Institute of Information Technology

Jahangirnagar University

জাহাঙ্গীরনগর বিশ্ববিদ্যালয়

IT-4259: Computer Network Security

for

4th Year 2nd Semester of B.Sc (Honors) in IT (5th Batch)

Lecture: 12

Key Management & Certification

Prepared by:

K M Akkas Ali

akkas_khan@yahoo.com, akkas@juniv.edu

Associate Professor

Institute of Information Technology (IIT)

Jahangirnagar University, Dhaka-1342

Prepared by: K M Akkas Ali, Assistant Prof

Objectives of this Lecture:

- To explain the need for a key-distribution center.
- To show how a KDC can create a session key between two parties.
- To show how two parties can use a symmetric-key agreement protocol to create a session key between themselves without using the service of a KDC.
- To describe Kerberos as a KDC and an authentication protocol.
- To describe symmetric-key agreement protocol.
- To explain the need for digital certificates and certification authorities for public key distribution.
- To introduce the idea of a Public-Key Infrastructure (PKI) and explain some of its duties.

Prepared by: K M Akkas Ali, Assistant Professor, IIT, JU

Introduction

Trusted Intermediaries:

Symmetric key problem:

- How do two entities establish shared secret key over network?

Solution:

- Trusted key distribution center (KDC) acting as intermediary between entities.

Public key problem:

- When Alice obtains Bob's public key (from web site, e-mail, diskette), how does she know it is Bob's public key, not Trudy's?

Solution:

- Trusted certification authority (CA)

Key-Distribution Center: KDC

- In symmetric-key cryptography, a shared secret key is needed to be exchanged between two parties involved in a communication.
- If Alice and Bob want to communicate, they need a way to exchange a secret key between them; if Alice wants to communicate with one million people, how can she exchange one million keys with one million people? Using the Internet is definitely not a secure method. It is obvious that we need an efficient way to maintain and distribute secret keys.
- A practical solution to maintain and distribute secret keys 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, as shown in the figure below.

- A secret key is established between the KDC and each member.
- Alice has a secret key K_{Alice} with the KDC; Bob has a secret key K_{Bob} with the KDC; and so on.
- They communicate with each other via the KDC.

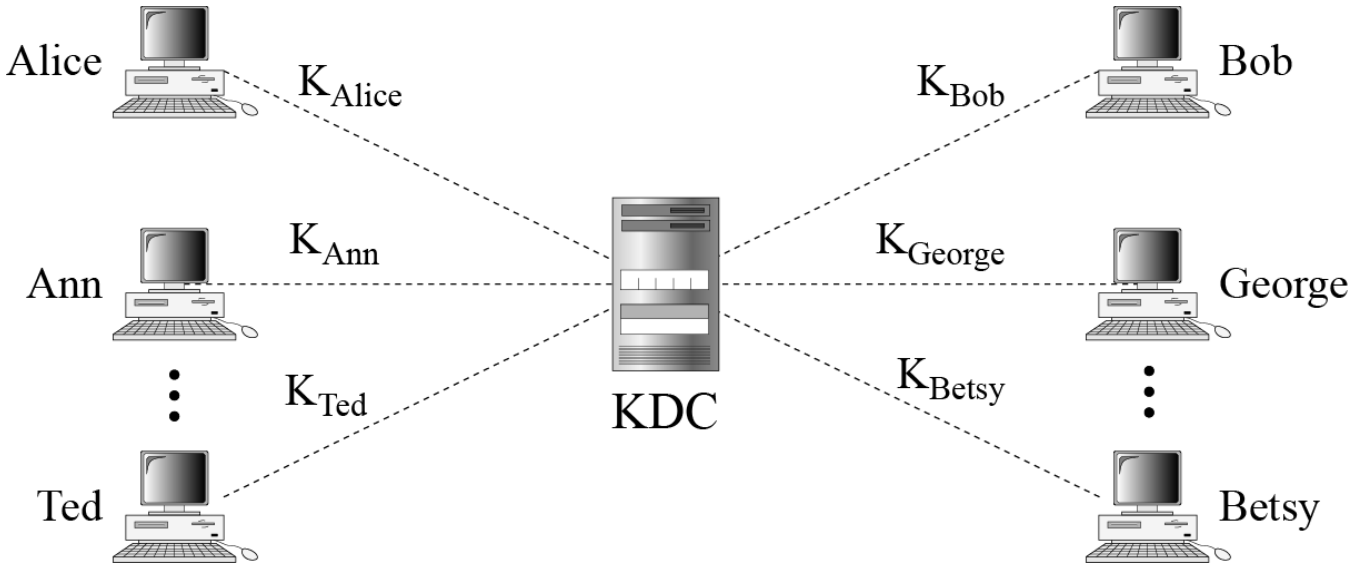

Figure: Key-distribution center (KDC)

Key-Distribution Center: KDC

Q: How can Alice send a confidential message to Bob using the KDC?

- 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 parties.
- The established session key between Alice and Bob with the KDC is used to authenticate Alice and Bob to the KDC which prevents Eve from impersonating either of them.

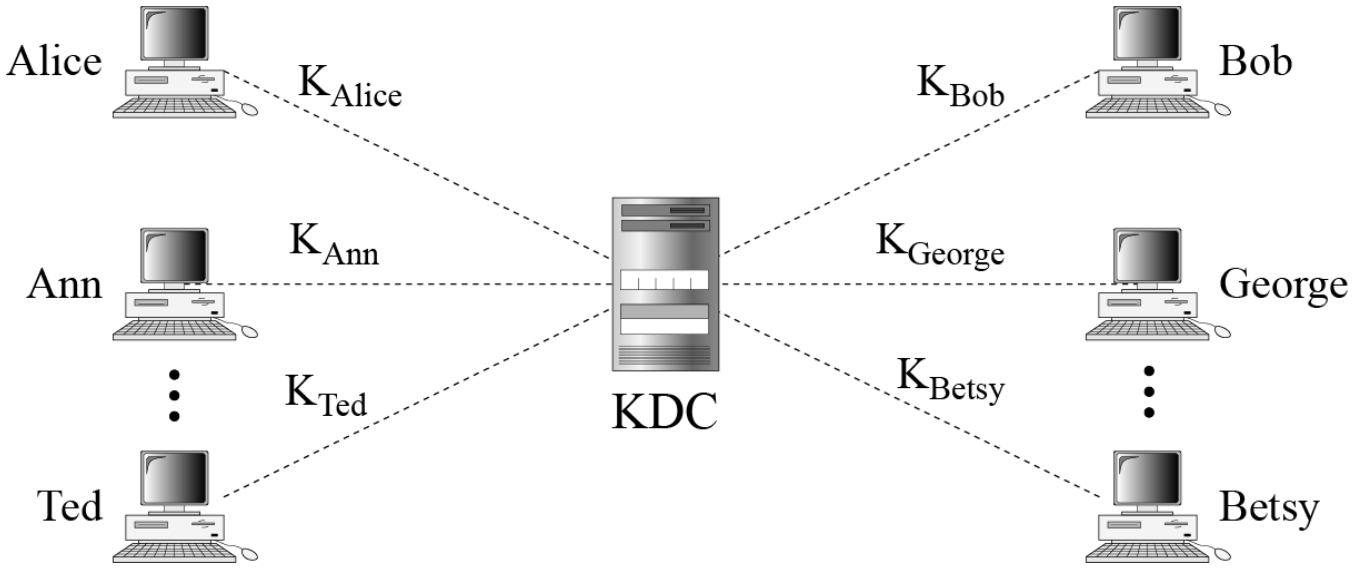

Figure: Key-distribution center (KDC)

Key-Distribution Center: KDC

Session Key:

- The KDC creates a secret key for each member which can be used only between the member and the KDC to authenticate each member with the KDC, not between two members .
- If Alice needs to communicate secretly with Bob, she needs a secret key between herself and Bob.
- The KDC can create a session key between Alice and Bob, using their keys with the center. The session key is used to authenticate Alice and Bob to each other. After authentication, they can exchange message. After communication is terminated, the session key is no longer useful. If Alice and Bob again need to communicate, another session key is established between them.
- Therefore, a session symmetric key between two parties is used only once.

Key-Distribution Center: KDC

Q: How is a session key established between Alice and Bob for communication?

- The KDC creates a temporary secret key for each member which can be used only once between the member and the KDC, not between two members . 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 parties.
- The established session key between Alice and Bob with the KDC is used to authenticate Alice and Bob to the KDC which prevents Eve from impersonating either of them.

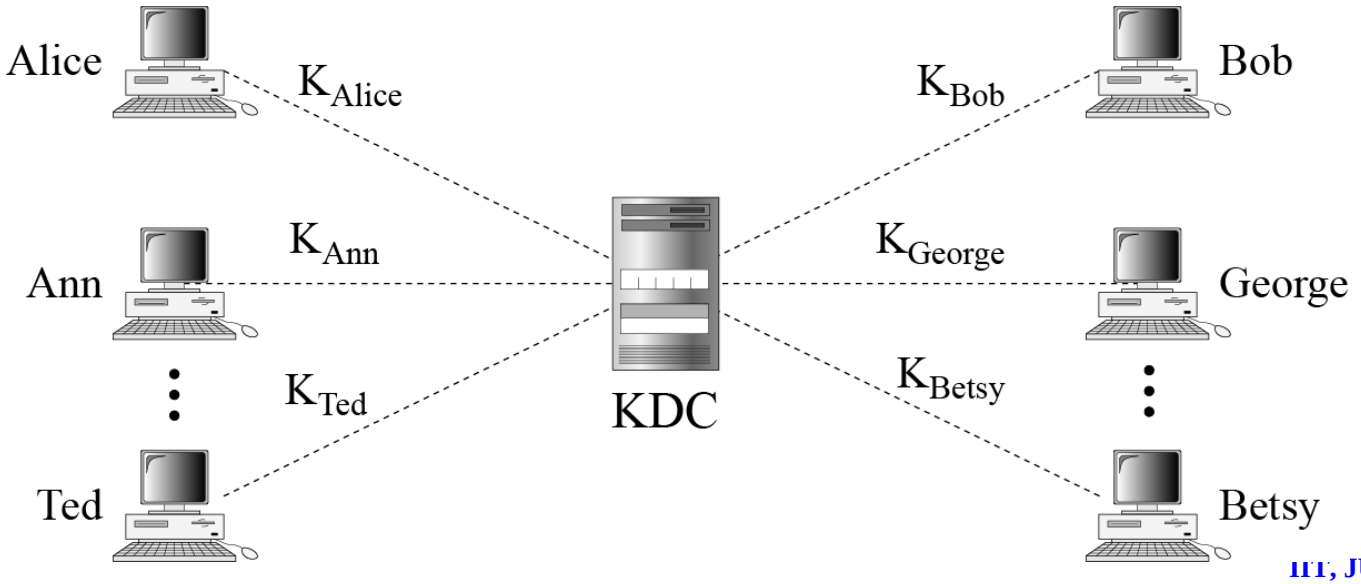

Figure: Key-distribution center (KDC)

Protocols for Creating Session Key using KDCs

- There are several different approaches to create the session key.

Approach-1: Creating Session key using a KDC

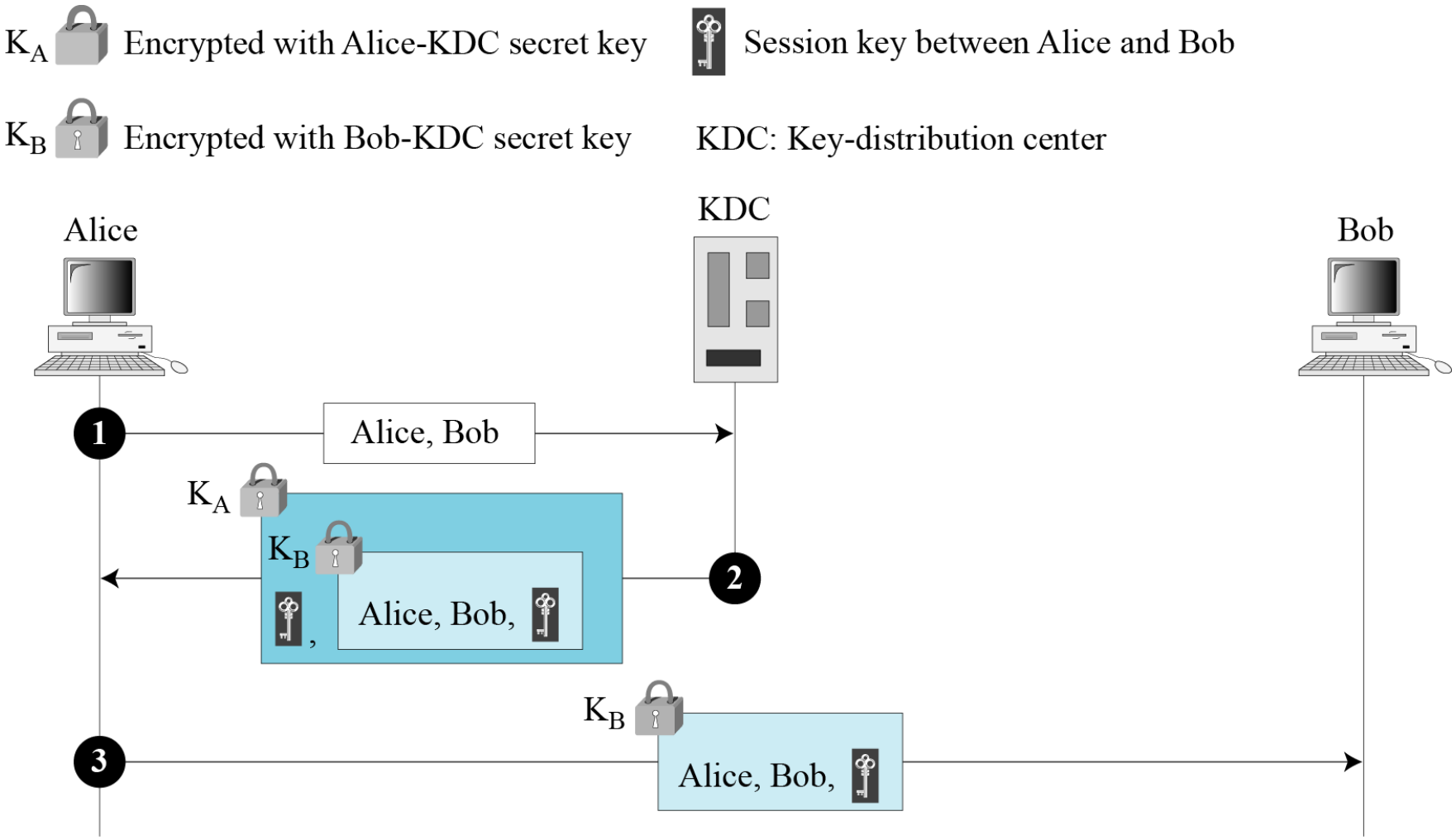

Figure: ***Creating session key using KDC***

Protocols for Creating Session Key using KDCs

- The steps for creating session key using the first approach is summarized below:

1. Alice sends an unencrypted plaintext message to the KDC to obtain a symmetric session key between Bob and herself.
   - The message contains Alice's registered identity (Alice in this case) and the identity of Bob (Bob in this case).
2. After receiving the message from Alice, the KDC creates a ticket which contains the identities of Alice and Bob, and the session key (K_{AB}). The KDC then encrypts the ticket using Bob's key (K_B).

   The session key along with the encrypted ticket is again encrypted using Alice's key K_A. Then it is sent to Alice. Alice receives the message. Decrypts it, and extracts the session key.
   - Alice can not decrypt the ticket, which is only for Bob.
   - In this message, Alice is actually authenticated to the KDC, because only she can open the whole message using her secret key with the KDC.
3. After getting the session key, Alice sends the encrypted ticket to Bob. Bob decrypts the ticket using his key K_B and knows that Alice needs to send message to him using K_{AB} as the session key.
   - In this message, Bob is authenticated to the KDC, because only he can open the ticket.
   - Since Bob is authenticated to the KDC, he is also authenticated to Alice, who trusts the KDC. In the same way, Alice is also authenticated to Bob, Bob trusts the KDC and the KDC has sent Bob the ticket that includes the identity of Alice.

Note: *Unfortunately, this protocol has a flaw. Eve can use the replay attack. That is, she can save the message in step 3 and replay it later.*

Protocols for Creating Session Key using KDCs

Approach-2: Needham-Schroeder Protocol

- This protocol is a foundation for many other protocols that uses multiple challenge-response interactions between parties to achieve a flawless protocol.
- Needham and Schroeder uses two nonce: R_A and R_B.
- Figure below shows the five steps used in this protocol.

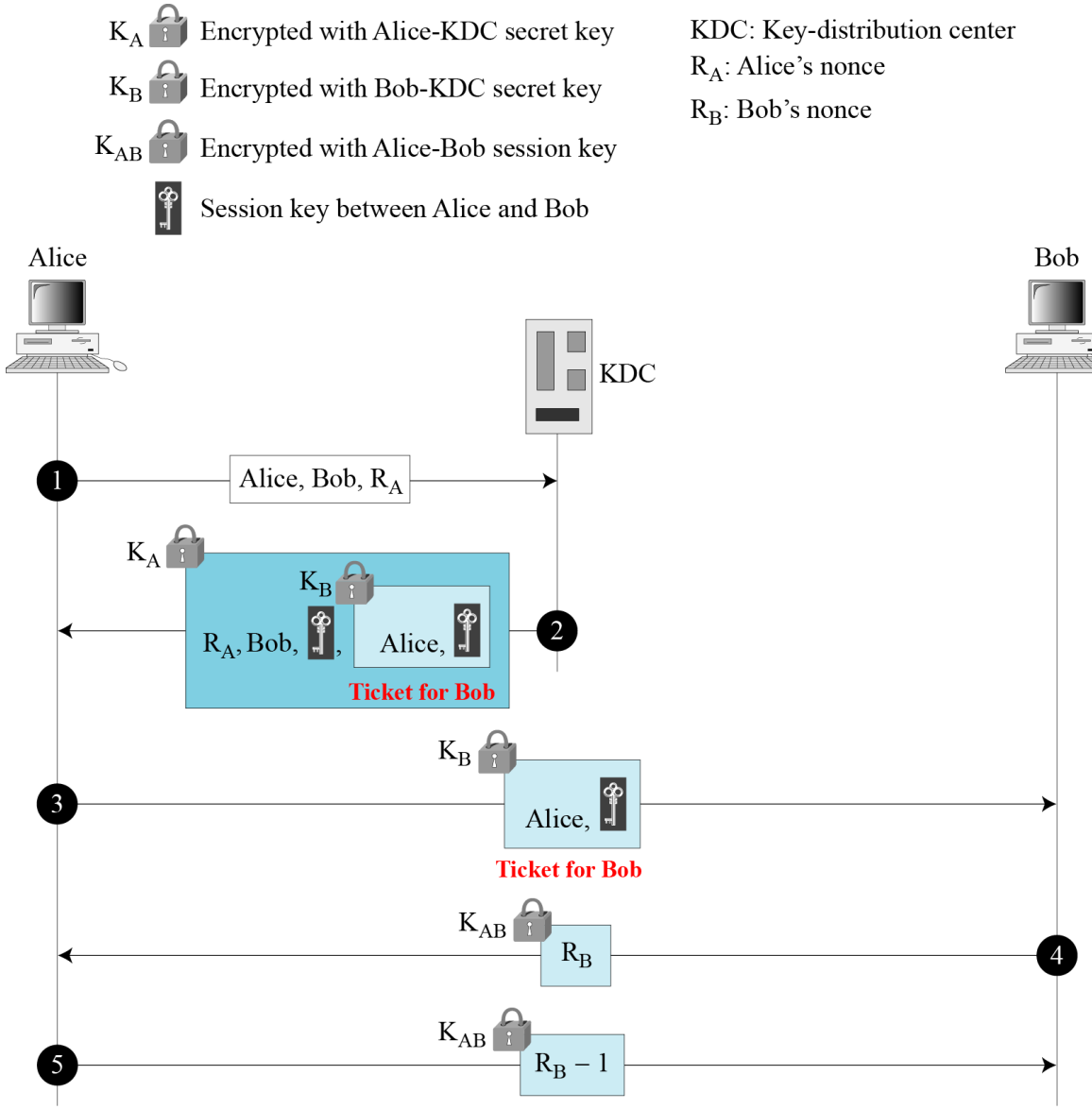

Figure: Needham-Schroeder protocol

Protocols for Creating Session Key using KDCs

Approach-2: Needham-Schroeder Protocol (continue...)

- The steps for creating session key using Needham-Schroeder protocol is summarized below:

1. Alice sends a message to the KDC that includes her nonce (R_A), her identity, and Bob's identity.
2. After receiving the message from Alice, the KDC creates a ticket which contains the identities of Alice, and the session key K_{AB} between Alice and Bob. The KDC then encrypts the ticket using Bob's key (K_B).

   Alice's nonce R_A, Bob's identity, the session key K_{AB}, and the encrypted ticket for Bob is again encrypted using Alice's key K_A. Then it is sent to Alice. Alice receives the message. Decrypts it, and extracts the session key.

   - Alice can not decrypt the ticket, which is only for Bob.
   - In this message, Alice is actually authenticated to the KDC, because only she can open the whole message using her secret key with the KDC.

1. After getting the session key, Alice sends the encrypted ticket to Bob. Bob decrypts the ticket using his key K_B and comes to know the session key.
2. Bob encrypts his challenge R_B with the session key K_{AB} and sends the encrypted challenge to Alice.
3. Alice decrypts Bob's encrypted challenge and extracts it. She then decrease Bob's challenge by 1 (i.e. R_B-1), encrypts it using the session key and finally sends it back to Bob.

Note: *The response carries $\mathcal{R}_B - 1$ instead of $\mathcal{R}_B$.*

Prepared by: K M Akkas Ali, Assistant Professor, IIT, JU

Protocols for Creating Session Key using KDCs

Approach-3: Otway-Rees Protocol

- The steps for creating session key using this approach is given below:

1. Alice sends a message to Bob that includes a common nonce R, the identities of Alice and Bob, and an encrypted 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.
2. Bob creates the same type of ticket, but with his own nonce R_B. Both tickets are sent to the KDC.
3. The KDC creates a message that contains the common nonce R, 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}.
4. Bob sends Alice her ticket.
5. Alice sends a short message encrypted with her session key K_{AB} to show that she has the session key.

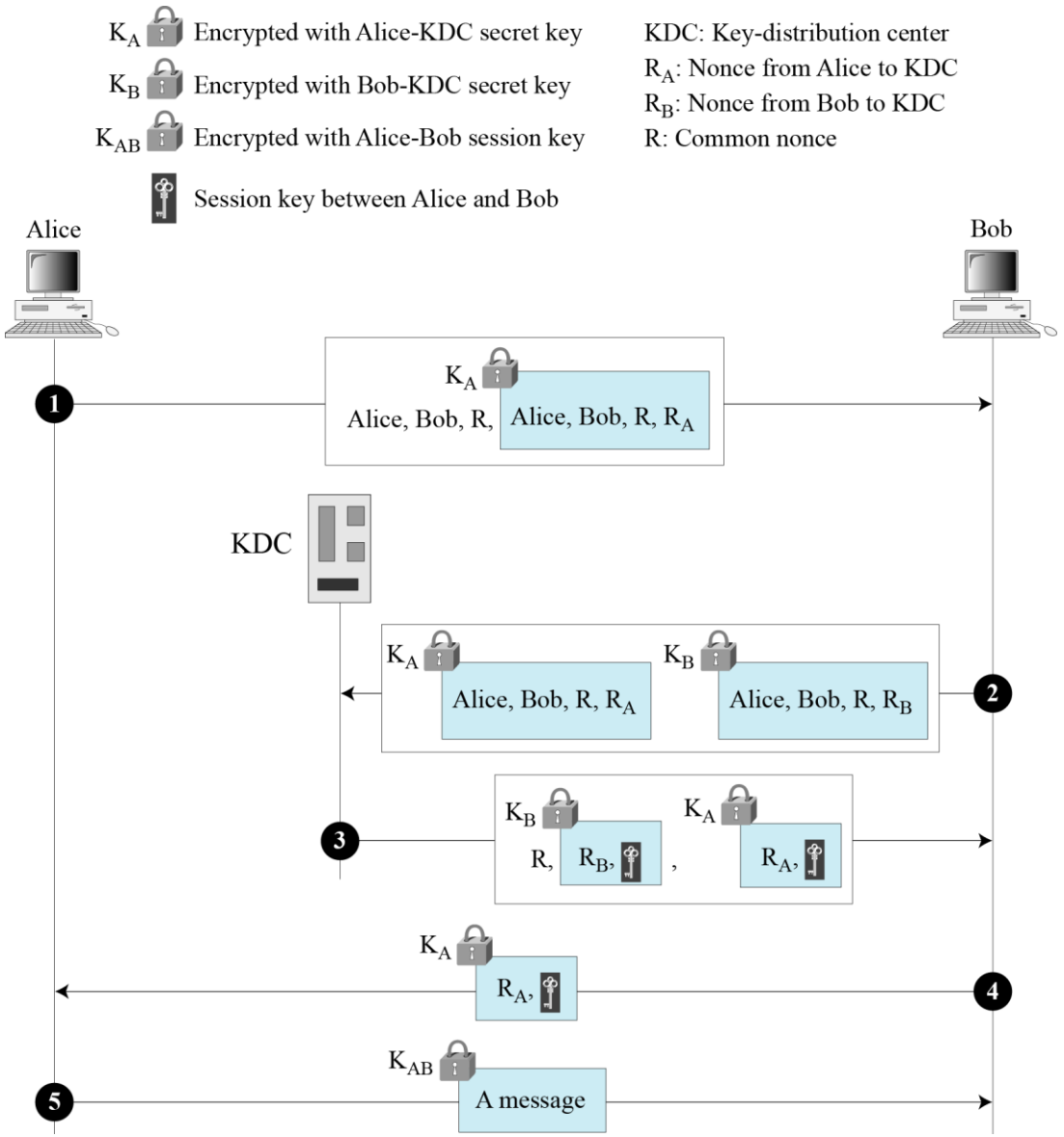

Figure: Otway-Rees protocol

Using Multiple KDCs:

- When the number of people using a KDC increases, the system becomes unmanageable and a bottleneck can result.
- Using multiple KDCs can solve this problem.
- There are two approaches to use multiple KDCs:
    1. Flat multiple KDCs
    2. Hierarchical multiple KDCs

Using Multiple KDCs:

Flat Multiple KDCs:

- In this approach, the world is divided into domains where each domain can have one or more KDCs.
  - If Alice want 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.
- Figure below shows the flat multiple KDCs.

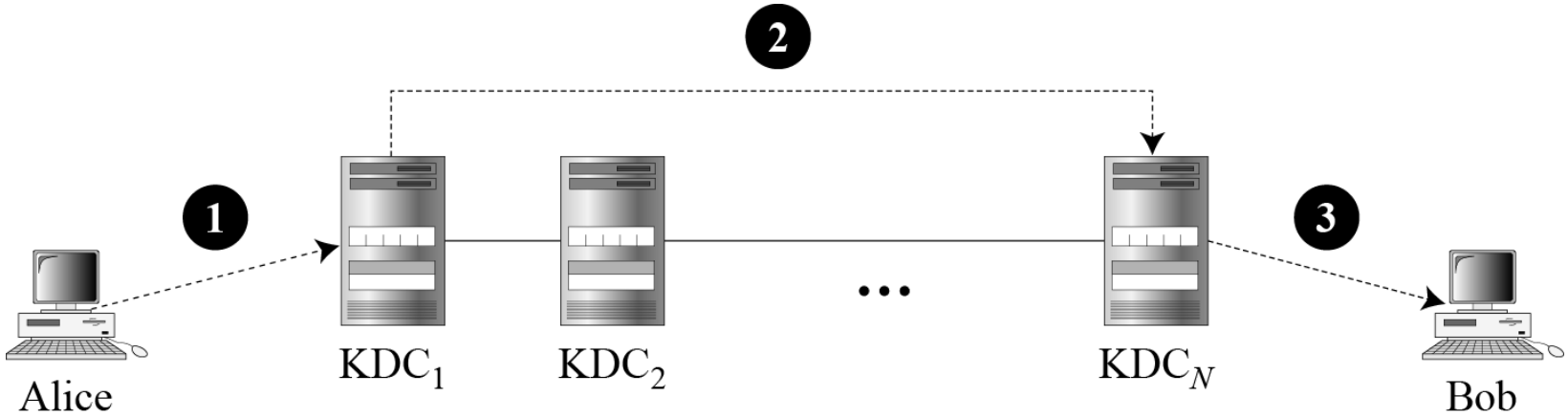

Figure: Flat multiple KDCs

Using Multiple KDCs:

Hierarchical Multiple KDCs:

- In hierarchical multiple KDC system, the concept of flat multiple KDCs can be extended with one or more KDCs at the top of the hierarchy.
- For example, there can be local, national and international KDCs.
  - When Alice needs to communicate with Bob, who lives in another country, she needs 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.
- Figure below shows a configuration of hierarchical multiple KDCs.

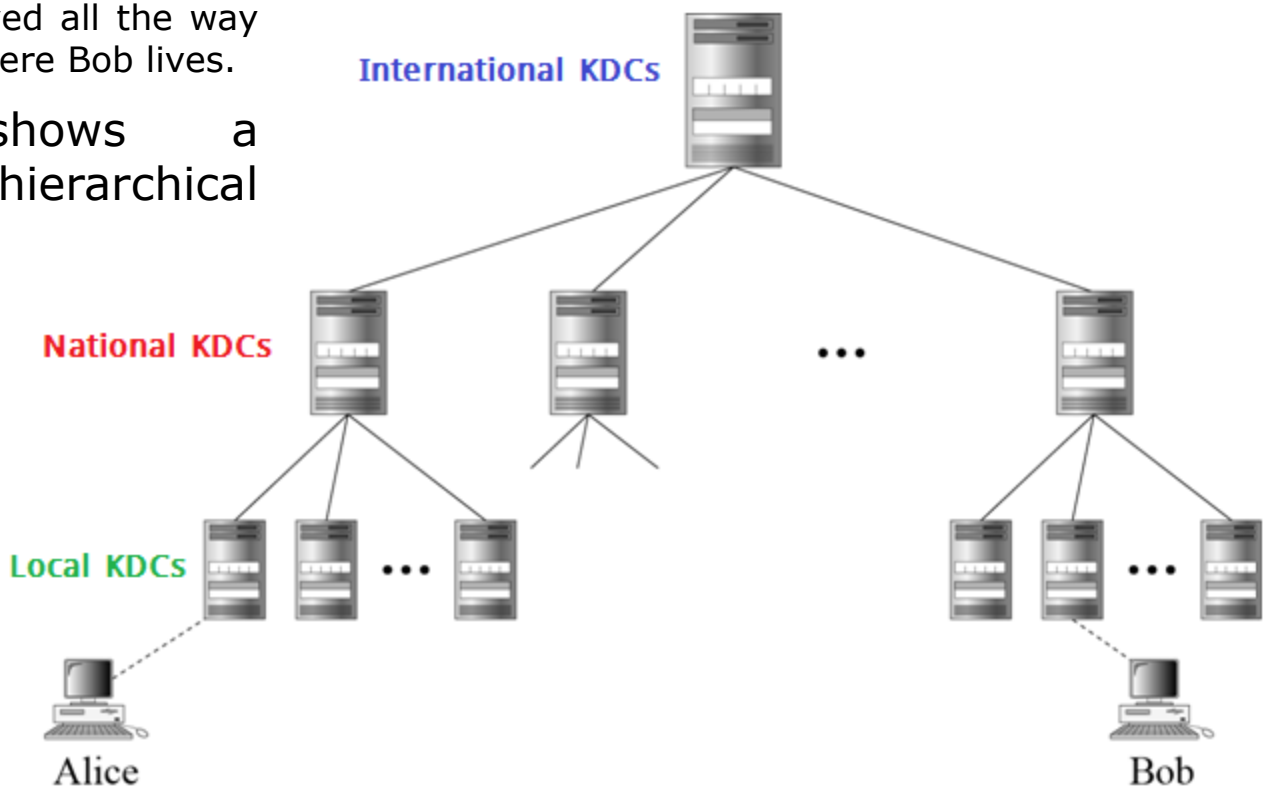

Figure: Hierarchical multiple KDCs

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 so after the three-headed dog in Greek mythology that guards the gates of Hades.
- Originally designed at MIT, it has gone through several versions.

Servers Involved in Kerberos:

- Three servers are involved in the Kerberos protocol:

1. Authentication Server (AS):

- The authentication server (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 ($K_{A\text{-}TGS}$) to be used between Alice and the TGS, and
  - ✓ sends a ticket for the TGS.

2. Ticket-Granting Server (TGS):

- The TGS –
  - ✓ issues a ticket for the real server (Bob).
  - ✓ provides the session key ($K_{A\text{-}B}$) between Alice and Bob.

3. 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.**

Servers Involved in Kerberos:

- Figure below shows the relationship between these three servers.
- In our examples and figures, Bob is the real server and Alice is the user requesting service.

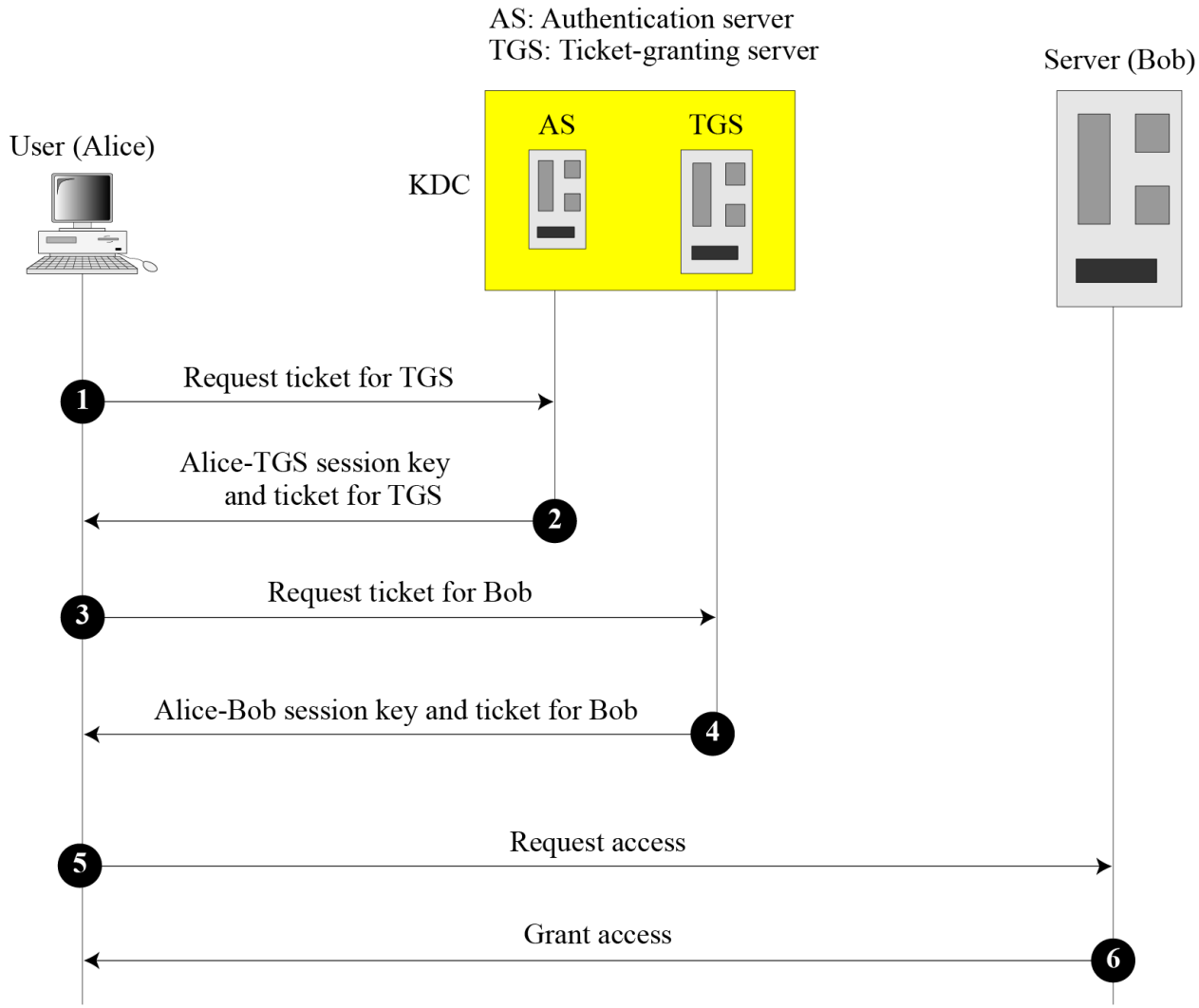

Figure: Kerberos servers

Operation of Kerberos:

In Kerberos, a client process (Alice) can access a process running on the real server (Bob) in six steps, which are summarized below:

1. Alice sends her request to the AS in plain text using her registered identity.
2. The AS sends a message encrypted with Alice's permanent symmetric key, $K_{A\text{-}AS}$.
   - The message contains two items:
     - a session key $K_{A\text{-}TGS}$, that is used by Alice to contact the TGS.
     - a ticket for TGS that is encrypted with the TGS symmetric key $K_{AS\text{-}TGS}$.
   - Alice does not know $K_{A\text{-}AS}$, but when the message arrives, she types her symmetric password. The password and the appropriate algorithm together create $K_{A\text{-}AS}$ 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 $K_{A\text{-}AS}$.
   - The process now uses $K_{A\text{-}AS}$ to decrypt the message sent. $K_{A\text{-}TGS}$ and the ticket are extracted.
3. 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 $K_{A\text{-}TGS}$. The timestamp prevents a replay by Eve.
4. Now, the TGS sends two tickets, each containing the session key $K_{A\text{-}B}$ between Alice and Bob:
   - One ticket for Alice is encrypted with $K_{A\text{-}TGS}$;
   - Another ticket for Bob is encrypted with Bob's key, $K_{TGS\text{-}B}$.
   - Note that Eve cannot extract $K_{A\text{-}B}$ because she does not know $K_{A\text{-}TGS}$ or $K_{TGS\text{-}B}$. She cannot replay step 3 because she cannot replace the timestamp with a new one (she does not know $K_{A\text{-}TGS}$). 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 $K_{A\text{-}B}$.
6. Bob confirms the receipt by adding 1 to the timestamp. The message is encrypted with $K_{A\text{-}B}$ and sent to Alice.

Operation of Kerberos:

The above six steps are shown in the figure below:

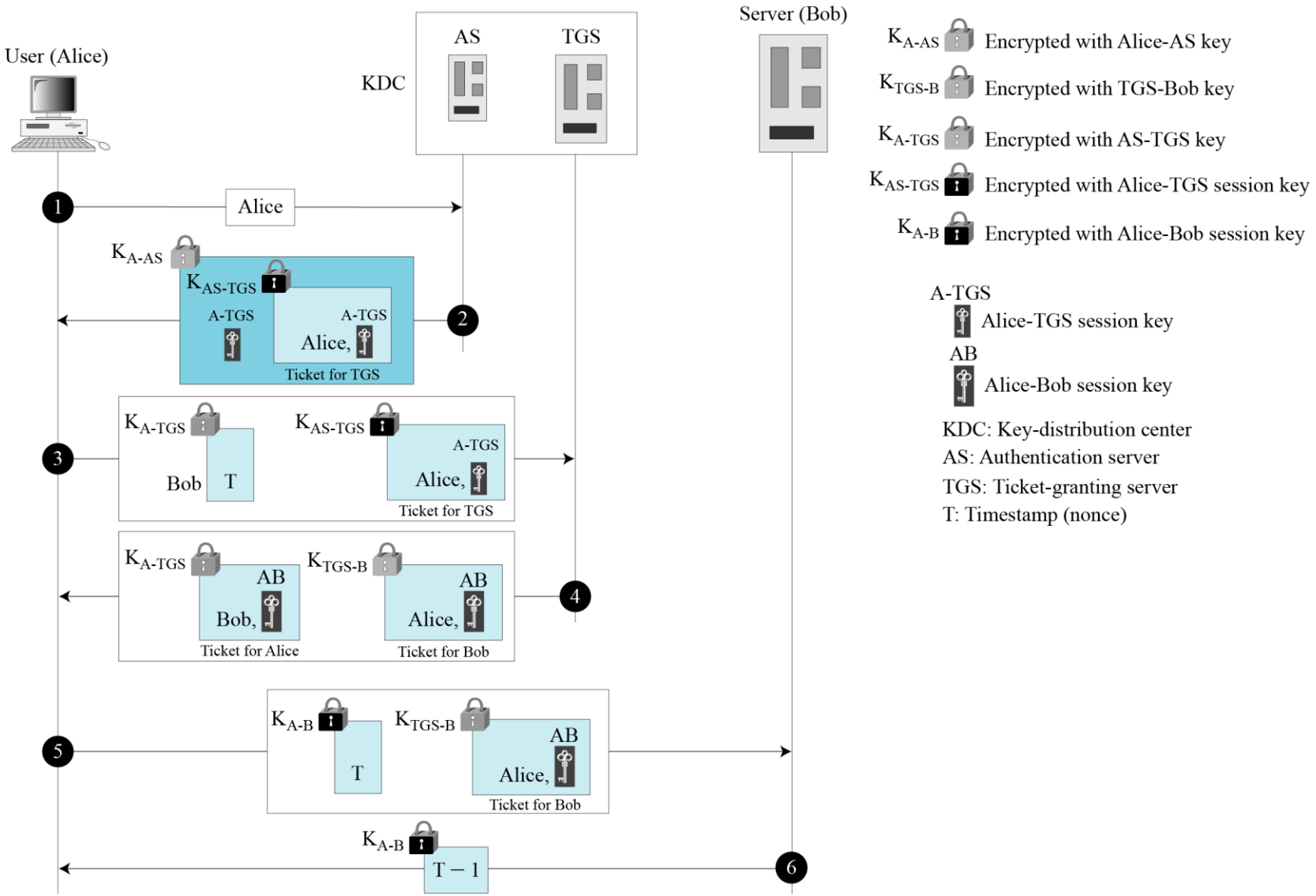

Figure: Kerberos example

Kerberos Version 5:

- After its inception, Kerberos has gone through several versions. Among them, version 4 is the most popular.
- The minor differences between version 4 and version 5 of Kerberos are briefly listed below:

1) Version 5 has a longer ticket lifetime.
2) Version 5 allows tickets to be renewed.
3) Version 5 can accept any symmetric-key algorithm.
4) Version 5 uses a different protocol for describing data types.
5) Version 5 has more overhead than version 4.

Symmetric-key Agreement

- Alice and Bob can create a session key between themselves without using a KDC. This method of session-key creation is referred to as the *symmetric-key agreement*.
- Two most common approaches for symmetric-key agreement are:
    1. Diffie-Hellman Key Agreement
    2. Station-to-Station Key Agreement

Symmetric-key Agreement: Diffie-Hellman Key Agreement

- Suppose Alice and Bob have no keys (shared or public), and want to come up with a joint key which they would use for private key cryptography. The Diffie-Hellman (DH) secret key exchange (SKE) protocol enables them to securely exchange a key without the need of a KDC that can then be used for subsequent encryption of messages.
- In Diffie-Hellman key exchange algorithm, there are two publicly known numbers:

1. a large prime number **q**.
2. an integer **α** called generator where α<q and α is a *primitive root* of q.

- The integer α termed as generator is called a primitive root of the prime number q if the powers of α modulo q generate all the integers from 1 to q-1.
- That is, if the numbers $\alpha \bmod q$, $\alpha^2 \bmod q$, $\alpha^3 \bmod q$, …., $\alpha^{q-1} \bmod q$ are distinct and consist of the integers ranging from 1 through q-1 in some permutation.

Symmetric-key Agreement: Diffie-Hellman Key Agreement

- Suppose two users Alice and Bob wish to exchange a secret key.
- The Diffie-Hellman key exchange method used in this case is shown in the figure below:

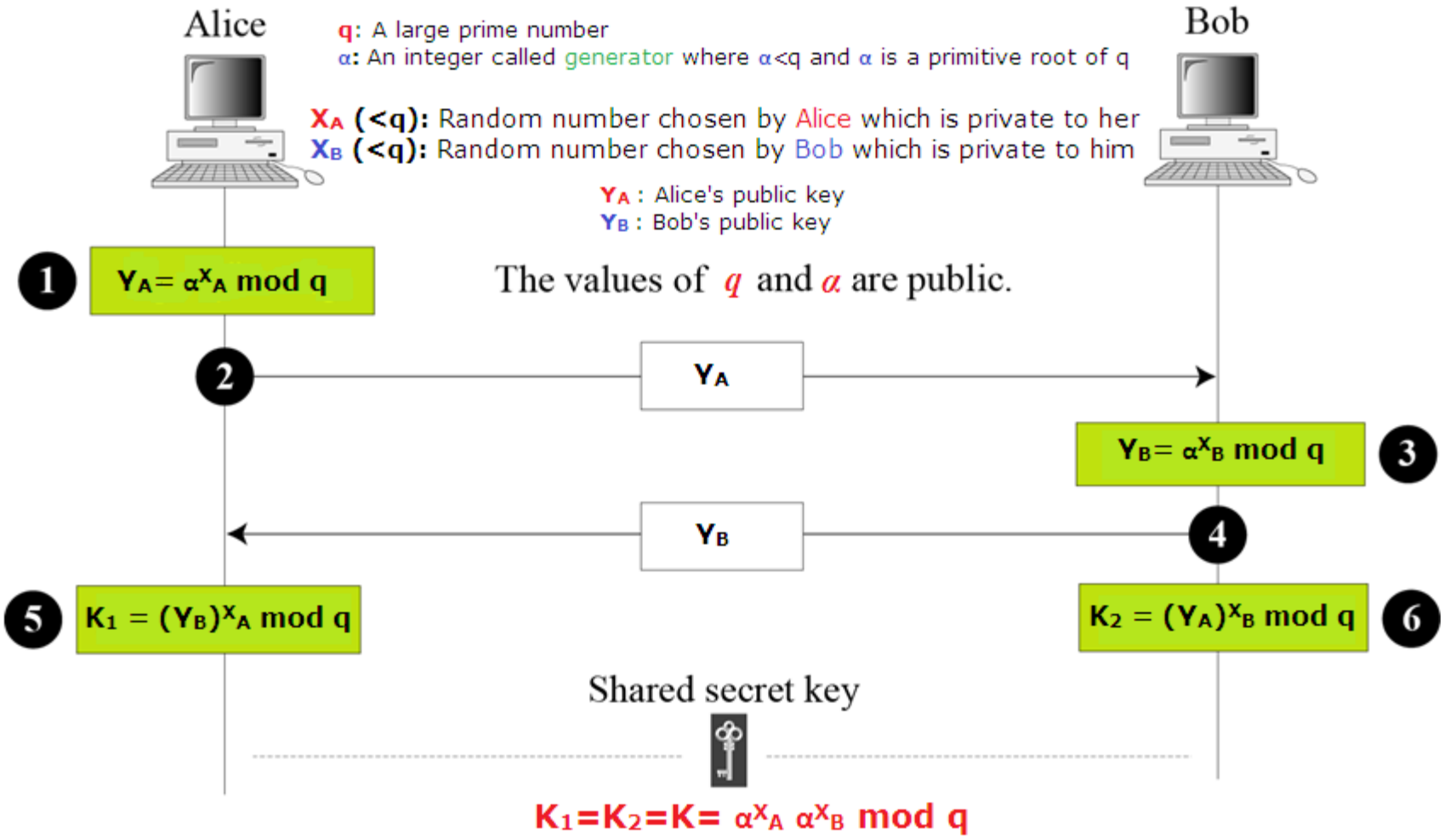

Figure: Diffie-Hellman method

Symmetric-key Agreement: Diffie-Hellman Key Agreement

- The steps of the Diffie-Hellman key exchange algorithm for this case is summarized below:

1. Alice selects a random integer X_A which is private to her where $X_A<q$. She then computes her public key $Y_A= \alpha^{X_A} \bmod q$.
2. Alice sends her public key Y_A to Bob.
3. Bob independently selects a random integer X_B which is private to him where $X_B<q$. He then computes his public key $Y_B= \alpha^{X_B} \bmod q$.
4. Bob sends his public key Y_B to Alice.
5. Using Bob's public key Y_B, Alice computes the secret key as $K_1 = (Y_B)^{X_A} \bmod q$.
6. Similarly, using Alice's public key Y_A, Bob computes the secret key as $K_2 = (Y_A)^{X_B} \bmod q$. These two calculations produce identical results which is proved below:

$K_1=(Y_B)^{X_A} \bmod q$	$K_2=(Y_A)^{X_B} \bmod q$
$=(\alpha^{X_B} \bmod q)^{X_A} \bmod q$ (Since $Y_B= \alpha^{X_B} \bmod q$)	$=(\alpha^{X_A} \bmod q)^{X_B} \bmod q$ (Since $Y_A= \alpha^{X_A} \bmod q$)
$=(\alpha^{X_B})^{X_A} \bmod q$ (by the rules of modular arithmetic)	$=(\alpha^{X_A})^{X_B} \bmod q$ (by the rules of modular arithmetic)
$=\alpha^{X_B X_A} \bmod q$	$=\alpha^{X_A X_B} \bmod q$
$=\alpha^{X_A X_B} \bmod q = K$	$=\alpha^{X_A X_B} \bmod q = K$

- In other words, the secret key K of both the users are identical and can be calculated from each other's public key and private random number X_A and X_B.
- Now Alice can send a message encrypted with her own copy of key K. Bob can decrypt the message using his own copy of key K.
- The result is that the two sides have exchanged a secret value. Furthermore, because X_A and X_B are private, an adversary only has the following ingredients to work with to determine the key, which is nearly impossible for large prime number: q, α, Y_A and Y_B.

Symmetric-key Agreement: Diffie-Hellman Key Agreement

- The Diffie-Hellman key exchange algorithm is summarized in the figure below.

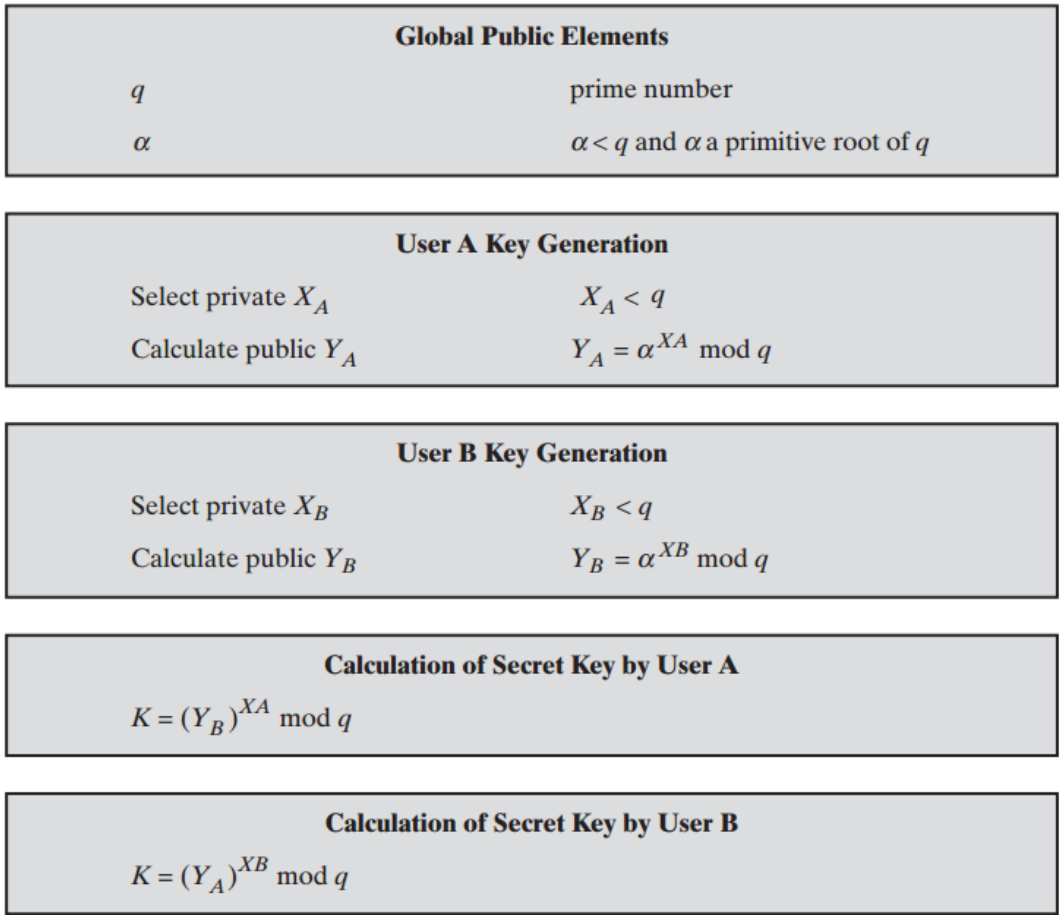

Figure: Diffie-Hellman key exchange algorithm

Note:

- The security of the Diffie-Hellman key exchange lies in the fact that, while it is relatively easy to calculate exponentials modulo a prime, it is very difficult to calculate discrete logarithms. For large primes, the latter task is considered infeasible.

Example:

- Here is a very trivial example to clearly show the idea behind Diffie-Hellman protocol. This example uses small numbers, but in a real situation, the numbers are very large.

- Assume that q = 23 and α = 7. The steps are as follows:

1. Alice chooses X_A = 3 which is private to her. She then calculates $Y_A = \alpha^{X_A} \bmod q = 7^3 \bmod 23 = 21$ which is her public key.
2. Bob chooses X_B = 6 which is private to him. He then and calculates $Y_B = \alpha^{X_B} \bmod q = 7^6 \bmod 23 = 4$ which is his public key.
3. Alice sends her public key 21 to Bob.
4. Bob sends his public key 4 to Alice.
5. Using Bob's public key Y_B=4, Alice computes the secret key as $K = (Y_B)^{X_A} \bmod q = 4^3 \bmod 23 = 18$.
6. Using Alice's public key Y_A=21, Bob computes the secret key as $K = (Y_A)^{X_B} \bmod q = 21^3 \bmod 23 = 18$.
7. The value of K is the same for both Alice and Bob which is the symmetric shared key in the Diffie-Hellman protocol.

Illustration of Diffie-Hellman Key Exchange Protocols:

- Figure below shows a simple protocol that makes use of the Diffie-Hellman calculation.
- Suppose that **user A** wishes to set up a connection with **user B** and use a secret key to encrypt messages on that connection.
- **User A** can generate a one-time private key X_A, calculate Y_A, and send that to **user B**.
- **User B** responds by generating a private value X_B, calculating Y_B, and sending Y_B to **user A**.
- Both users can now calculate the secret key K. The necessary public values q and α would need to be known ahead of time. Alternatively, **user A** could pick values for q and α and include those in the first message.

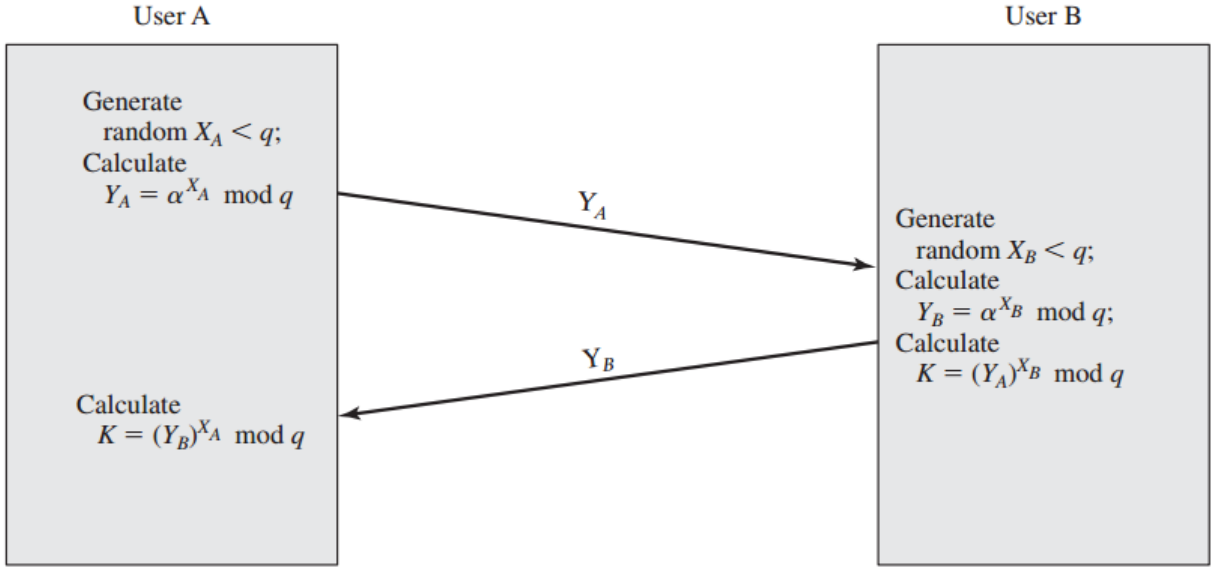

Figure: Illustration of Diffie-Hellman key exchange protocol

Security of Diffie-Hellman Key Exchange

- The Diffie-Hellman key exchange is susceptible to two attacks:

1. Discrete logarithm attack
2. Man-in-the-middle attack

Discrete Logarithm Attack:

- The security of the key exchange is based on the difficulty of the discrete logarithm problem. Eve can intercept Y_A and Y_B. If she can find X_A from $Y_A = \alpha^{X_A} \bmod q$ and X_B from $Y_B = \alpha^{X_B} \bmod q$, then she can calculate the symmetric key K. The secret key is not secret anymore.

- To make Diffie-Hellman safe from the discrete logarithm attack, the following are recommended:

1. The prime q must be very large (more than 300 decimal digits).
2. The prime q must be chosen such that q-1 has at least one large prime factor (more than 60 decimal digits).
3. The generator α must be a primitive root of the prime number q where $\alpha < q$. That is, the numbers $\alpha \bmod q$, $\alpha^2 \bmod q$, $\alpha^3 \bmod q$, …., $\alpha^{q-1} \bmod q$ are distinct and consist of the integers ranging from 1 through q-1 in some permutation.
4. Bob and Alice must destroy X_A and X_B after they have calculated the symmetric key. The values of X_A and X_B must be used only once.

Security of Diffie-Hellman Key Exchange

Man-in-the-Middle Attack:

- The Diffie-Hellman key exchange protocol has another weakness.
- Eve does not have to find the value of X_A and X_B to attack the protocol.
- She can fool Alice and Bob by creating two keys: one between herself and Alice, and another between herself and Bob.
- Figure below shows the situation.

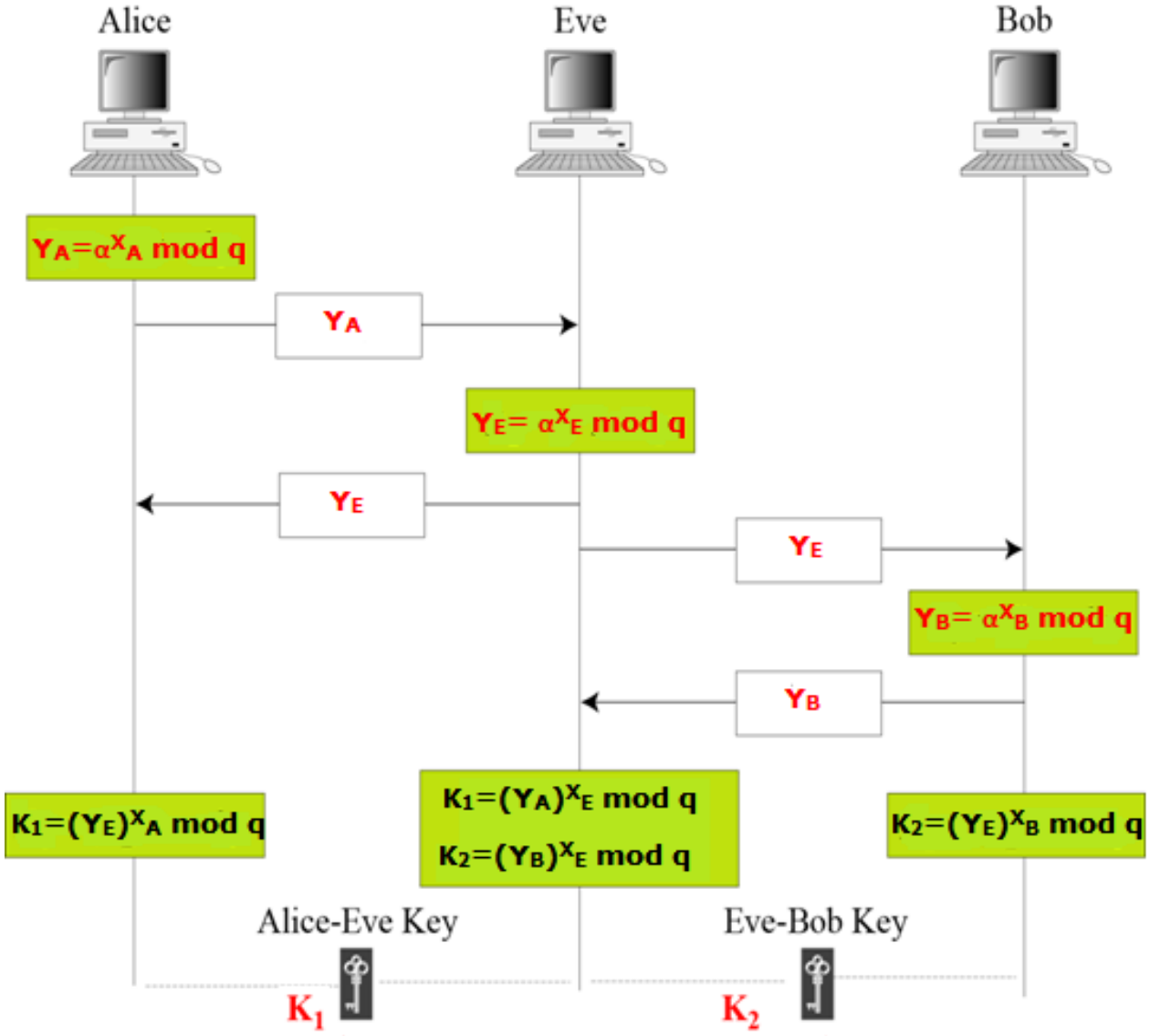

Figure: Man-in-the-middle attack

Security of Diffie-Hellman Key Exchange

Man-in-the-Middle Attack (cont...)

The following can happen:

1. Alice chooses X_A as her private key and calculates $Y_A = \alpha^{X_A} \bmod q$ as her public key, and sends Y_A to Bob.
2. Eve, the intruder, intercepts Y_A. She chooses X_E as her own key, calculates $Y_E = \alpha^{X_E} \bmod q$ as the public key, and sends Y_E to both Alice and Bob.
3. Bob chooses X_B as his private key and calculates $Y_B = \alpha^{X_B} \bmod q$, and sends Y_B to Alice. Y_B is intercepted by Eve and never reaches Alice.
4. Alice and Eve calculate K_1 which becomes a shared key between Alice and Eve. Alice, however, thinks that it is a key shared between Bob and herself.
5. Eve and Bob calculate K_2 which becomes a shared key between Eve and Bob. Bob, however, thinks that it is a key shared between Alice and himself.

- In other words, two keys, instead of one, are created: one between Alice and Eve, one between Eve and Bob.
- When Alice sends data to Bob encrypted with K_1 (shared by Alice and Eve), it can be deciphered and read by Eve. Eve can send the message to Bob encrypted by K_2 (shared key between Eve and Bob); or she can even change the message or send a totally new message. Bob is fooled into believing that the message has come from Alice. A similar scenario can happen to Alice in the other direction.
- This situation is called a man-in-the-middle attack because Eve comes in between and intercepts Y_A sent by Alice to Bob, and Y_B sent by Bob to Alice. It is also known as a bucket brigade attack because it resembles a short line of volunteers passing a bucket of water from person to person.

Symmetric-key Agreement: Station-to-Station Key Agreement

- The station-to-station protocol is a method of key agreement which is based on Diffie-Hellman.
- It uses digital signatures with public-key certificates to establish a session key between Alice and Bob, as shown in the figure here.

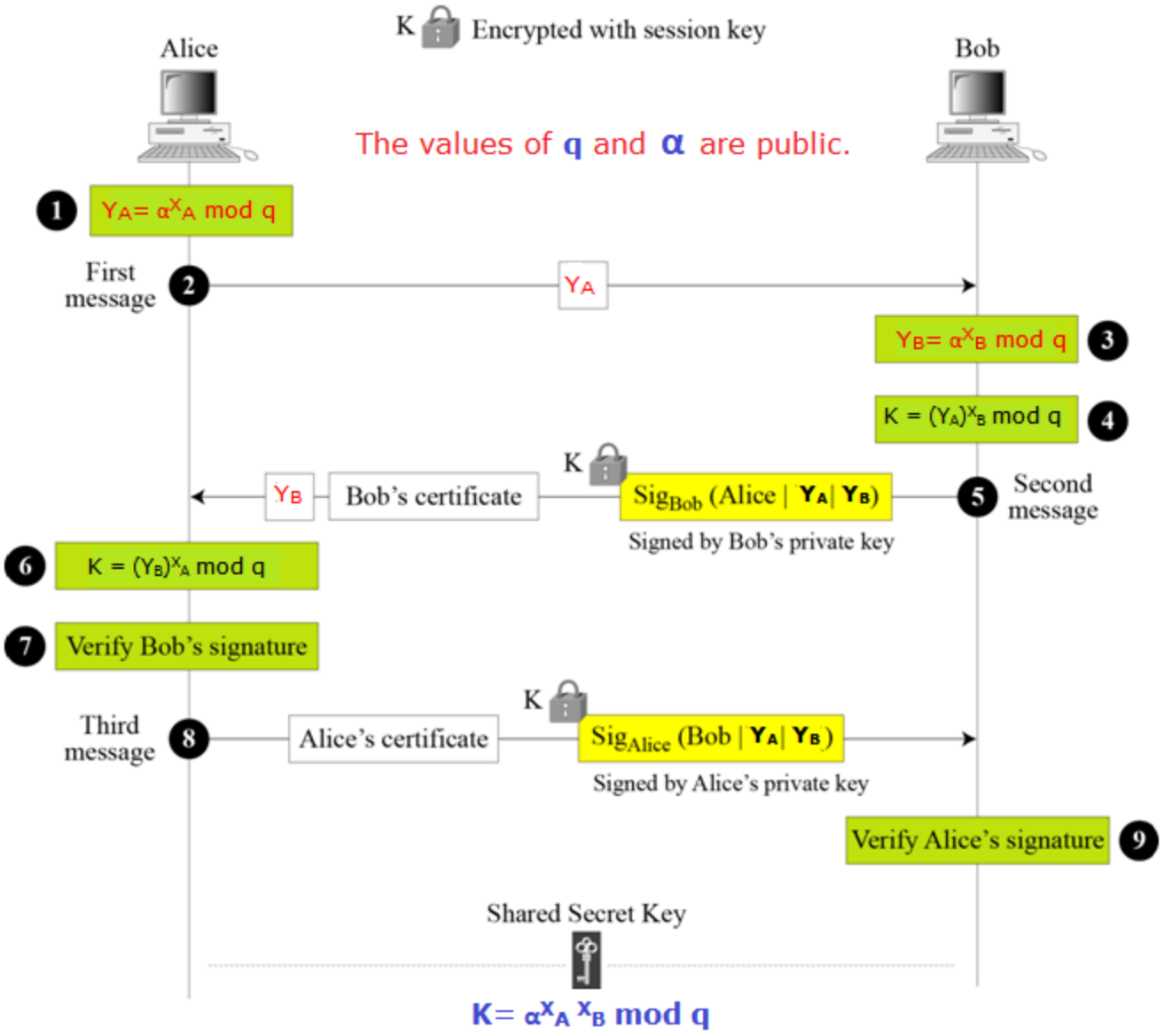

Figure: Station-to-station key agreement method

Symmetric-key Agreement: Station-to-Station Key Agreement

- The steps of the station-to-station key agreement protocol is summarized below:
  - After calculating Y_A, Alice sends it to Bob (steps 1 and 2 in the above figure).
  - After calculating Y_B and the session key, Bob concatenates Alice's ID, Y_A , and Y_B. He then signs the result with his private key. Bob now sends Y_B, the signature, and his own public-key certificate to Alice. The signature is encrypted with the session key (steps 3, 4, and 5 in the above figure).
  - After calculating the session key, if Bob's signature is verified, Alice concatenates Bob's ID, Y_A, and Y_B. She then signs the result with her own private key and sends it to Bob. The signature is encrypted with the session key (steps 6, 7, and 8 in the above figure).
  - If Alice's signature is verified, Bob keeps the session key (step 9 in the figure).

Security of Station-to-Station Protocol:

- The station-to-station protocol prevents man-in-the-middle attacks.
  - After intercepting Y_A, Eve cannot send her own Y_B to Alice and pretend it is coming from Bob, because Eve cannot forge the private key of Bob to create the signature—the signature cannot be verified with Bob's public key defined in the certificate.
  - In the same way, Eve cannot forge Alice's private key to sign the third message sent by Alice.
  - The certificates are trusted because they are issued by trusted authorities.

Public-key Distribution

- In asymmetric-key cryptography, people do not need to know a symmetric shared key; everyone shields a private key and advertises a public 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.
  - Similarly, 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, everyone shields a private key and advertises a public key.
- Like secret keys, **public keys need to be distributed.**
- Now we will briefly discuss the way public keys can be distributed.

Ways of Distributing Public Key:

- There are several approaches to distribute a public key:
  1. Public announcement
  2. Trusted center
  3. Certifying authority

Public Announcement:

- The naive approach is to announce public keys publicly.

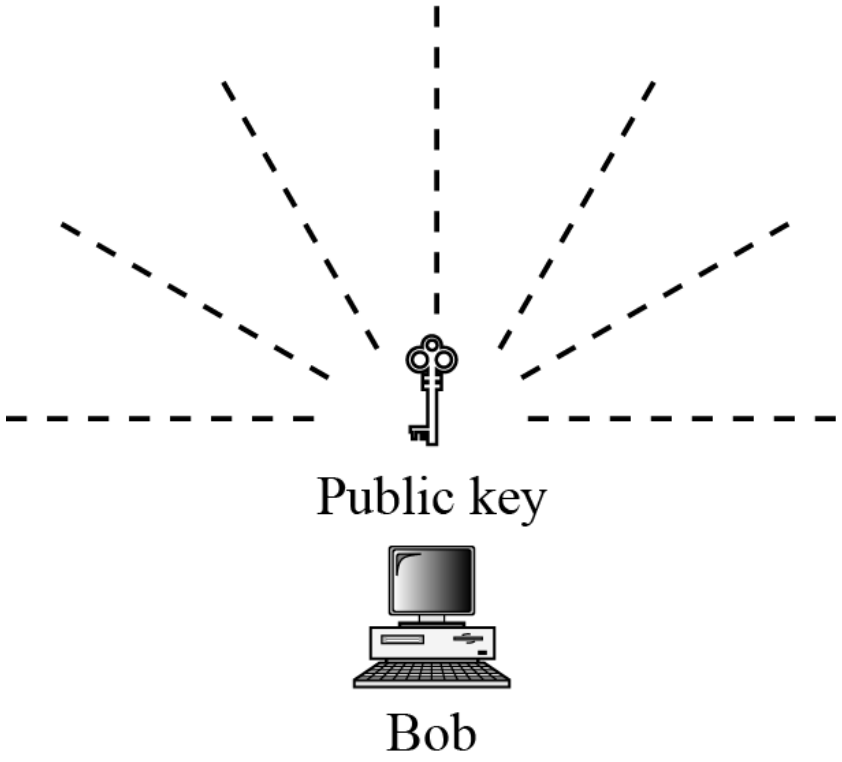

Figure: Announcing a public key

- 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.
- Figure below shows the situation.
- However, **this approach 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 can fool Alice into sending her a message that is intended for Bob.
  - 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.

Ways of Distributing Public Key:

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.
  - Figure shows the concept.

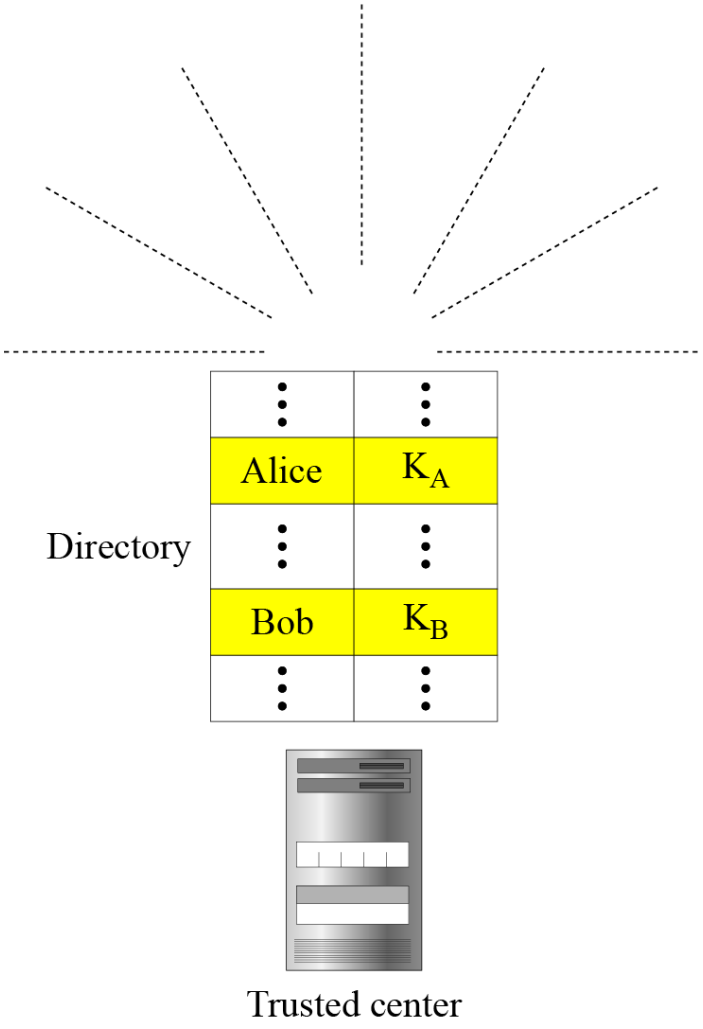

Figure: Trusted center

Ways of Distributing 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.
  - Figure shows one scenario.

Directory

Alice	K_A
Bob	K_B

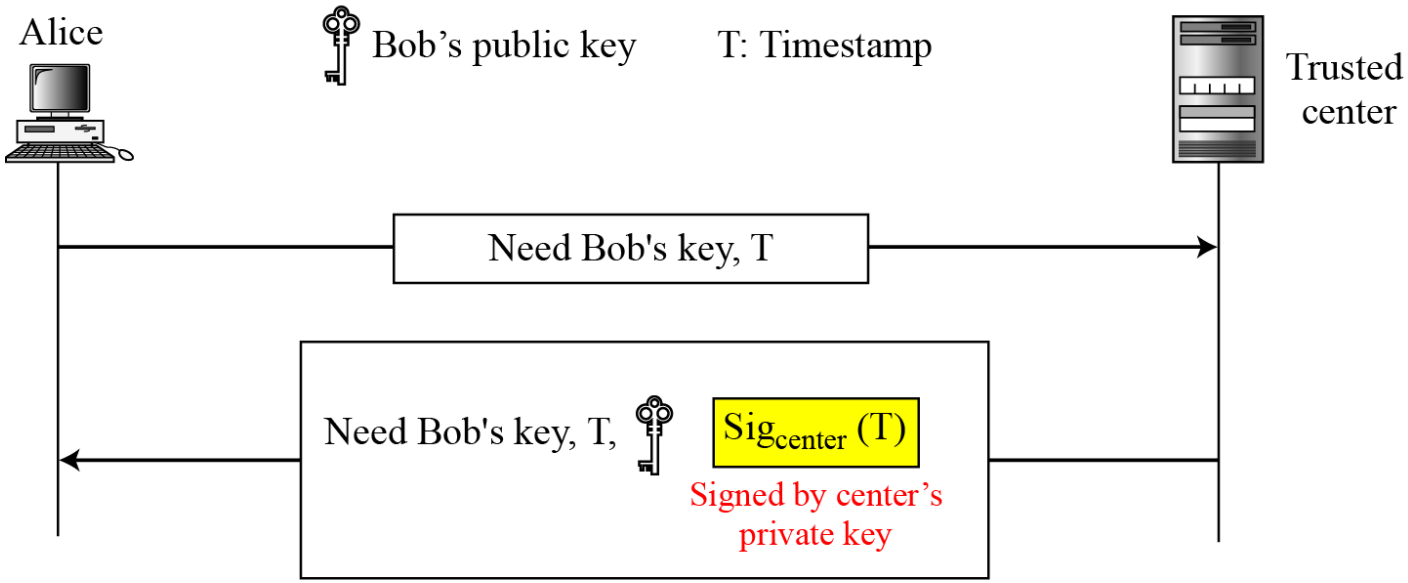

Figure: Controlled trusted center

Ways of Distributing Public Key:

Certification Authority (CA):

- The distribution of public key through controlled trusted center (discussed before) can create a heavy load on the center if the number of requests is large. The alternative is to create **public-key certificates.**
- Suppose, Bob wants two things:
  1. He wants people to know his public key.
  2. He wants no one to accept a forged public key as his.
     - How can he do this?
- The steps he may follow are:
  - 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 (using a picture ID along with other proof).
  - It then asks for Bob's public key and writes it on the certificate it will issue for Bob.
    - 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 CA's public key to extract Bob's public key.

Ways of Distributing Public Key:

Certification Authority (Cont...):

- The steps stated before are illustrated in the figure below.

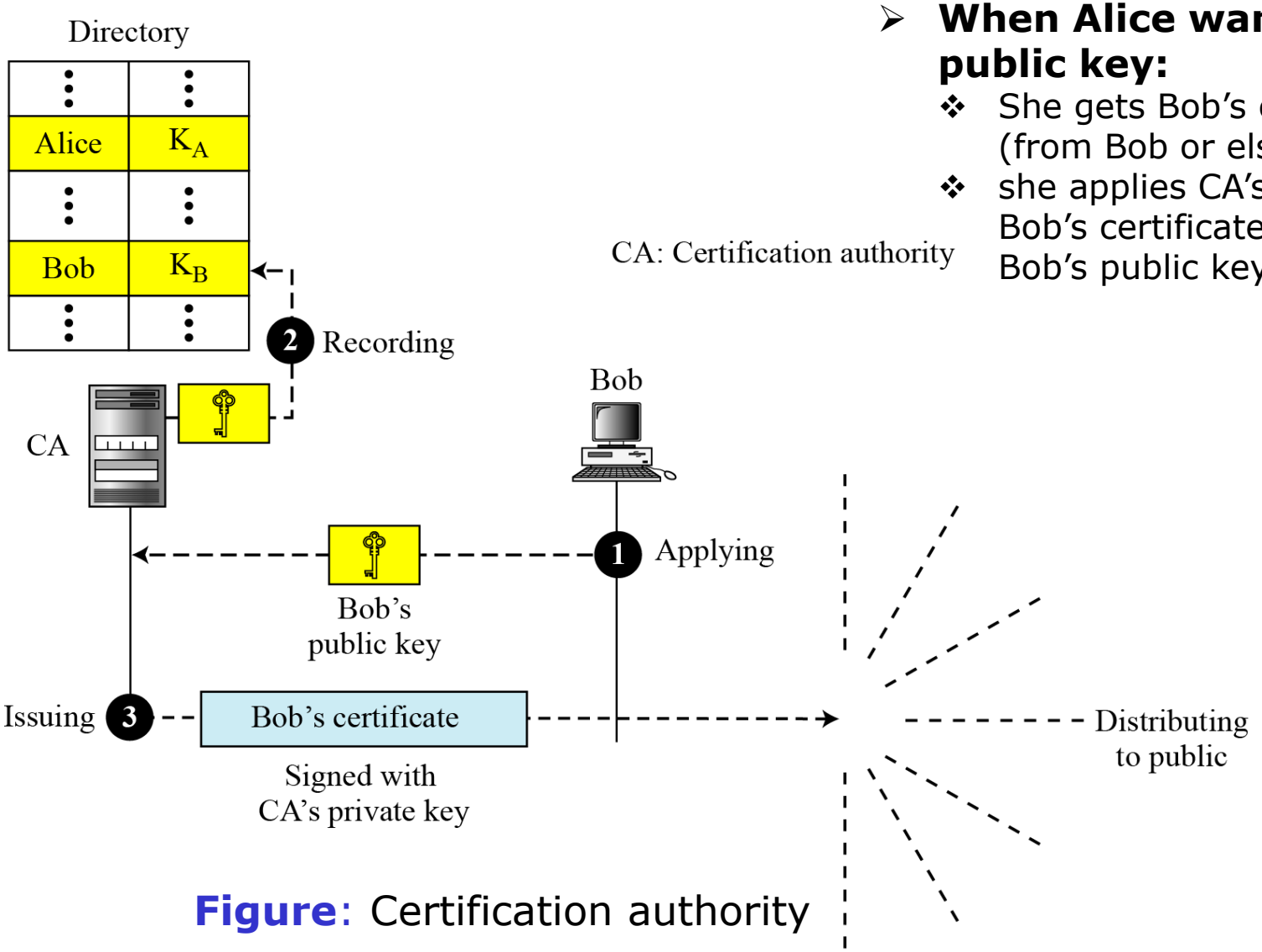

Figure: Certification authority

- **When Alice wants Bob's public key:**
  - She gets Bob's certificate (from Bob or elsewhere).
  - she applies CA's public key to Bob's certificate, and gets Bob's public key

Digital Certificate

What is a digital certificate?

- A problem in public-key systems is the authenticity of the public key.
  - An attacker may offer the sender her own public key and pretend that it origins from the legitimate receiver.
  - The sender then uses the fake public key to perform her encryption and the attacker can simply decrypt the message using her private key.
  - This technique may be used to set up a man-in-the-middle attack in which a third party is able to monitor and modify the communication between two parties, even when encryption is used.
- In order to thwart an attacker that attempts to substitute her public key for the victim's one, digital certificates are used.
  - A certificate combines user information with the user's public key and the digital signature of a trusted third party that guarantees that the key belongs to the mentioned person.
  - The trusted third party is usually called a certification authority (CA).
- A digital certificate is just a file or a software program, digitally signed by a signing authority, that can be installed in a browser. Once installed, the digital certificate identifies the user of that browser to websites equipped to check it automatically. It is like an electronic "credit card" that establishes one's credentials when doing business on the Web.

Digital Certificate

What is a digital certificate?

- Therefore, a digital certificate, issued by a certifying authority, is an electronic attachment to an electronic message that is used to verify that a user sending a message is who they claim to be. The certificate provides the receiver of the message with the means to encode a reply. Digital certificate also allows a user to send an encrypted message.
- Those wishing to send encrypted messages obtain a digital certificate from a certifying authority.
  - The certifying authority issues an encrypted digital certificate.
- The recipient of an encrypted message uses the certifying authority's public key to decode the digital certificate attached to the message.
  - The recipient verifies it as issued by the certifying authority.
  - Then it obtains the sender's public key and identification information held within the digital certificate.
  - With this information, the recipient can then send an encrypted reply.
- The most widely used standard for digital certificates is X.509. Hence, digital certificates are sometimes called X.509 certificates.

Digital Certificate

What information a digital certificate carries?

- A digital certificate may contain information such as certificate holder's email address, globally accessible name, mailing address, birth-date, gender, SSN, passport number, company name, website's URL, and other information including public key for cryptographic use, name of the certificate authority, issue and expiry date (i.e. the duration of the certificate), the class of the certificate, and the certificate's ID number.
- The certificate holder may be required to provide additional information such as Business License and Certificate of Business Registration which are kept on file at the signing authority. A digital certificate does not contain holder's private key. The signing authority guarantees that the information is true. They use their private key to sign the certificate attesting to its authenticity.
- The contents of a digital certificate are outlined below:

1.	Basic identity of user/owner (name, e-mail address, postal-mail address, SSN, etc)
2.	Digital signature and ID information of issuing authority.
3.	User's/owner's public key, Certifying authority's public key.
4.	Dates of validity and expiration of the certificate
5.	Version or class of certification (i.e. class 1, class 2, etc)
6.	Certificate's ID number/ Serial number (an integer assigned by the issuer)

Digital Certificate

For what purposes you can use the digital certificates?

- You can use the digital certificate to digitally sign email, documents, files etc. to prove you were the author, and that they have not been tampered with.
- You can also use some types of certificate as digital ID. Others can electronically challenge you to prove you know the private key that fits with the public key in the certificate by encrypting a message they provide.
  - The problem with that is, all the information in the certificate is revealed to whoever you show it to.
  - If you want to selectively reveal information, you need several certificates.
    - You might want one with just your birth date for entry to porn sites, but no other information. You might want one that revealed only a very minimal amount of information when dealing with on-line vendors to avoid being bombarded with junk electronic and snail mail.
- Digital certificates can also be used instead of passwords to verify who you are to some site.
  - The site challenges you by sending you a message that you digitally sign and send back. If some spy had snooped on you logging in before, it would not help him to spoof you, the way it would had you used a password.
  - Thus, a digital certificate eliminates remembering multiple passwords and enhances security, because it can not be guessed, forgotten, forged, or intercepted.

Digital Certificate

For what purposes you can use the digital certificates (cont…)?

- Other types of certificate allow you to encrypt and sign all HTML traffic leaving your web server, thus proving it came from you and providing privacy.
  - Recipients can determine whether data did indeed come from you by checking the digital signature.
  - To verify, all they need is a master certificate from the signing authority, which comes built into their browser or email software. They don't need to check up your key in an on-line database unless they want to check to see if the certificate has been revoked.
- In many ways, digital certificates are the heart of secure online transactions.
  - In shopping on the Internet, buyers need evidence that they can trust the vendor. Digital certificate establishes a merchant's identity and thus ensures secure e-commerce transaction.
  - MasterCard and Visa have designed the SET certificate that can be used for secure financial transactions over the web. VeriSign supplies the certificates.

Digital Certificate

Different Classes of Digital Certificate:

- A digital certificate can be issued (for a fee) in one of FOUR classes:

1. Class 1 Certificate:

- Certificates of this class are the quickest and simplest to issue because they contain minimum checks on the user's background. Only the name, address and e-mail address of the user are checked. Think of it as a library card.

2. Class 2 Certificate:

- Certificates of this class check for information like real name, SSN (social security number), and date of birth of the user. They require proof of physical address, locale, and e-mail address as well. This is more like a credit card, because the company giving out the certificate will consult with a credit database for verification with a third party.

3. Class 3 Certificate:

- Certificates of this class are the strongest type in terms of specifics. They are like a driver's license: To get them, you need to prove exactly who you are and that you are responsible. Organizations whose specialty is the security business foresee class 3 certificates being used for things like loans acquired online and other sensitive transactions.

4. Class 4 Certificate:

- Certificates of this class are the most thorough. In addition to class 3 requirements, the certificate authority checks on things like the user's position at work.

Prepared by: K M Akkas Ali, Assistant Professor, IIT, JU

X.509 Digital Certificate

- 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 been accepted by the Internet with some changes.
- 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 Certificate

Format of X.509 Certificate:

- Figure below shows the format of X.509 certificate.

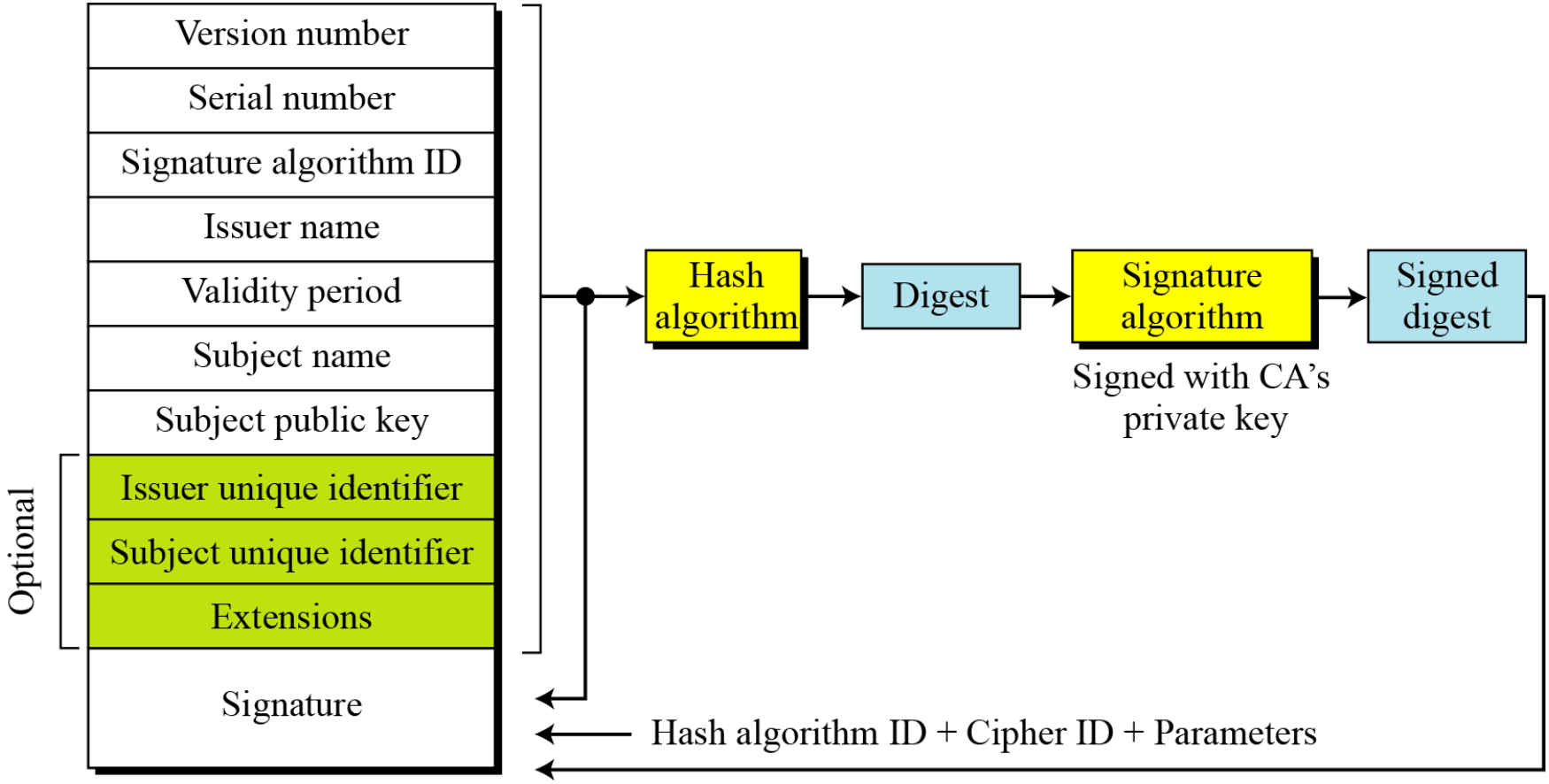

Figure: X.509 certificate format

X.509 Certificate

Format of X.509 Certificate (continue…):

- A certificate has the following fields:
  - **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.

X.509 Certificate

Format of X.509 Certificate (continue...):

- **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.

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.

X.509 Certificate

Certificate Revocation:

- In some cases a certificate must be revoked before its expiration.
- Here are some examples:
  a) The user's (subject's) private key (corresponding to the public key listed in the certificate) might have been comprised.
  b) 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.
  c) 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 Format:

- Figure below shows the certificate revocation list.

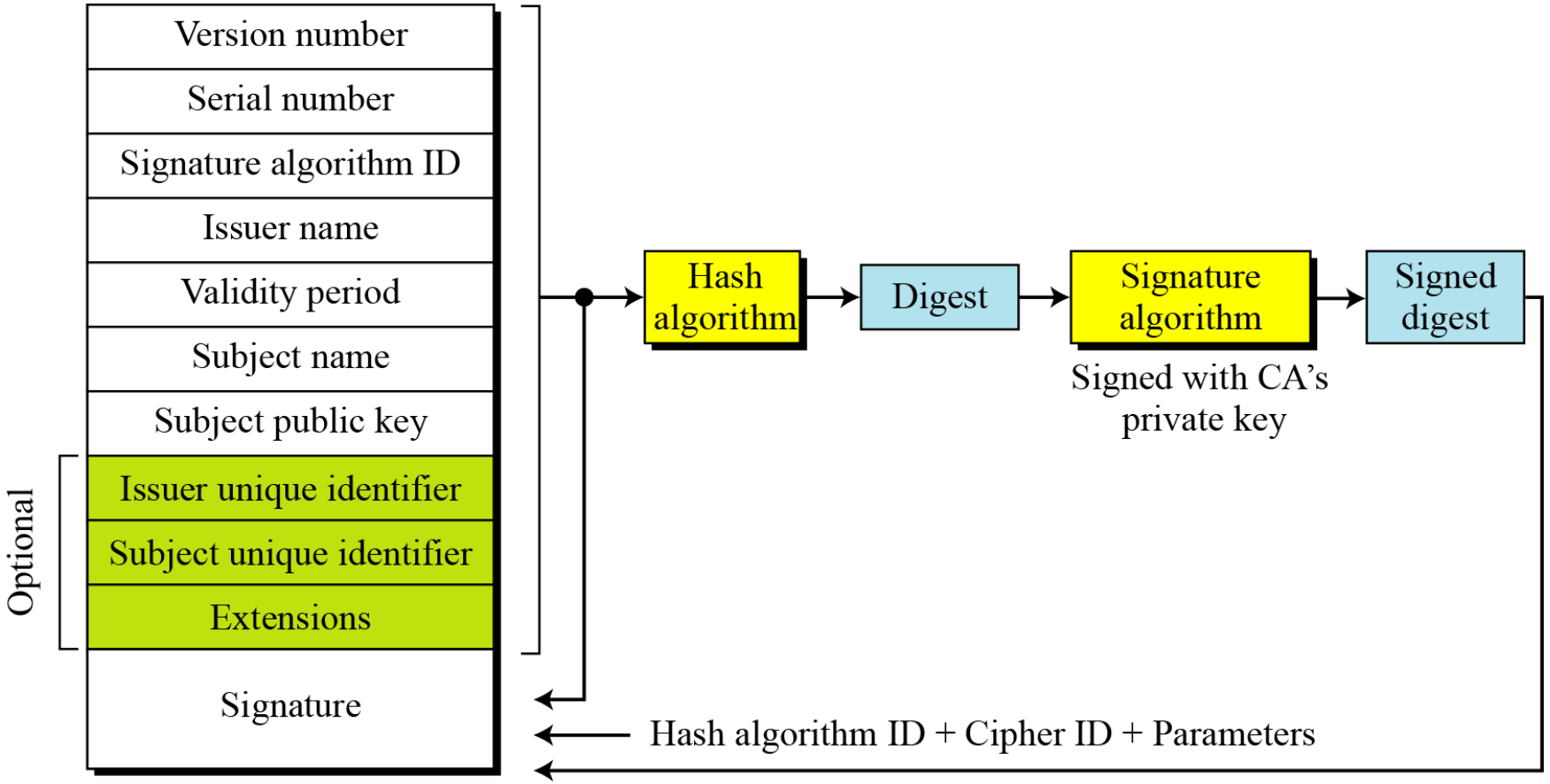

Figure: Certificate revocation format

Certificate Revocation Format (cont...):

- 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 ha 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.

Certificate Authority

Digital Certificate Vendors:

- A CA vendor (such as VeriSign, Entrust, etc) issues certificates that contain the identities and affiliations of individuals, along with their public keys.
- These certificates are bound together with the digital signature and stored in a special directory.
- The sender's browser looks up the recipient's certificate in the directory, and the message can be encrypted using the key embedded in the certificate.
- The sender can then sign the message using his own private key, and the recipient can verify the signature by using the sender's public key that is vouched for by the CA.
- The table below lists some popular certificate authorities:

Company	Types of Certificate Sold
Actalis	Italian certificate authority. Website is only in Italian.
Certum	A Polish company offering over a dozen different types of certificate.
Cren	Institutional Certificates, e.g. entire universities. Cost is per institution based on size.
CyberTrust (USA)	Sell SSL certs for a year.
DigitCert.com	SSL cert per year.
Ebizid	SSL Cert to per year.
Entrust (USA)	personal email certificates(free), SSL Server(free), VPN (free), SET (free). Free certs are 60-days for testing only.
PGP Pretty Good Privacy	Certificate server software you install issues PGP certificates.
TC Trust Center	personal email certificates.
thawte Thawte Certification	(South Africa)
VeriSign (USA)	Verisign is the prestige company for certs. If any cert will be supported, recognized and accepted, it will be Verisign.

Certificate Authority

Selecting a Certificate Vendor:

- Some criteria to consider when buying your certificate are:
  - Cost, both initial and renewal.
  - Does the company provide all the different kinds of certificate you will need. It is much less hassle to get everything from one source.
  - Are the root certificates (*root certificates are typically pre-installed at the factory in browsers*) of that vendor built into the browsers your clients will be using? If not, it will be a hassle for your users to manually install them.
  - What sort of reputation does the vendor have for service? Basically you are paying them to verify that you are you. You want them to do that thoroughly without driving you crazy.

Public-key Infrastructure (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 of PKI:

- Several duties have been defined for a PKI. The most important ones are shown in the figure below:

- **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.

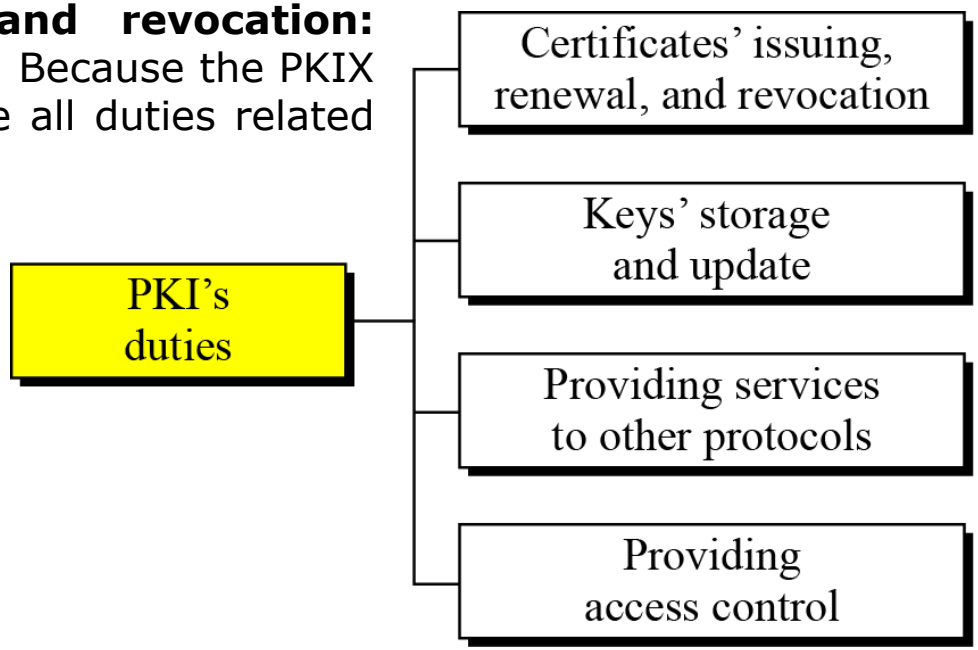

Figure: Some duties of a PKI

Duties of PKI (cont…):

- **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.

Public Key Infrastructure (PKI)

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.

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.

Hierarchical Model (cont..):

- Figure below shows a trust model of this kind with three hierarchical levels. The number of levels can be more than three in a real situation.

- The figure shows that the CA (the root) has signed certificates for CA1, CA2, and CA3; CA 1 has signed certificates for User1, User2, and User3; and so on. PKI uses X<<Y>> as the notation to mean the certificate issued by authority X for entity Y.

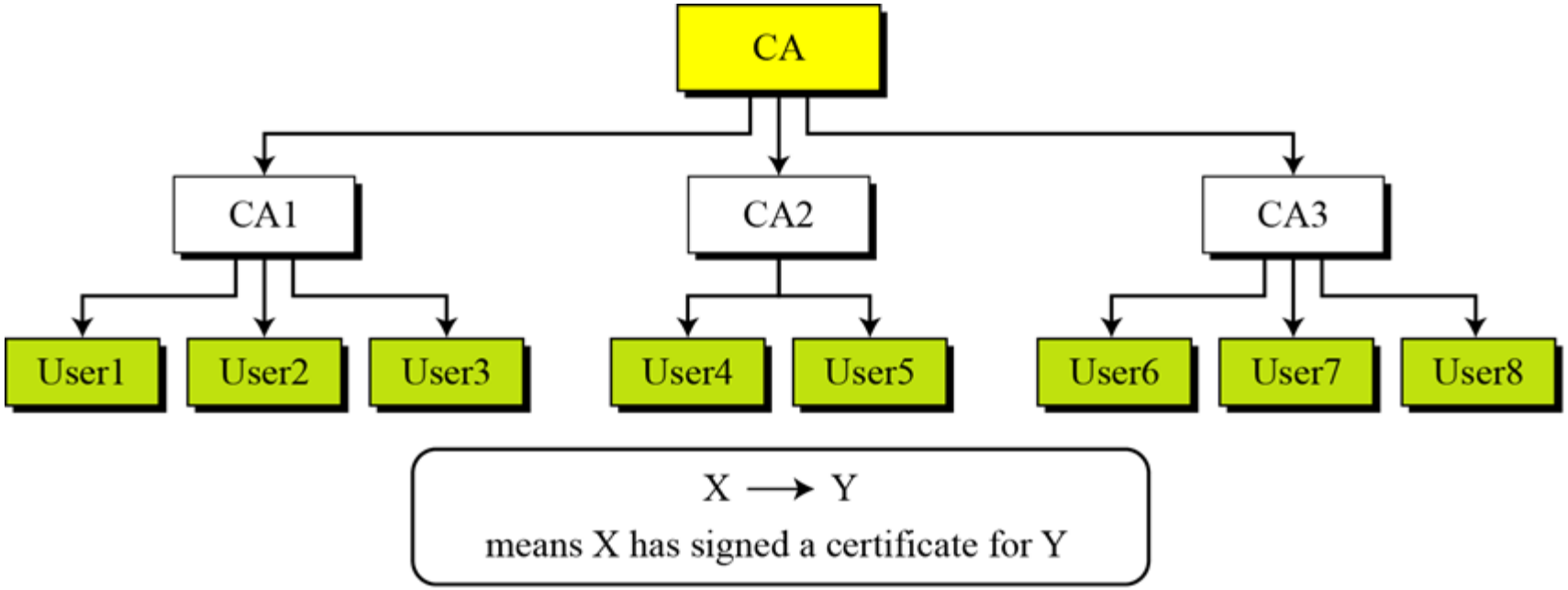

Figure: PKI hierarchical model

Public Key Infrastructure (PKI)

Example-1:

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

Solution:

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

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

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

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

d) User1 extracts the public key of User3 from CA1<<User3>>.

Example -2:

Some Web browsers, such as Netscape and Internet Explorer, 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 Internet Explorer at Tools/Internet Options/Contents/Certificate/Trusted roots (using pull-down menu). The user then can choose any of this root and view the certificate.