are **added controls** on the **distribution of the public key.**

- The public-key announcements can include a **timestamp** and be signed by an **authority** to prevent **interception** and **modification** of the response.

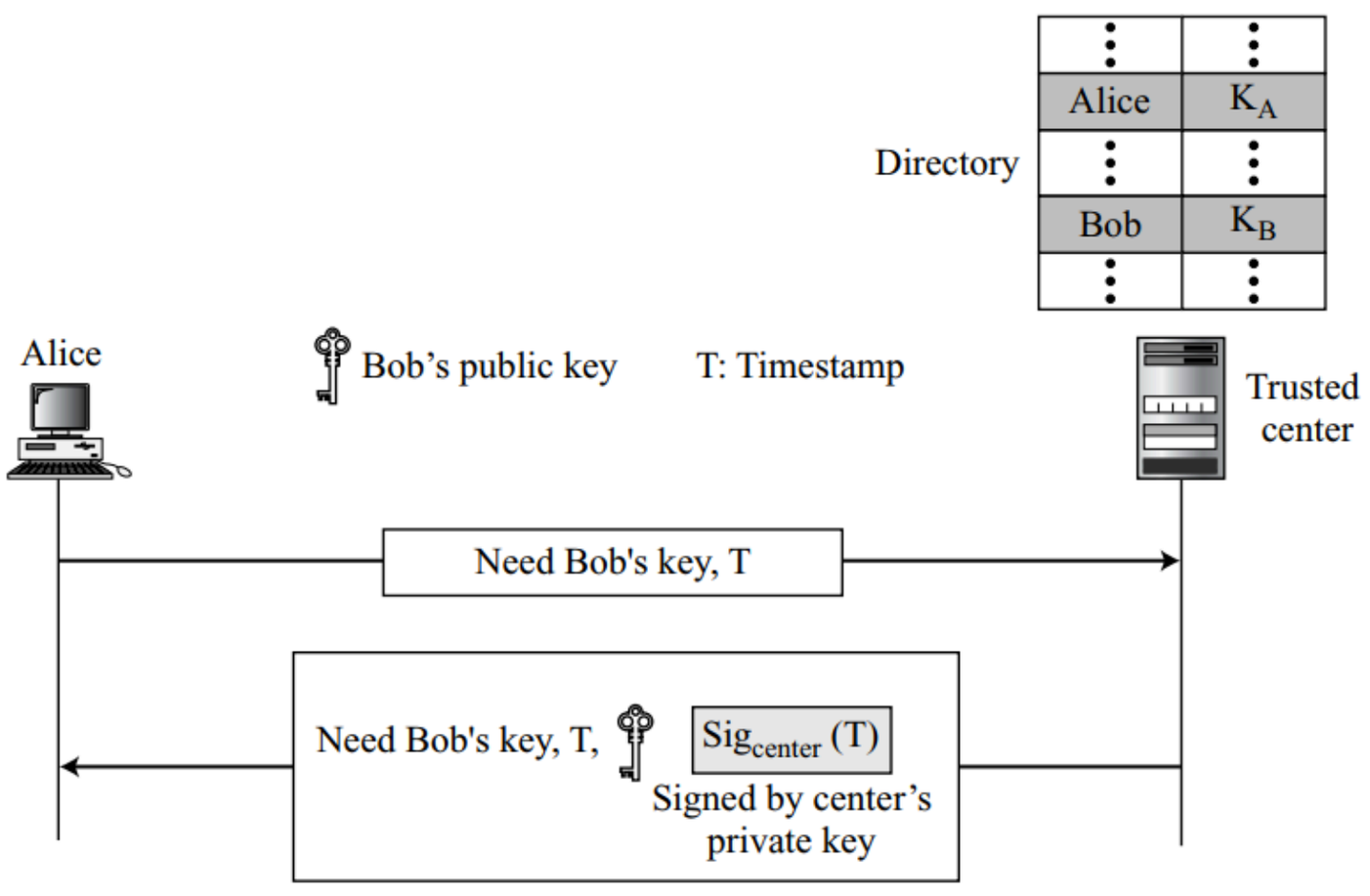
- If Alice needs to know **Bob's public key**, she can send a **request** to the **center** including **Bob's name and a timestamp.**

- The center responds with **Bob's public key, the original request, and the timestamp signed with the private key of the center.**

- Alice uses the **public key of the center**, known by all, to **verify the timestamp.**

- If the timestamp is verified, she extracts Bob's public key.

Controlled Trusted Center



Certification Authority

- The previous approach can create a heavy load on the center if the number of requests is large.

- The alternative is to create **public-key certificates.**

- Bob wants two things:

  - people should know his public key, and

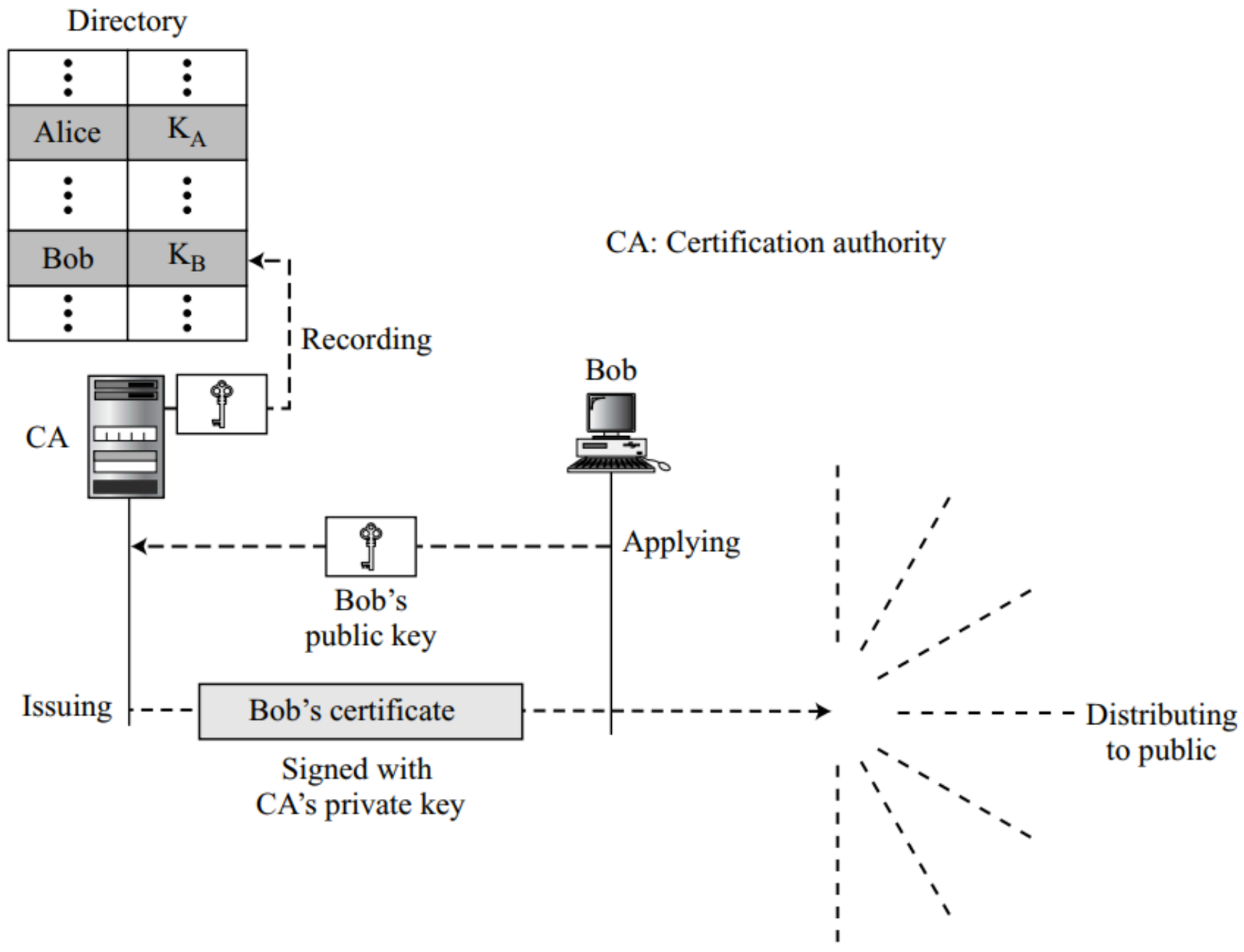
  - no one should accept a forged public key as his.

- Bob can go to a certification authority (CA), a federal or state organization that **binds a public key to an entity and issues a certificate.**

Certification Authority

- The CA has a **well-known public key itself** that cannot be forged.
- The CA checks Bob’s identification (using a picture ID along with other proof).
- It then asks for **Bob’s public key** and **writes it on the certificate.**
- To prevent the certificate itself from being forged, **the CA signs the certificate with its private key.**
- Now **Bob can upload the signed certificate.**
- Anyone who wants Bob’s public key **downloads the signed certificate** and uses the **center’s public key to extract Bob’s public key**

Certification Authority



X.509

- Although the use of a CA has solved the **problem of public-key fraud**, it has created a **side-effect.**

- Each certificate may have a **different format.**

- If Alice wants to use a program to **automatically download different certificates** and **digests** belonging to different people, the program may not be able to do this.

- One certificate may have the **public key in one format and another in a different format.**

- The public key may be on the first line in one certificate, and on the third line in another.

- Anything that needs to be used universally must have a **universal format**

X.509

- X.509 is an International Telecommunication Union (ITU) **standard** defining the format of public key certificates.

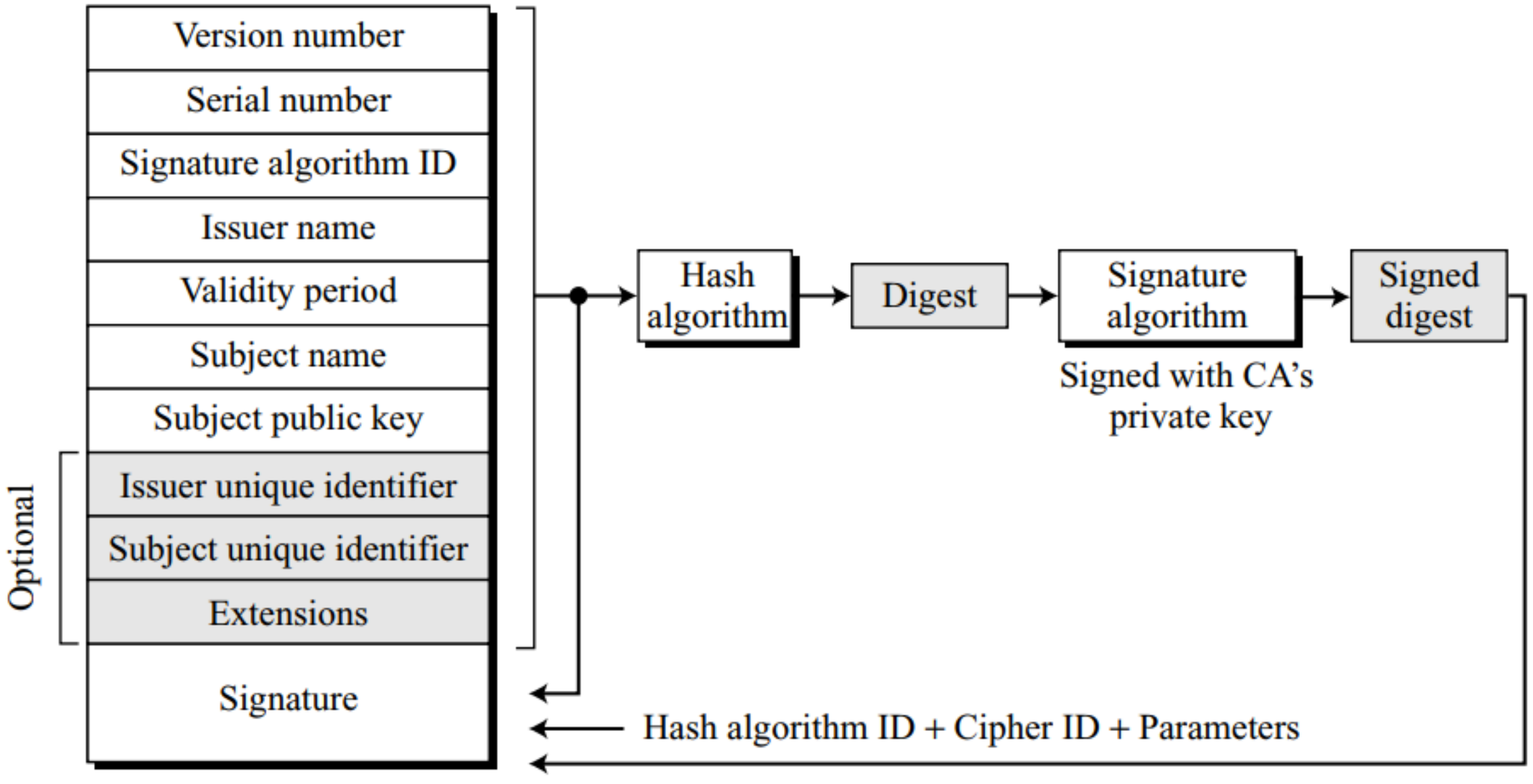
- X.509 certificates are used in many Internet protocols, including TLS/SSL, which is the basis for HTTPS, the secure protocol for browsing the web.

- They are also used in offline applications, like electronic signatures.

- X.509 is a way to describe the certificate in a structured way.

- It uses a well-known protocol called **ASN.1 (Abstract Syntax Notation 1)** that defines fields familiar to C programmers.

X.509


Version number
Serial number
Signature algorithm ID
Issuer name
Validity period
Subject name
Subject public key
Issuer unique identifier
Subject unique identifier
Extensions
Signature
Optional
Hash algorithm
Digest
Signature algorithm
Signed with CA's private key
Signed digest
Hash algorithm ID + Cipher ID + Parameters

X.509

- **Version number.**
    - This field defines the **version of X.509** of the certificate**.**
    - The version number started at 0; second version is 1, third version is 2, which is still used today to support the expansion and new applications of internet use**.**
        - Now version 9 is the current version of the standard, having been defined in October 2019**.**
- **Serial number.** This field defines a number assigned to each certificate**. The value is unique for each certificate issuer.**
- **Signature algorithm ID.** This field identifies the **algorithm** used to sign the certificate**.** Any **parameter** that is needed for the signature is also defined in this field**.**
- **Issuer name.** This field identifies the **certification authority** that issued the certificate**.** The name is normally a **hierarchy of strings** that defines a country, a state, organization, department, and so on**.**

X.509

- **Validity Period.** This field defines the **earliest time** (not before) and the **latest time** (not after) the certificate is valid.

- **Subject name.** This field defines the **entity** to which the public key belongs. It is also a **hierarchy of strings**. Part of the field defines what is called the **common name,** which is the actual name of the beholder of the key.

- **Subject public key.** This field defines the **owner's public key**, the heart of the certificate. The field also defines the corresponding **public-key algorithm (RSA, for example) and its parameters.**

X.509

- **Issuer unique identifier.** This optional field allows **two issuers** to have the **same issuer field value**, if the **issuer unique identifiers are different.**

- **Subject unique identifier.** This optional field allows two different subjects to have the same **subject field value**, if the **subject unique identifiers are different.**

- **Extensions.** This optional field allows issuers to add more private information to the certificate.

X.509

- **Signature.** This field is made of **three** sections.
  - The first section contains **all other fields** in the certificate.
  - The second section contains the **digest of the first section encrypted with the CA's public key.**
  - The third section contains the **algorithm identifier used to create the second section.**

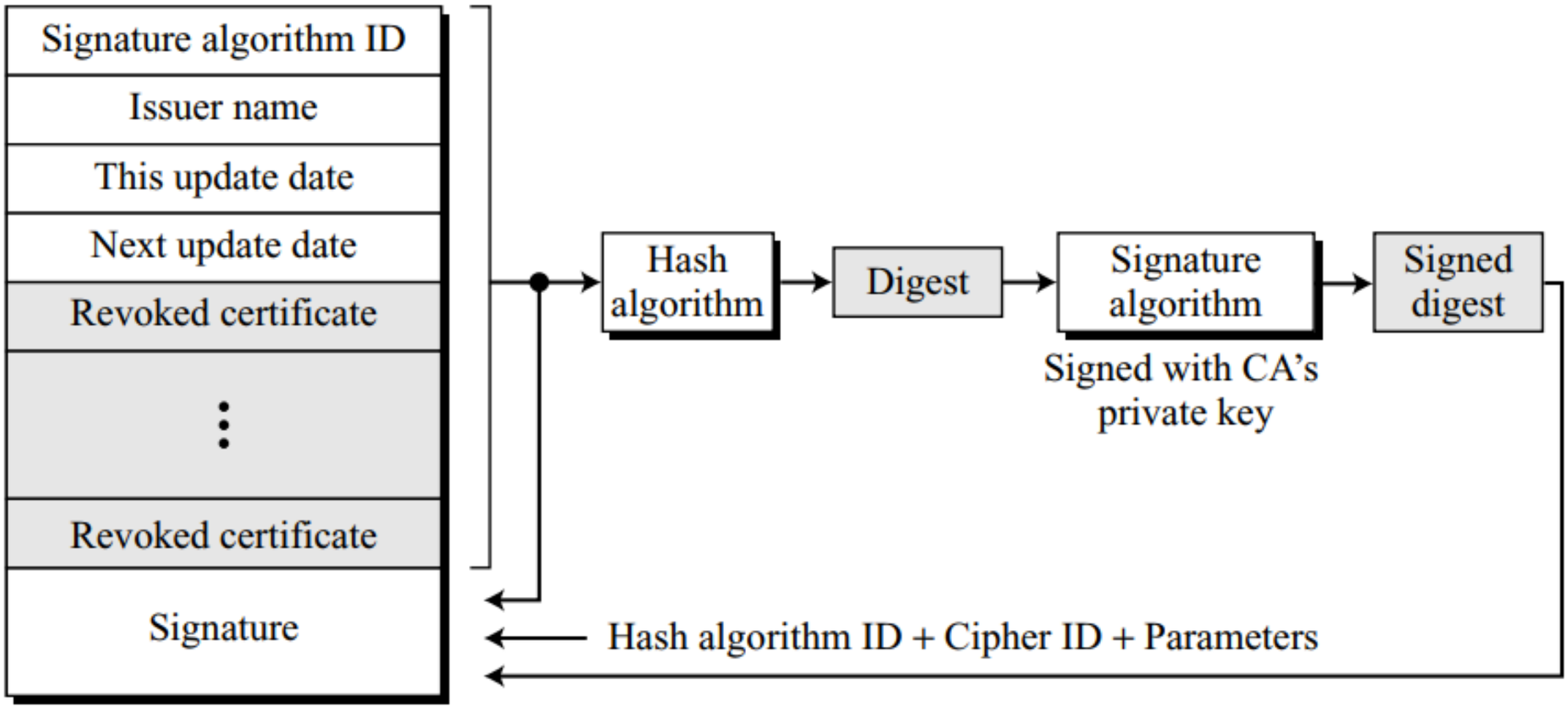
Certificate Renewal

- Each certificate has a period of validity.
- If there is no problem with the certificate, **the CA issues a new certificate before the old one expires.**
- The process is like the renewal of credit cards by a credit card company; the credit card holder normally receives a renewed credit card before the one expires.

Certificate Revocation

- In some cases a certificate must be revoked before its expiration. Here are some examples:
    - The **user's (subject's) private key** (corresponding to the public key listed in the certificate) might have been **comprised**.
    - The **CA is no longer willing to certify the user.** For example, the user's certificate relates to an organization that she no longer works for.
    - The **CA's private key,** which can verify certificates, may have been **compromised**. In this case, the CA needs to revoke all unexpired certificates.
- The revocation is done by periodically issuing a **certificate revocation list (CRL).**
- The list contains all revoked certificates that are not expired on the date the CRL is issued.
- When a user wants to use a certificate, she first needs to check the directory of the corresponding CA for the last certificate revocation list.

Certificate Revocation



Certificate Revocation

- A certificate revocation list has the following fields:

- **Signature algorithm ID.** This field is the same as the one in the certificate.

- **Issuer name.** This field is the same as the one in the certificate.

- **This update date.** This field defines when the list is released.

- **Next update date.** This field defines the next date when the new list will be released

Certificate Revocation

- **Revoked certificate.** This is a repeated list of all unexpired certificates that have been revoked. Each list contains two sections: **user certificate serial number and revocation date.**

- **Signature.** This field is the same as the one in the certificate list.

Delta Revocation

- To make revocation more efficient, the delta certificate revocation list (delta CRL) has been introduced.

- A **delta CRL is created and posted on the directory if there are changes after this update date and next update date.**

- For example, if CRLs are issued every month, but there are revocations in between, the CA can create a delta CRL when there is a change during the month.

- However, a delta CRL contains only the changes made after the last CRL

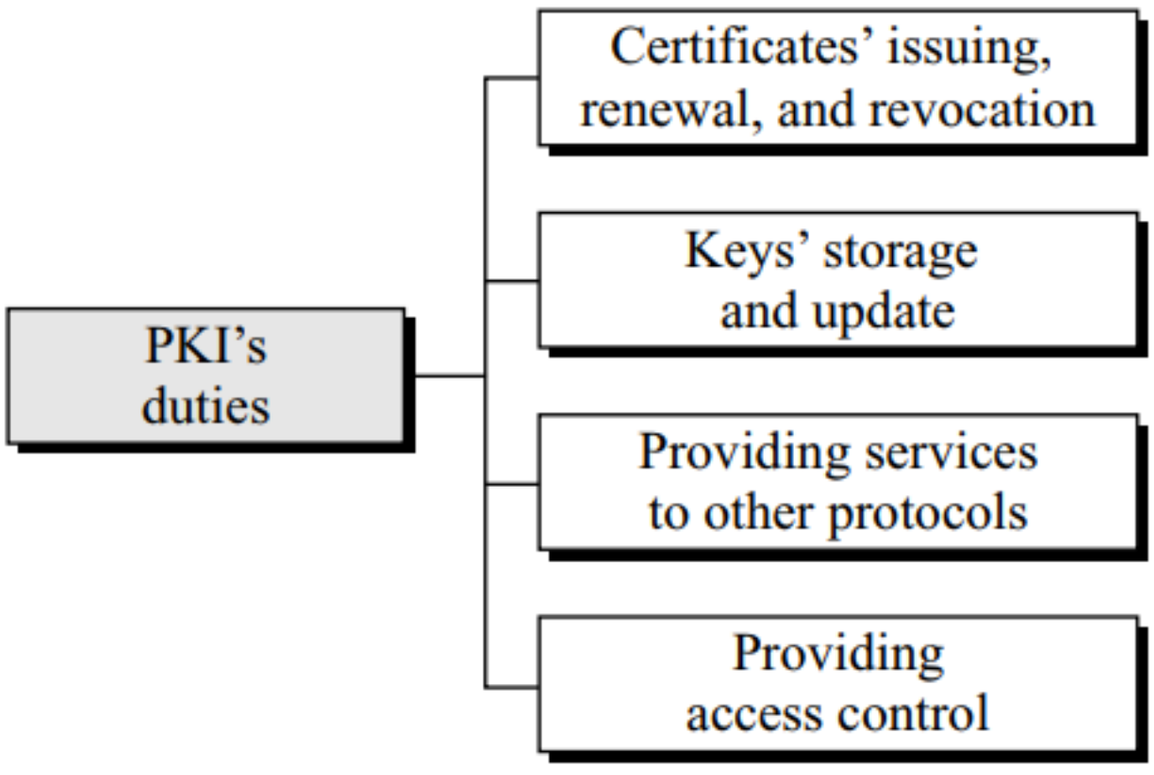
Public-Key Infrastructure(PKI)

Public-Key Infrastructures (PKI)

- Public-Key Infrastructure (PKI) is a model for creating, distributing, and revoking certificates based on the X.509.

- The Internet Engineering Task Force has created the Public-Key Infrastructure **X.509 (PKIX)**

Public-Key Infrastructures (PKI) - Duties



Public-Key Infrastructures (PKI) - Duties

- **Certificates' issuing, renewal, and revocation.**
  - These are duties defined in the X.509.
  - Because the PKIX is based on X.509, it needs to handle all duties related to certificates.
- **Keys' storage and update.**
  - A PKI should be a **storage place for private keys** of those members that need to hold their private keys somewhere safe.
  - In addition, a PKI is responsible for **updating these keys on members' demands.**

Public-Key Infrastructures (PKI) - Duties

- **Providing services to other protocols.** Some Internet security protocols, such as IPSec and TLS, are relying on the services by a PKI.

- **Providing access control.**

  - A PKI can provide different levels of access to the information stored in its database.

  - For example, an organization PKI may provide access to the whole database for the top management, but limited access for employees

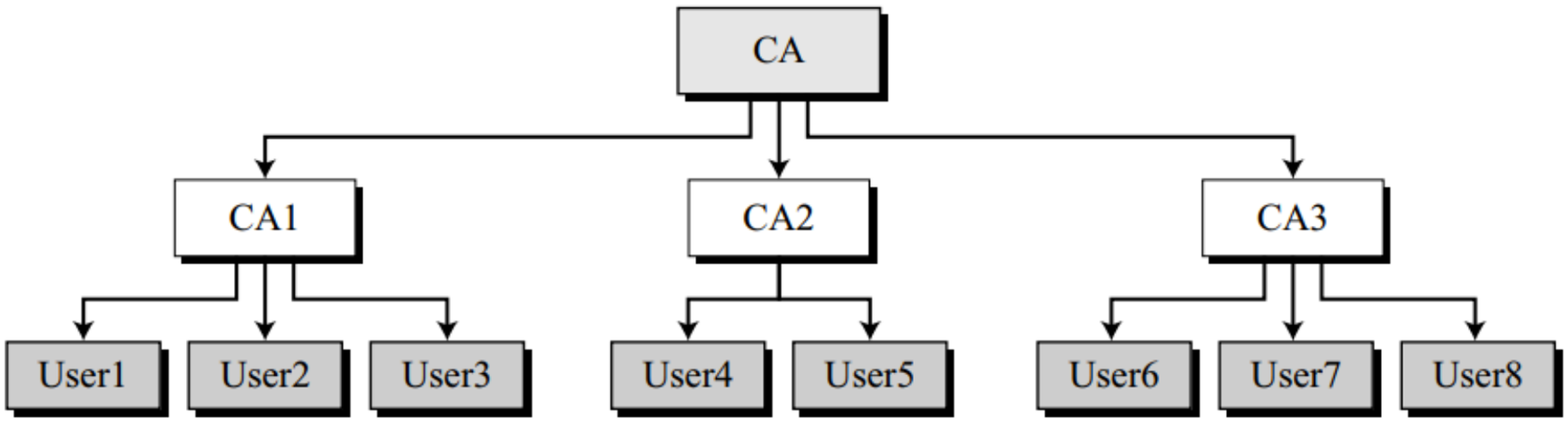
Trust Model

- It is not possible to have just one CA issuing all certificates for all users in the world.
- There should be **many CAs**, each responsible for creating, storing, issuing, and revoking a limited number of certificates.
- The **trust model** defines rules that specify how a user can verify a certificate received from a CA.
- Three models are discussed
    - Hierarchical Model
    - Mesh Model
    - Web of Trust

Hierarchical Model

- In this model, there is a tree-type structure with a root CA.
- The root CA has a **self-signed, self-issued certificate**; it needs to be trusted by other CAs and users for the system to work.
- The number of levels can be more than three in a real situation.

Hierarchical Model



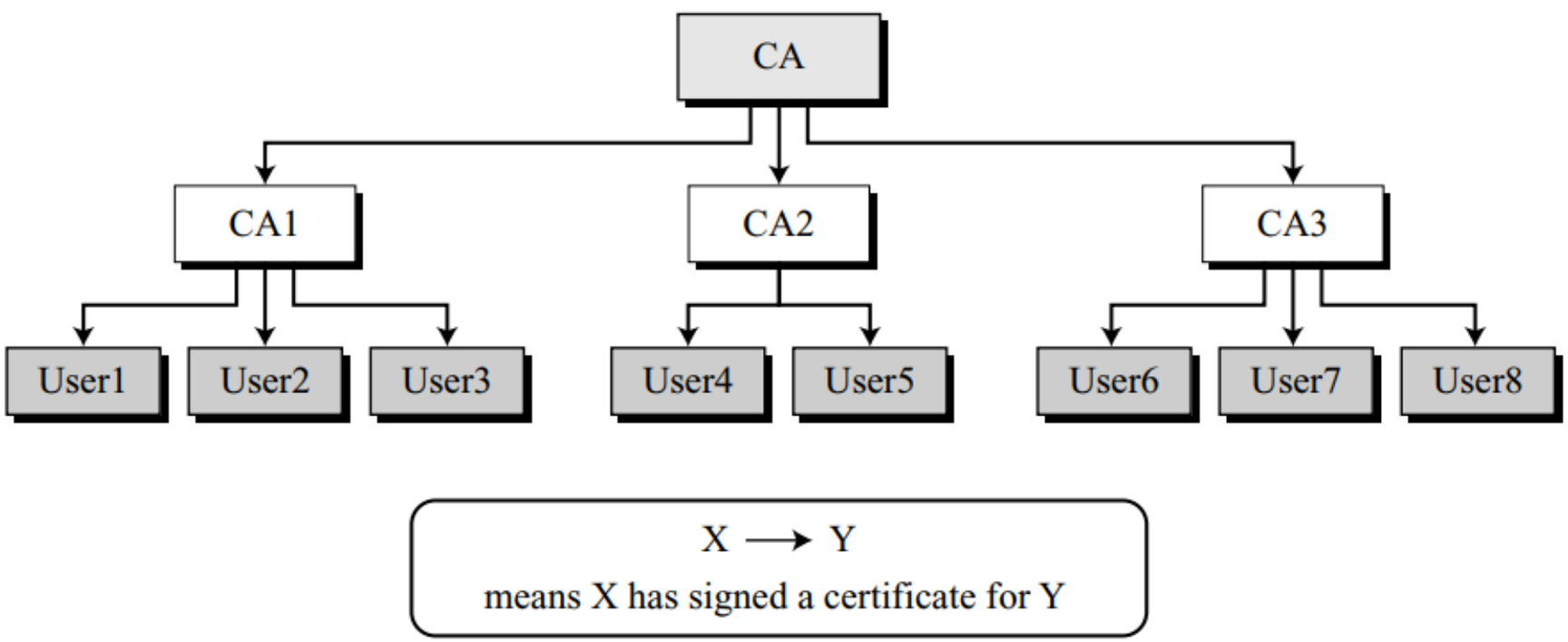
X ⟶ Y
means X has signed a certificate for Y

- The figure shows that the **CA (the root)** has **signed certificates** for **CA1, CA2, and CA3**; CA1 has signed certificates for User1, User2, and User3; and so on.

- PKI uses the following notation to mean the certificate issued by authority X for entity Y - **X<<Y>>**

Example

- Show how User1, knowing only the public key of the CA (the root), can obtain a verified copy of User3's public key



Example 1

- User3 sends a chain of certificates, CA<<CA1>> and CA1<<User3>>, to User1.

- User1 validates CA<<CA1>> using the public key of CA.

- User1 extracts the public key of CA1 from CA<<CA1>>.

- User1 validates CA1<<User3>> using the public key of CA1.

- User1 extracts the public key of User 3 from CA1<<User3>>.

Example 2

- Some Web browsers include a set of certificates from independent roots without a single, high-level, authority to certify each root.

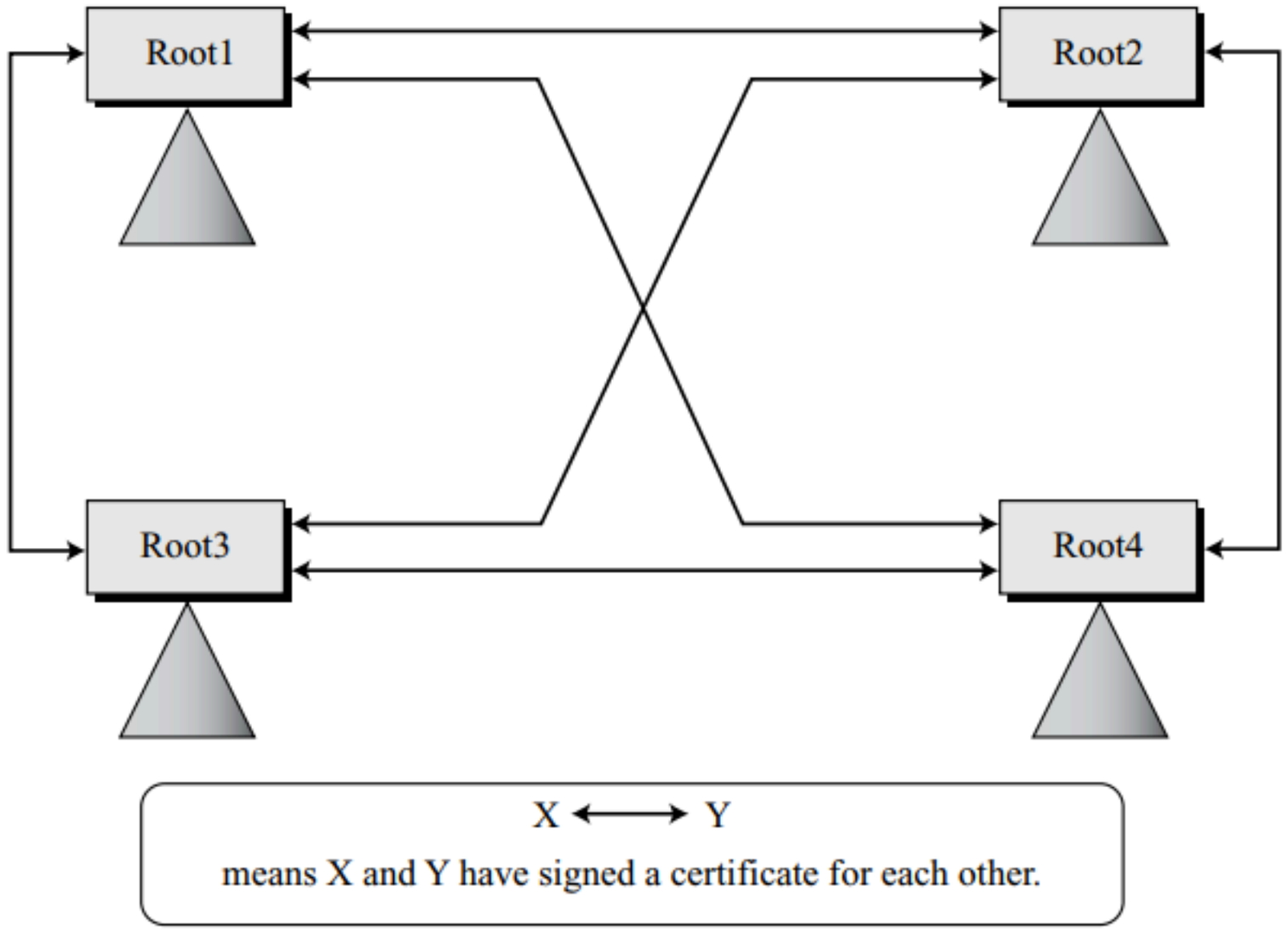
- One can find the list of these roots in the Browsers Certificate/Trusted roots.

- The user then can choose any of this root and view the certificate.

Mesh Model

- The hierarchical model may work for an organization or a small community.
- A larger community may need several hierarchical structures connected together.
- One method is to use a mesh model to connect the roots together.
- In this model, each root is connected to every other root.

Mesh Model


Root1
Root2
Root3
Root4
X ⟷ Y
means X and Y have signed a certificate for each other.

Mesh Model

- Mesh structure connects only roots together; each root has its own hierarchical structure, shown by a triangle.

- The certifications between the roots are **cross-certificates;** each root certifies all other roots, which means there are N*(N – 1) certificates.

- There are 4 nodes, so we need 4 × 3 = 12 certificates.

- Note that each double-arrow line represents two certificates.

Example

- Alice is under the authority Root1; Bob is under the authority Root4. Show how Alice can obtain Bob’s verified public key.

- Bob sends a chain of certificates from Root4 to Bob.

- Alice looks at the directory of Root1 to find Root1<<Root1>> and Root1<< Root4>> certificates. Using the process shown in Figure, Alice can verify Bob’s public key

Web of Trust

- This model is used in Pretty Good Privacy, a security service for electronic mail .

- Its decentralized trust model is an alternative to the centralized trust model of a public key infrastructure (PKI), which relies exclusively on a certificate authority (or a hierarchy of such).

- As with computer networks, there are **many independent webs of trust**, and any user (through their public key certificate) can be a part of, and a link between, multiple webs.

Web of Trust

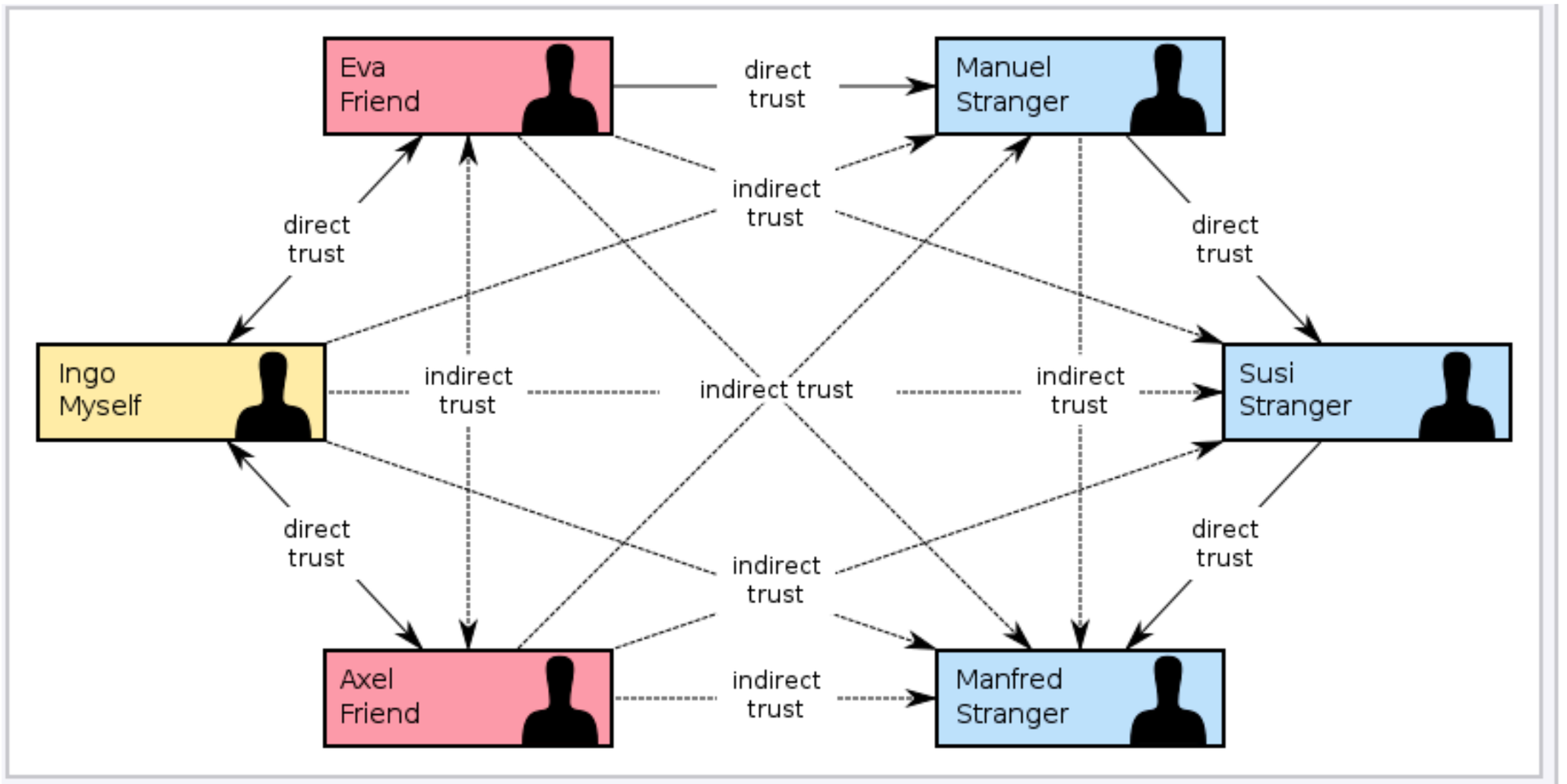
- In the web of trust, **each user has a ring with a group of people's public keys.**

- Users encrypt their information with the recipient's public key, and only the **recipient's private key will decrypt it.**

- Each user then **digitally signs the information** with their private key, so when the recipient verifies it against the user's own public key, they can confirm that it is the user in question.

- Doing this will ensure that the information came from the specific user and has not been tampered with, and only the intended recipient can read the information (because only they know their private key).

Web of Trust



Schematic diagram of a Web of Trust