

Key Management

Dr. Kunwar Pal
CSE
NITJ

Outline

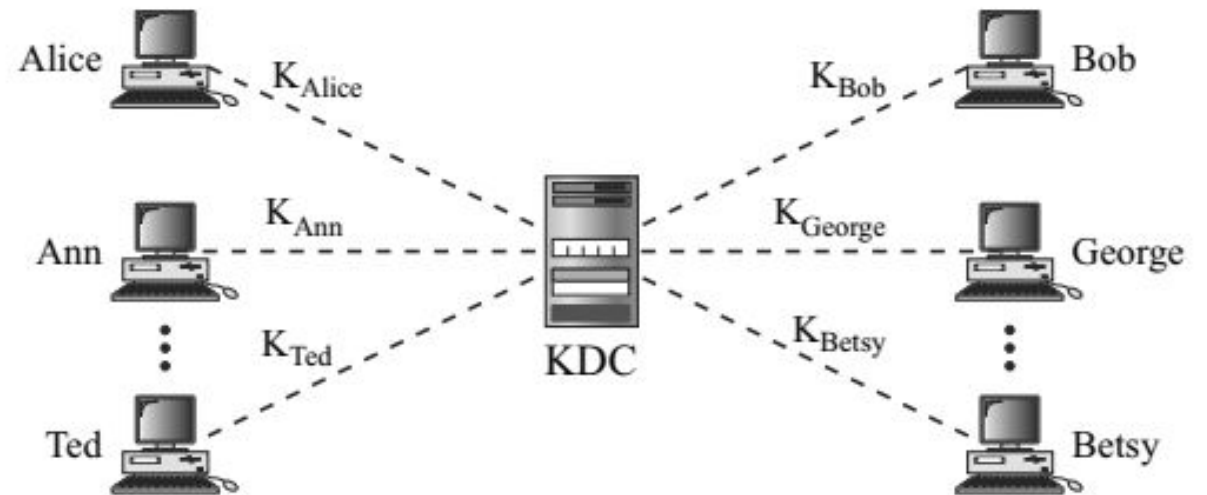
- We first discuss the distribution of symmetric keys using a trusted third party.
- Second, we show how two parties can establish a symmetric key between themselves without using a trusted third party.
- Third, we introduce Kerberos as both a KDC and an authentication protocol.
- Kerberos

SYMMETRIC-KEY DISTRIBUTION

- Symmetric-key cryptography is more efficient than asymmetric-key cryptography for enciphering large messages.
- Symmetric-key cryptography, however, needs a shared secret key between two parties.
- If Alice needs to exchange confidential messages with N people, she needs N different keys.
- What if N people need to communicate with each other?
- A total of $N(N - 1)$ keys is needed if we require that Alice and Bob use two keys for bidirectional communication; only $N(N - 1)/2$ keys are needed if we allow a key to be used for both directions.

Key-Distribution n Centre: KDC

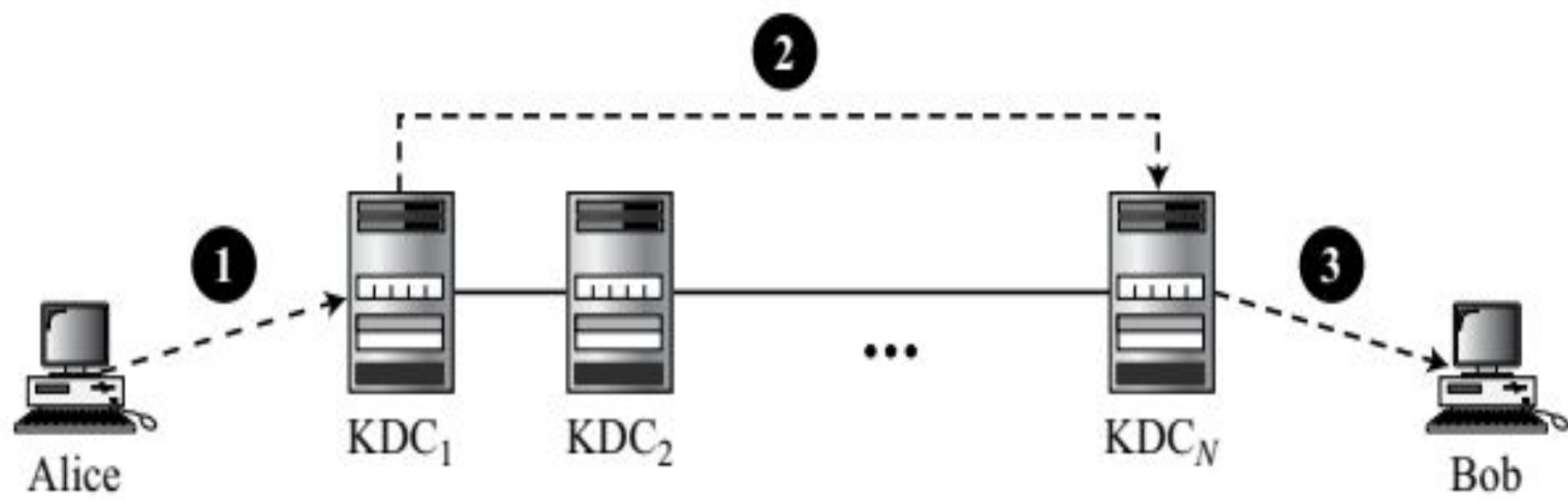
- A practical solution is the use of a trusted third party, referred to as a key-distribution center (KDC).
- To reduce the number of keys, each person establishes a shared secret key with the KDC



- A secret key is established between the KDC and each member. Alice has a secret key with the KDC, which we refer to as K_{Alice} ; Bob has a secret key with the KDC, which we refer to as K_{Bob} ; and so on.
- Now the question is how Alice can send a confidential message to Bob. The process is as follows:
 1. Alice sends a request to the KDC stating that she needs a session (temporary) secret key between herself and Bob.
 2. The KDC informs Bob about Alice's request.
 3. If Bob agrees, a session key is created between the two.

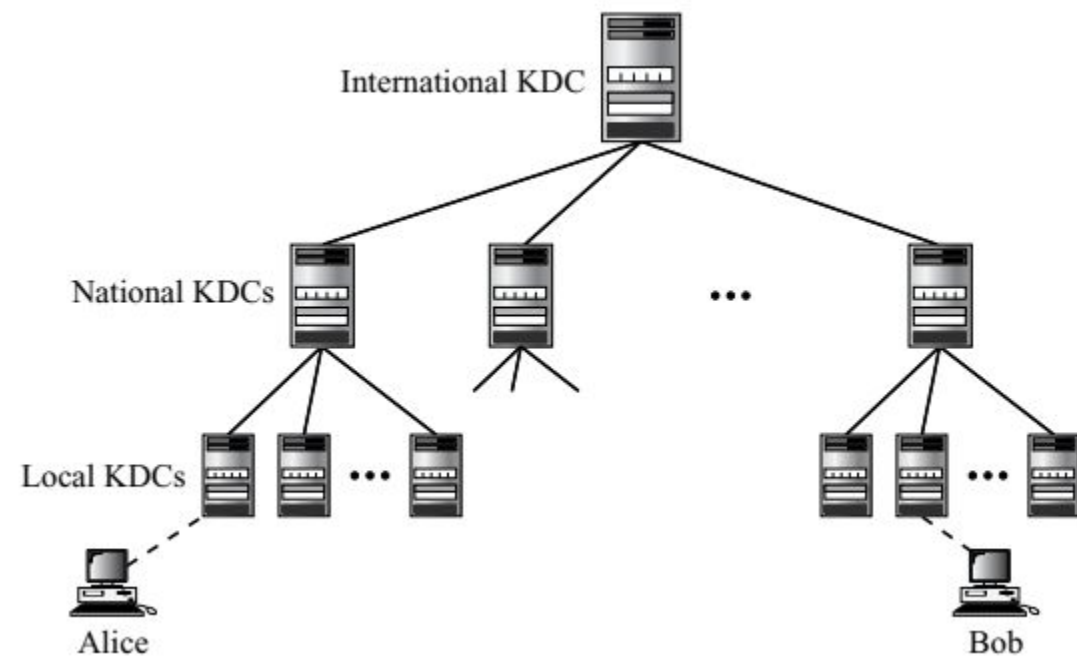
Flat Multiple KDCs

- When the number of people using a KDC increases, the system becomes unmanageable and a bottleneck can result.
- To solve the problem, we need to have multiple KDCs.
- We can divide the world into domains.
- Each domain can have one or more KDCs (for redundancy in case of failure).
- Now if Alice wants to send a confidential message to Bob, who belongs to another domain, Alice contacts her KDC, which in turn contacts the KDC in Bob's domain.
- The two KDCs can create a secret key between Alice and Bob.



Hierarchical Multiple KDCs

- The concept of flat multiple KDCs can be extended to a hierarchical system of KDCs, with one or more KDCs at the top of the hierarchy.
- For example, there can be local KDCs, national KDCs, and international KDCs.
- When Alice needs to communicate with Bob, who lives in another country, she sends her request to a local KDC; the local KDC relays the request to the national KDC; the national KDC relays the request to an international KDC.
- The request is then relayed all the way down to the local KDC where Bob lives.





Session Keys

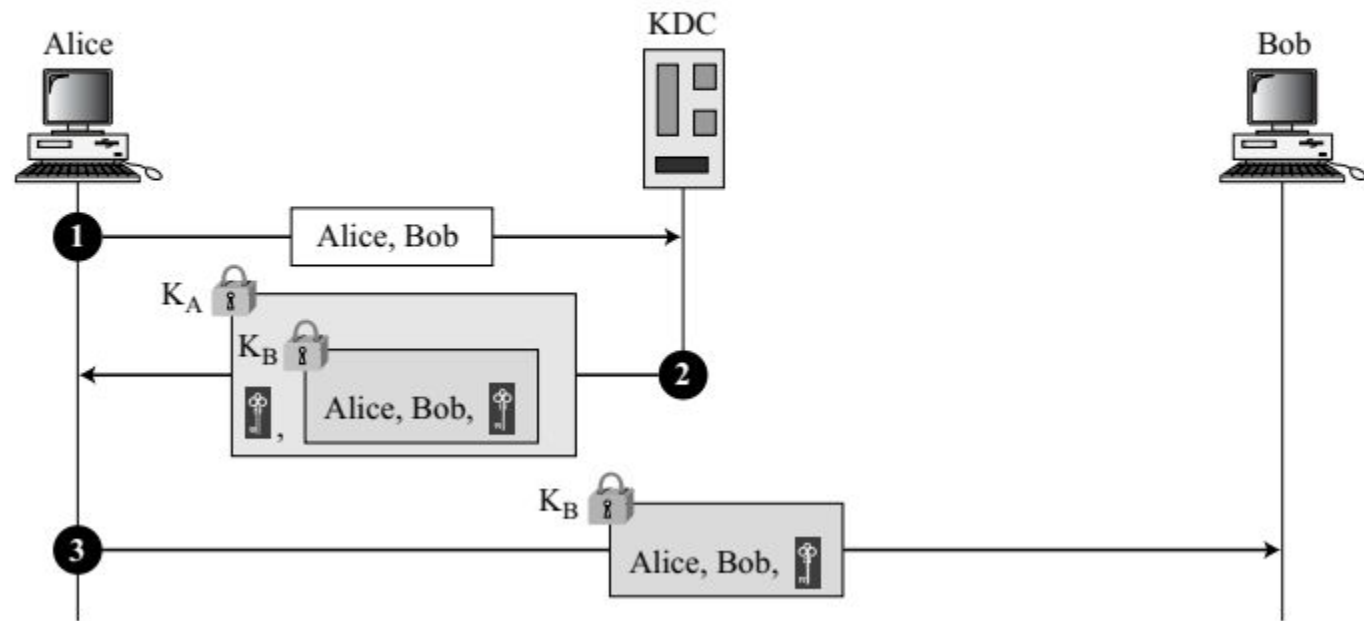
- A KDC creates a secret key for each member.
- This secret key can be used only between the member and the KDC, not between two members.
- If Alice needs to communicate secretly with Bob, she needs a secret key between herself and Bob.
- A KDC can create a session key between Alice and Bob, using their keys with the center.

A Simple Protocol Using a KDC

- Alice sends a plaintext message to the KDC to obtain a symmetric session key between Bob and herself. The message contains her registered identity (the word Alice in the figure) and the identity of Bob (the word Bob in the figure).
- The KDC creates a ticket. The ticket is encrypted using Bob's key. The ticket contains the identities of Alice and Bob and the session key (KAB). The ticket with a copy of the session key is sent to Alice. Alice receives the message, decrypts it, and extracts the session key. She cannot decrypt Bob's ticket; the ticket is for Bob, not for Alice.
- Alice sends the ticket to Bob. Bob opens the ticket and knows that Alice needs to send messages to him using KAB as the session key.

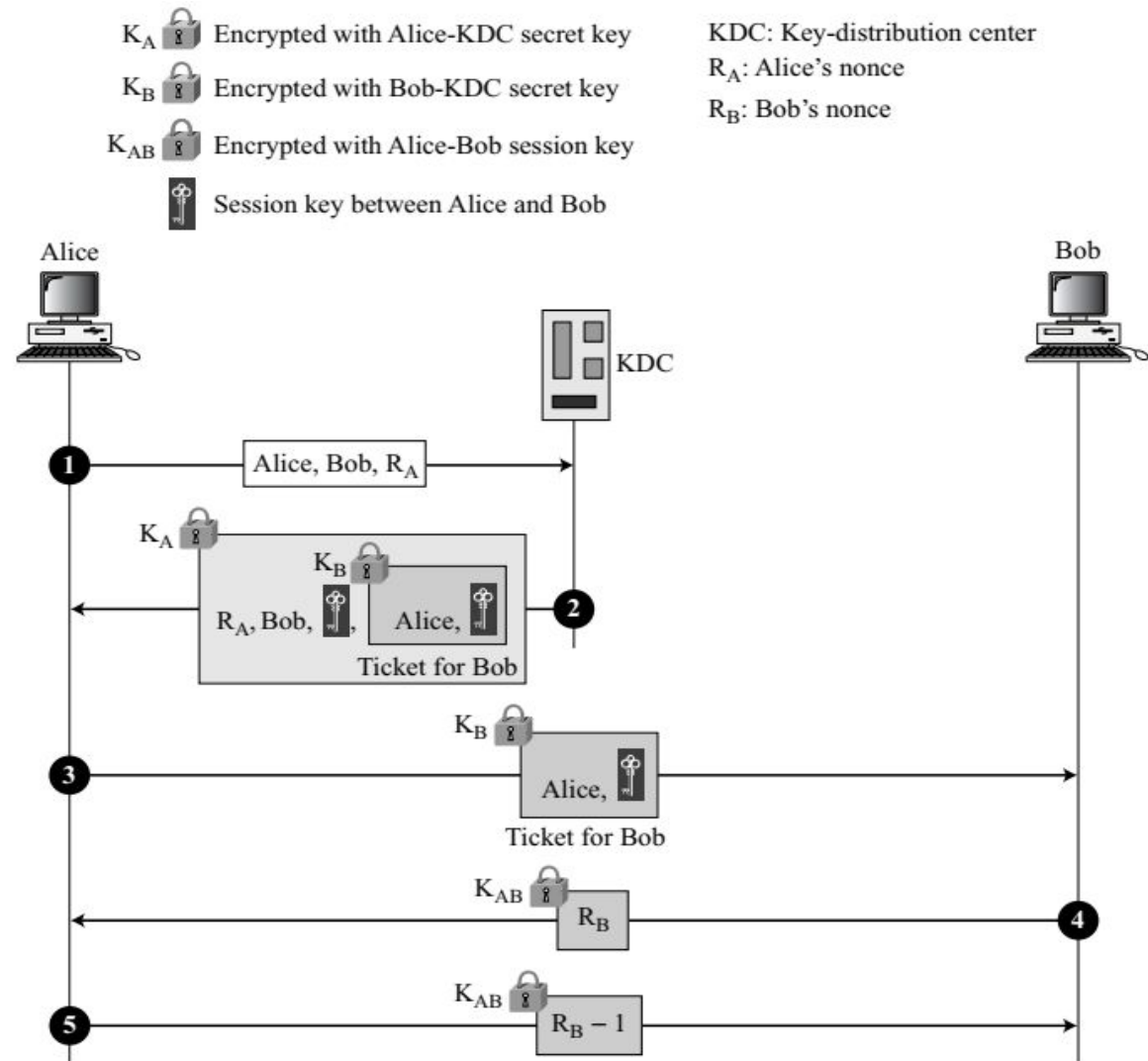
K_A  Encrypted with Alice-KDC secret key  Session key between Alice and Bob

K_B  Encrypted with Bob-KDC secret key KDC: Key-distribution center



Needham-Schroeder Protocol

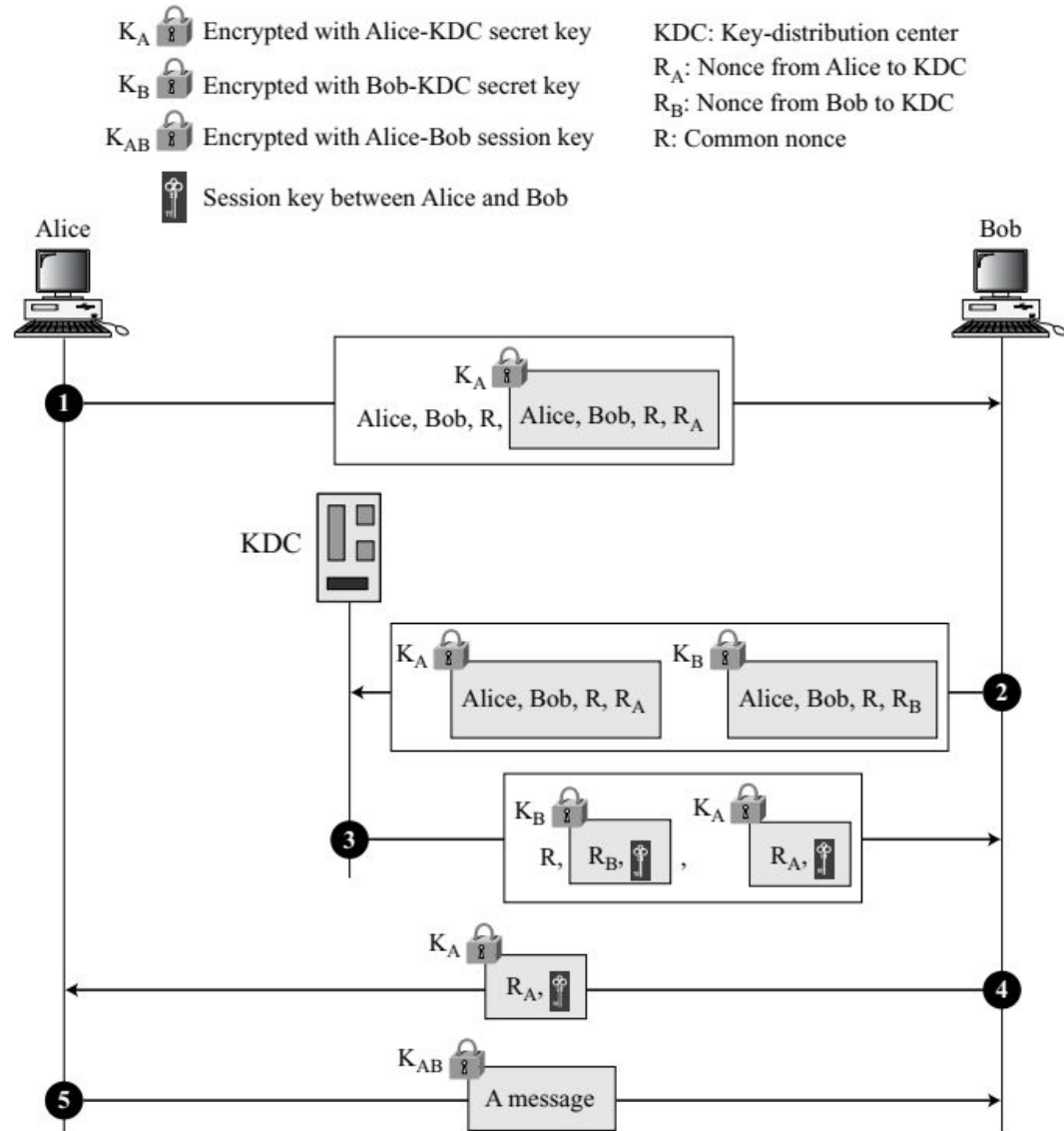
- Another approach is the elegant Needham-Schroeder protocol, which is a foundation for many other protocols.
- This protocol uses multiple challenge-response interactions between parties to achieve a flawless protocol.
- Needham and Schroeder uses two nonces: R_A and R_B .



- Alice sends a message to the KDC that includes her nonce, RA, her identity, and Bob's identity.
- The KDC sends an encrypted message to Alice that includes Alice's nonce, Bob's identity, the session key, and an encrypted ticket for Bob. The whole message is encrypted with Alice's key.
- Alice sends Bob's ticket to him.
- Bob sends his challenge to Alice (RB), encrypted with the session key.
- Alice responds to Bob's challenge. Note that the response carries RB - 1 instead of RB

Otway-Ree's Protocol

- Alice sends a message to Bob that includes a common nonce, R, the identities of Alice and Bob, and a ticket for KDC that includes Alice's nonce R_A (a challenge for the KDC to use), a copy of the common nonce, R, and the identities of Alice and Bob.
- Bob creates the same type of ticket, but with his own nonce R_B . Both tickets are sent to the KDC.
- The KDC creates a message that contains R, the common nonce, a ticket for Alice and a ticket for Bob; the message is sent to Bob. The tickets contain the corresponding nonce, R_A or R_B , and the session key, K_{AB} .
- Bob sends Alice her ticket.
- Alice sends a short message encrypted with her session key K_{AB} to show that she has the session key



KERBEROS

- Kerberos is an authentication protocol, and at the same time a KDC, that has become very popular.
- Several systems, including Windows 2000, use Kerberos.
- It is named after the three-headed dog in Greek mythology that guards the gates of Hades.
- Originally designed at MIT, it has gone through several versions.
- We only discuss version 4, the most popular

Servers

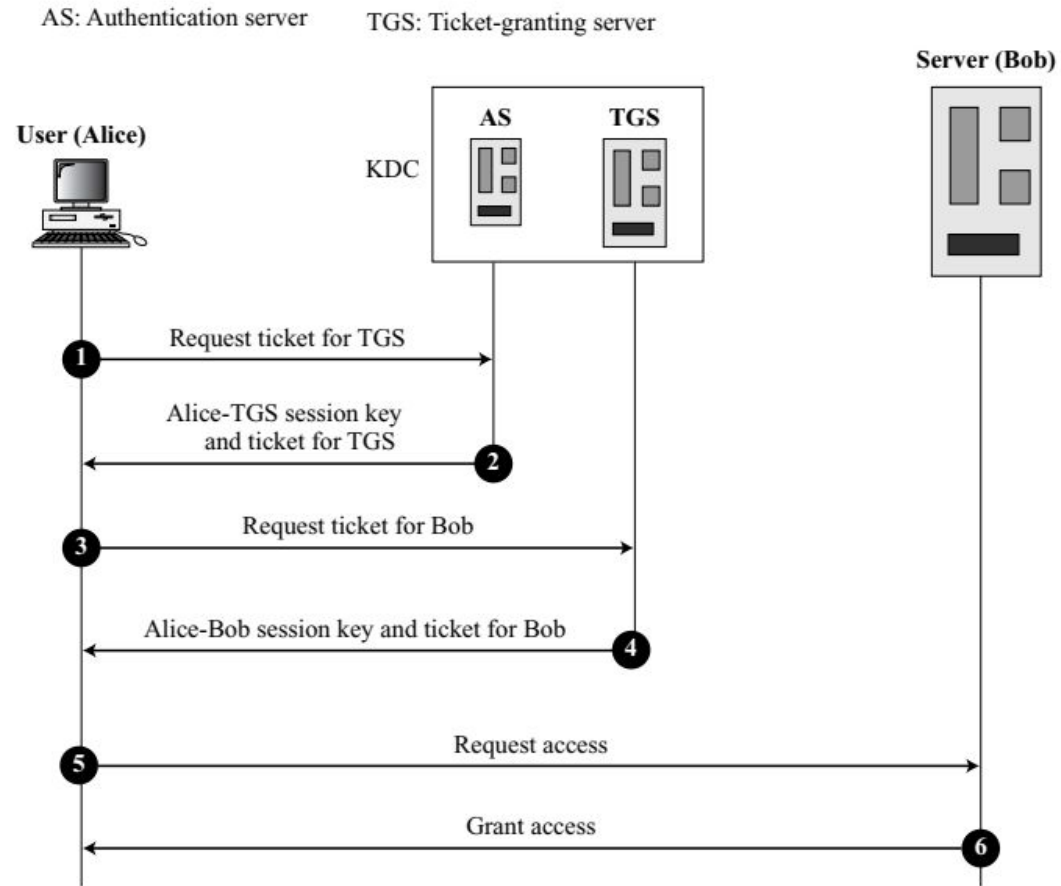
- Three servers are involved in the Kerberos protocol:
 - an authentication server (AS),
 - a ticket-granting server (TGS),
 - and a real (data) server that provides services to others.
- Authentication Server (AS) :
 - AS is the KDC in the Kerberos protocol.
 - Each user registers with the AS and is granted a user identity and a password.
 - The AS has a database with these identities and the corresponding passwords.
 - The AS verifies the user, issues a session key to be used between Alice and the TGS, and sends a ticket for the TGS.

- Ticket-Granting Server (TGS):

- The ticket-granting server (TGS) issues a ticket for the real server (Bob).
- It also provides the session key (KAB) between Alice and Bob.
- Kerberos has separated user verification from the issuing of tickets.
- In this way, though Alice verifies her ID just once with the AS, she can contact the TGS multiple times to obtain tickets for different real servers.

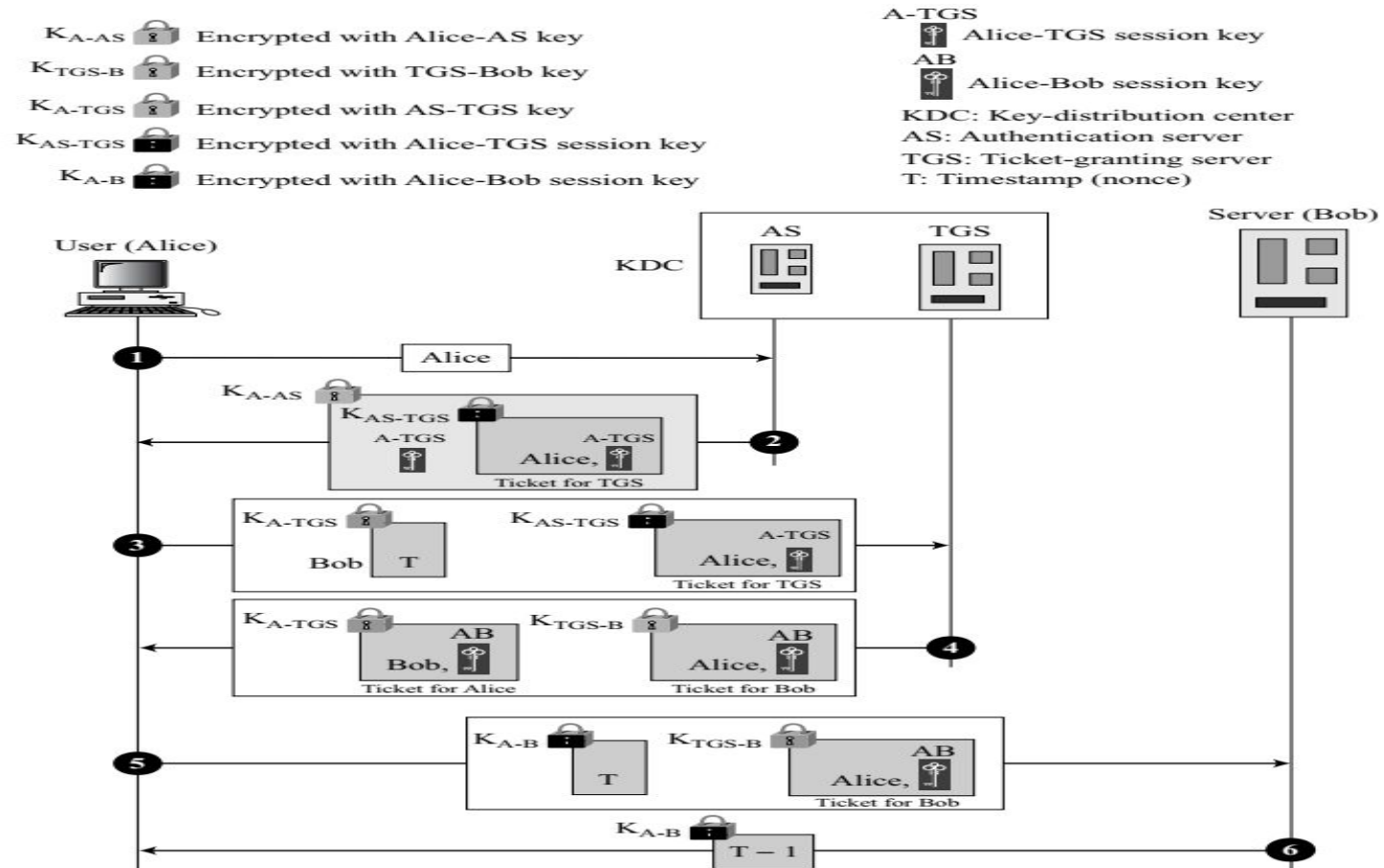
- Real Server:

- The real server (Bob) provides services for the user (Alice).
- Kerberos is designed for a client-server program, such as FTP, in which a user uses the client process to access the server process.
- Kerberos is not used for person-to-person authentication.



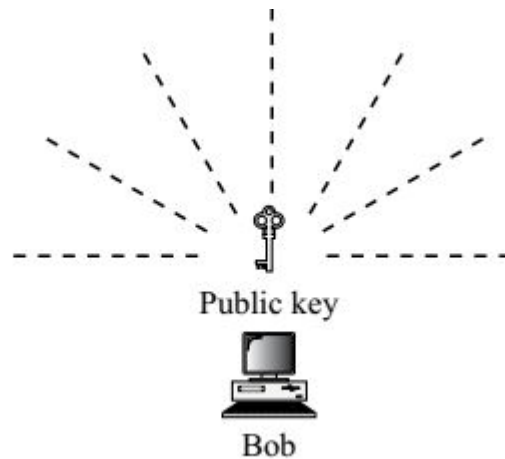
- Alice sends her request to the AS in plain text using her registered identity.
- The AS sends a message encrypted with Alice's permanent symmetric key, KA-AS. The message contains two items: a session key, KA-TGS, that is used by Alice to contact the TGS, and a ticket for the TGS that is encrypted with the TGS symmetric key, KAS-TGS. Alice does not know KA-AS, but when the message arrives, she types her symmetric password. The password and the appropriate algorithm together create KA-AS if the password is correct. The password is then immediately destroyed; it is not sent to the network and it does not stay in the terminal. It is used only for a moment to create KA-AS. The process now uses KA-AS to decrypt the message sent. KA-TGS and the ticket are extracted.
- Alice now sends three items to the TGS. The first is the ticket received from the AS. The second is the name of the real server (Bob), the third is a timestamp that is encrypted by KA-TGS. The timestamp prevents a replay by Eve.

- 4. Now, the TGS sends two tickets, each containing the session key between Alice and Bob, K_{A-B} . The ticket for Alice is encrypted with K_{A-TGS} ; the ticket for Bob is encrypted with Bob's key, K_{TGS-B} . Note that Eve cannot extract K_{A-B} because Eve does not know K_{A-TGS} or K_{TGS-B} . She cannot replay step 3 because she cannot replace the timestamp with a new one (she does not know K_{A-TGS}). Even if she is very quick and sends the step 3 message before the timestamp has expired, she still receives the same two tickets that she cannot decipher.
- 5. Alice sends Bob's ticket with the timestamp encrypted by K_{A-B} .
- 6. Bob confirms the receipt by adding 1 to the timestamp. The message is encrypted with K_{A-B} and sent to Alice



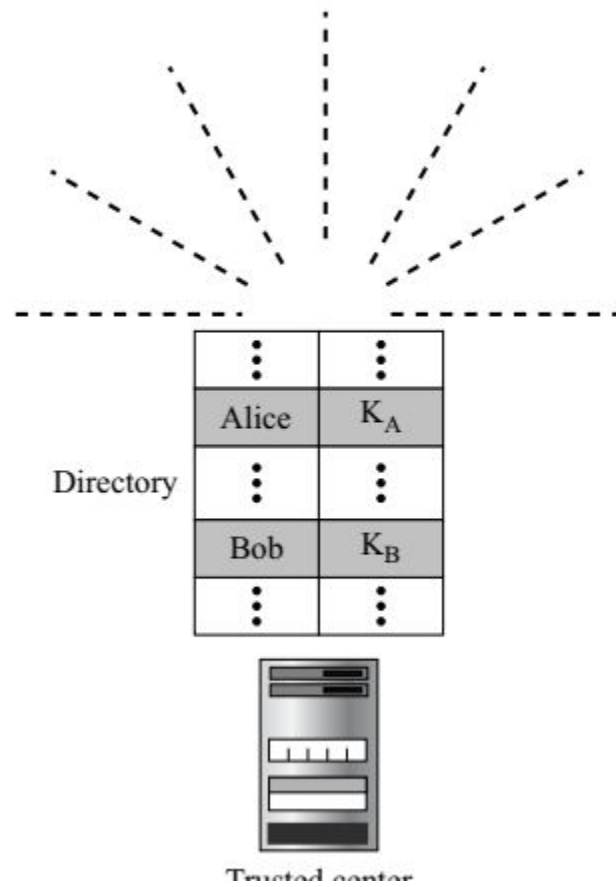
PUBLIC-KEY DISTRIBUTION

- In public-key cryptography, everyone has access to everyone's public key; public keys are available to the public.
- Public Announcement



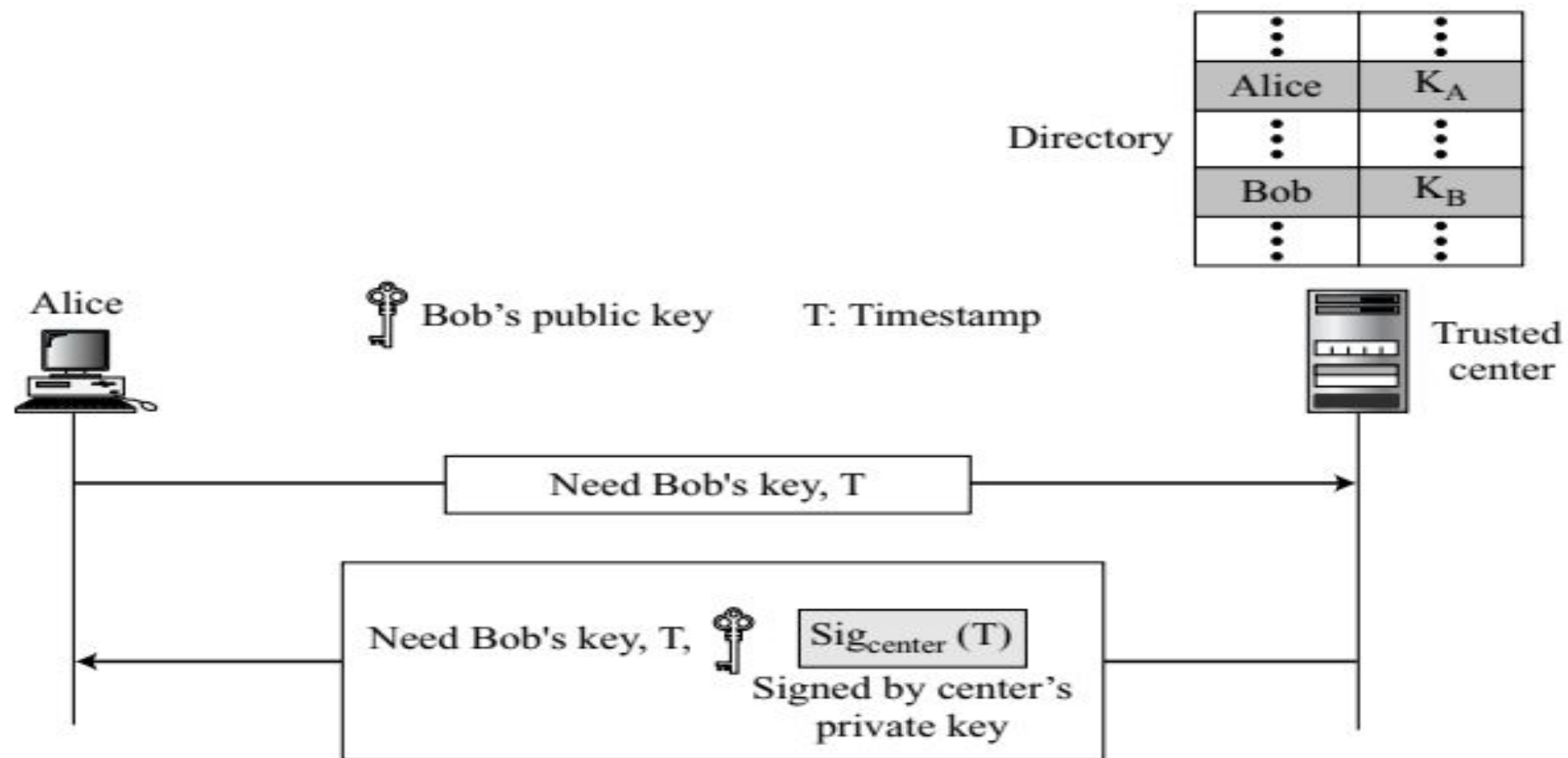
Trusted Center

- 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.



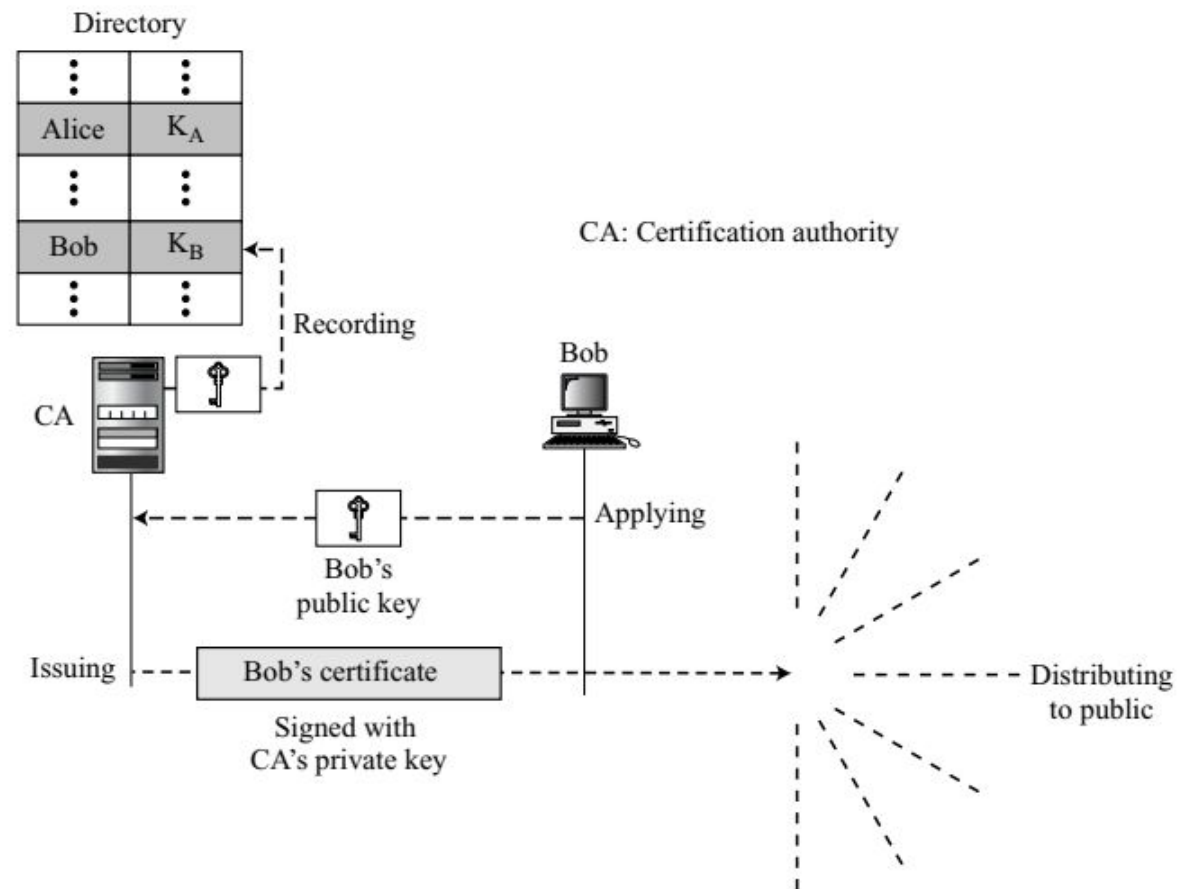
Controlled Trusted Center

- 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.



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; he wants people to know his public key, and he wants no one to 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.
- The CA has a well-known public key itself that cannot be forged.
- The CA checks Bob's identification. 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.



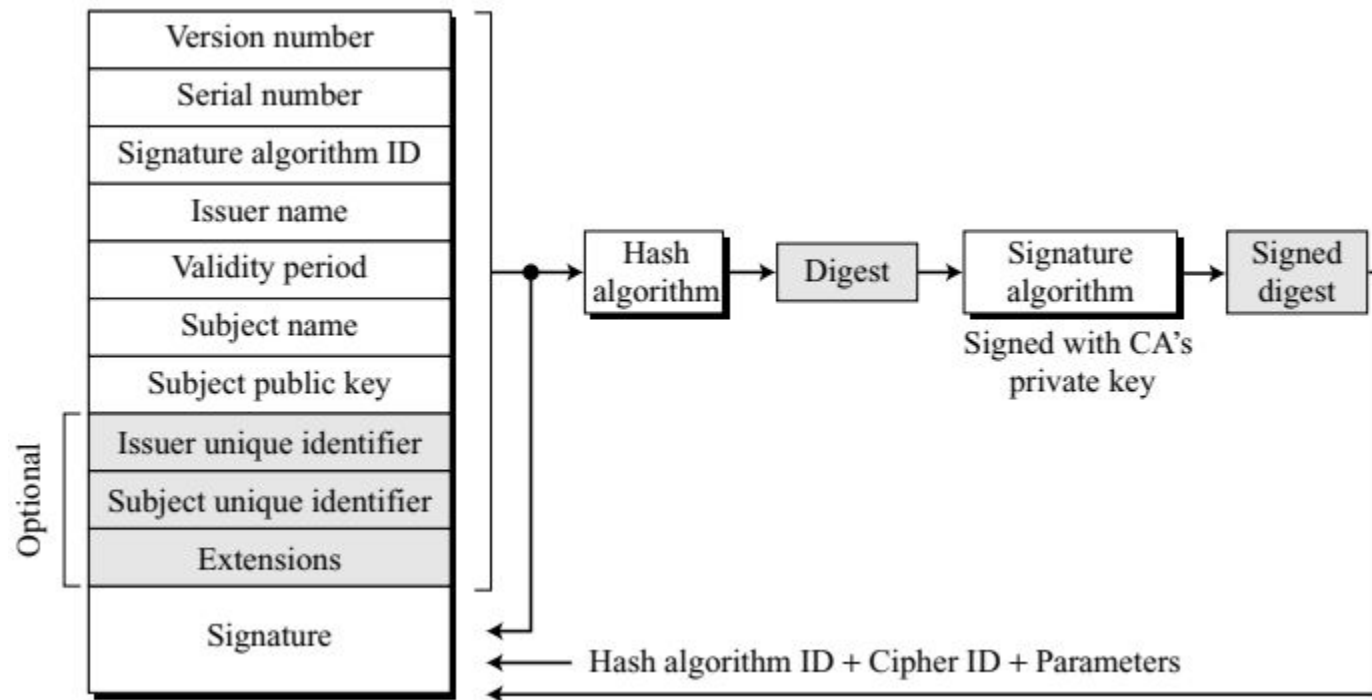
X.509

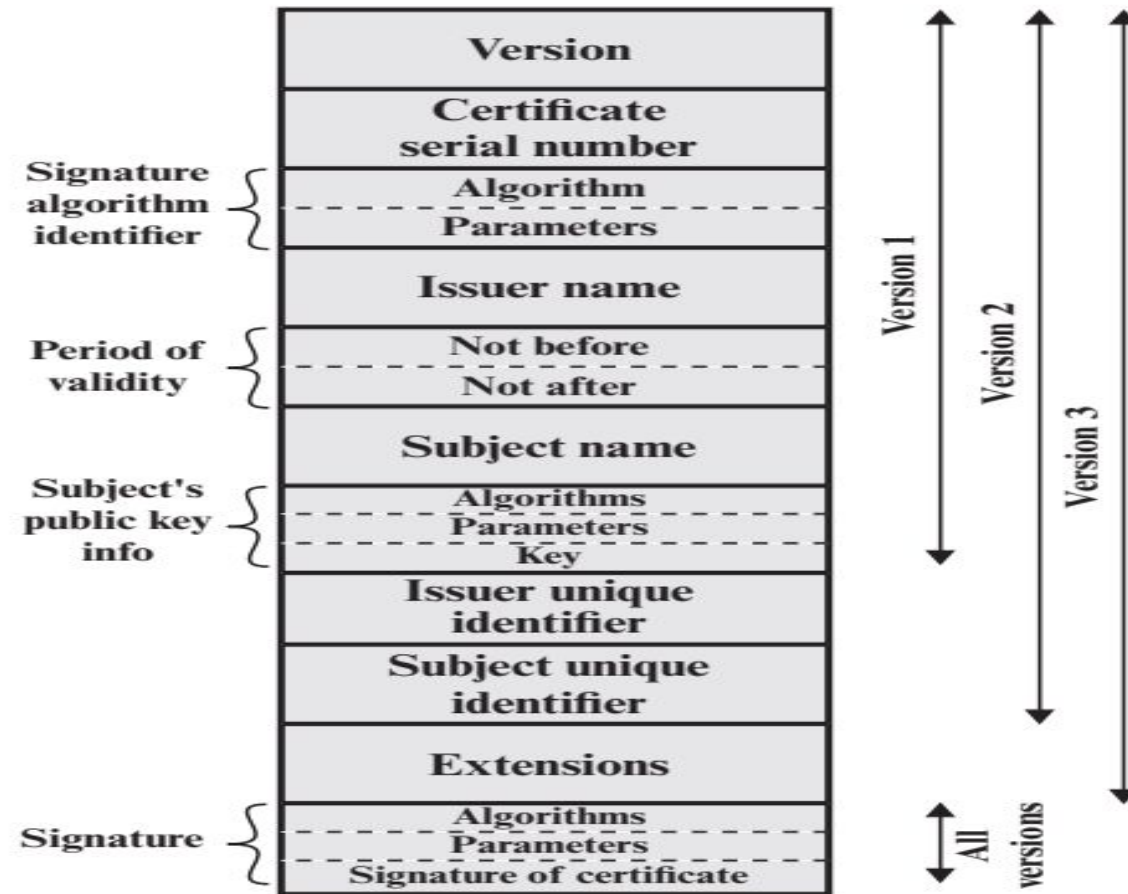
- Although the use of a CA has solved the problem of public-key fraud, it has created a side-effect.
- Each certificate may have a different format.
- If Alice wants to use a program to automatically download different certificates and digests belonging to different people, the program may not be able 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.
- To remove this side effect, the ITU has designed X.509, a recommendation that has been accepted by the Internet with some changes.
- X.509 is a way to describe the certificate in a structured way.

Certificate

- **Version number:** This field defines the version of X.509 of the certificate. The version number started at 0; the current version (third version) is 2.
- **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 .
- **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.
- **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.
- **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.





(a) X.509 certificate

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.

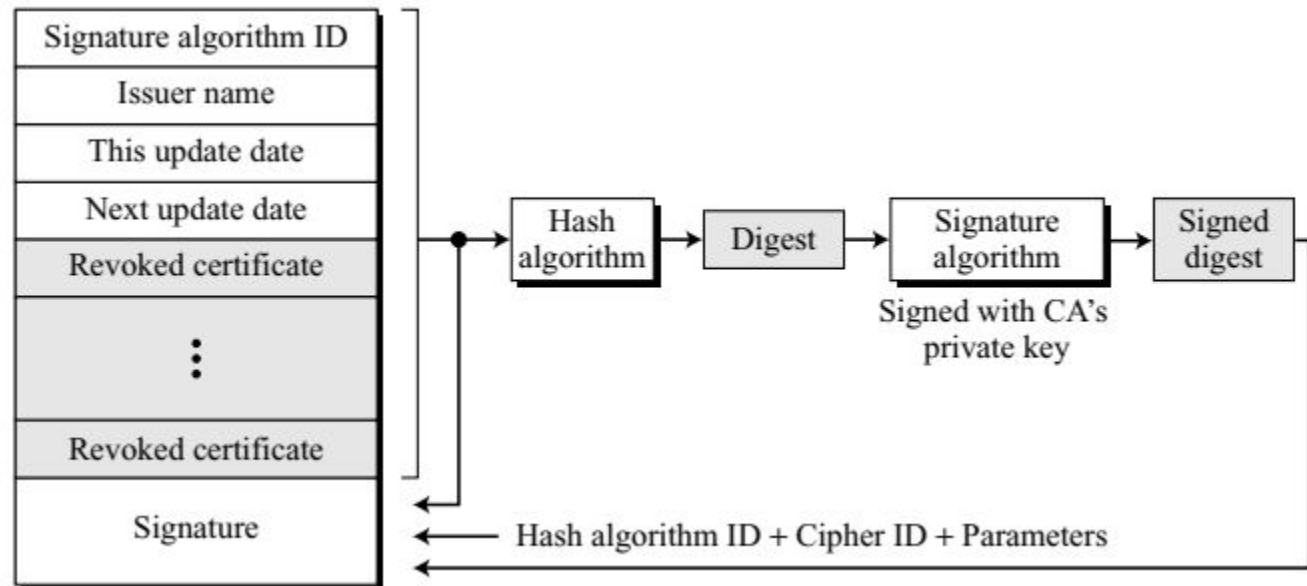
Certificate Revocation

- 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.

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.
- **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.



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)

Duties

- Several duties have been defined for a PKI. The most important ones are shown in Figure 15.19.
- 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.