**PANIMALAR INSTITUTE OF TECHNOLOGY**

**DEPARTMENT OF IT**

**CCS354 NETWORK SECURITY**

# UNIT II KEY MANAGEMENT AND AUTHENTICATION

Key Management and Distribution: Symmetric Key Distribution, Distribution of Public Keys, X.509 Certificates, Public-Key Infrastructure. User Authentication: Remote User-Authentication Principles, Remote User-Authentication Using Symmetric Encryption, Kerberos Systems, Remote User Authentication Using Asymmetric Encryption.

## KEY MANAGEMET AND KEY DISTRIBUTION

## SYMMETRIC KEY DISTRIBUTION USING ASYMMETRIC ENCRYPTION

For symmetric encryption to work, the two parties to an exchange must share the same key, and that keymust be protected from access by others.

For two parties A and B, key distribution can be achieved in a number of ways, as follows:

1. A can select a key and physically deliver it to B.
2. A third party can select the key and physically deliver it to A and B.
3. If A and B have previously and recently used a key, one party can transmit the new key to the other,encrypted using the old key.
4. If A and B each has an encrypted connection to a third party C, C can deliver a key on the encryptedlinks to A and B.

### A Key Distribution Scenario

- User A wishes to establish a logical connection with B and requires a one-time session key to protect the data transmitted over the connection.
- A has a master key, $K_a$, known only to itself and the KDC; similarly, B shares the master key $K_b$ with the KDC.
- The following steps occur:

1. A issues a request to the KDC for a session key to protect a logical connection

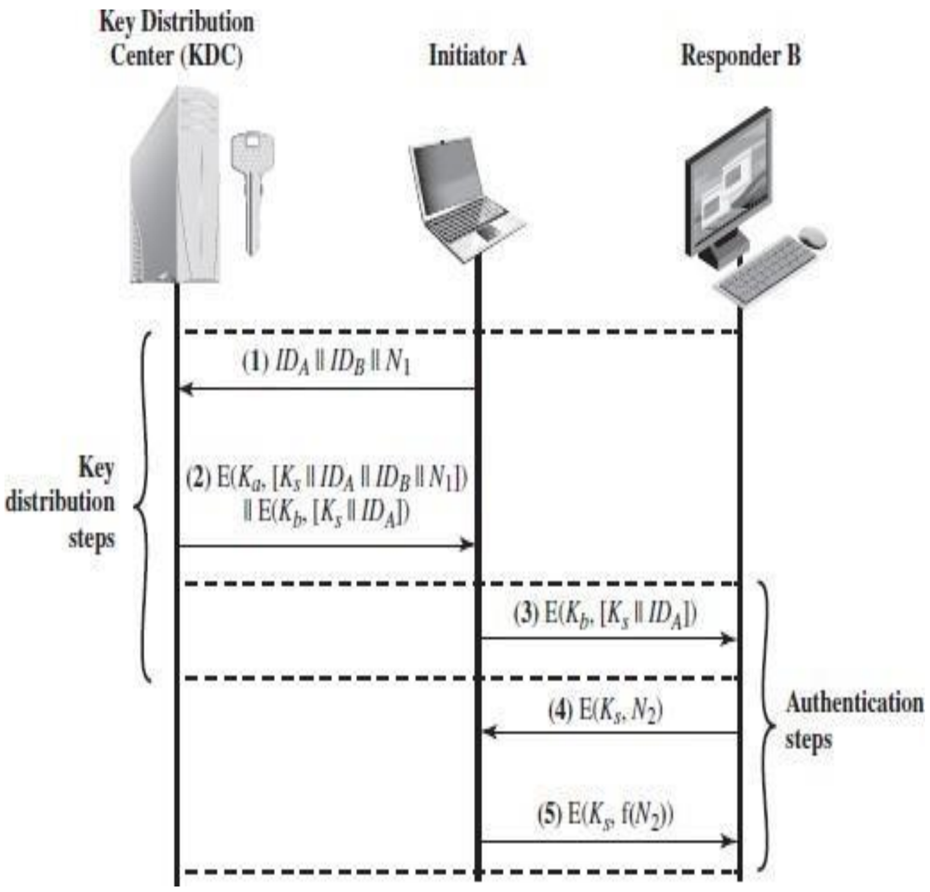
to B. The message includes the identity of A and B and a unique identifier, $N_1$, for this transaction, which we refer to as a **nonce**. The nonce may be a timestamp, a counter, or a random number.

2. The KDC responds with a message encrypted using $K_a$. Thus, A is the only one who can successfully read the message, and A knows that it originated at the KDC.

The message includes two items intended for A:

• The one-time session key, $K_s$, to be used for the session

• The original request message, including the nonce, to enable A to match this response with the
appropriate request.


Key Distribution Center (KDC)
Initiator A
Responder B
(1) $ID_A \| ID_B \| N_1$
Key distribution steps
(2) $E(K_a, [K_s \| ID_A \| ID_B \| N_1]) \| E(K_b, [K_s \| ID_A])$
(3) $E(K_b, [K_s \| ID_A])$
(4) $E(K_s, N_2)$
(5) $E(K_s, f(N_2))$
Authentication steps

3. A stores the session key for use in the upcoming session and forwards to B the information that originated at the KDC for B, namely, $E(K_b,[K_s \| ID_A])$.

4. Using the newly minted session key for encryption, B sends a nonce, $N_2$, to A.

5. Also, using Ks, A responds with f(N2), where f is a function that performs some transformation on N2

## Hierarchical Key Control

- It is not necessary to limit the key distribution function to a single KDC.
- A hierarchical scheme minimizes the effort involved in master key distribution, because most master keys are those shared by a local KDC with its local entities.

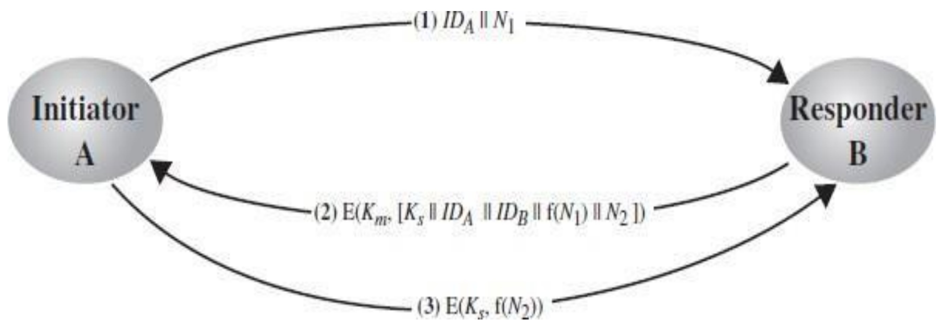
## Session Key Lifetime

The more frequently session keys are exchanged, the more secure they are, because the opponent has less ciphertext to work with for any given session key.

- For connection-oriented protocols, one choice is to use the same session key for the length of time that the connection is open, using a **new session key for each new session.**
- For a connectionless protocol, use **session key for a certain fixed period only or for a certain number of transactions.**

## Decentralized Key Control

A decentralized approach requires that each end system be able to communicate in a secure manner with all potential partner end systems for purposes of session key distribution.



A session key may be established with the following sequence of steps

1. A issues a request to B for a session key and includes a nonce, N1.
2. B responds with a message that is encrypted using the shared master key. The response includes the session key selected by B, an identifier of B, the

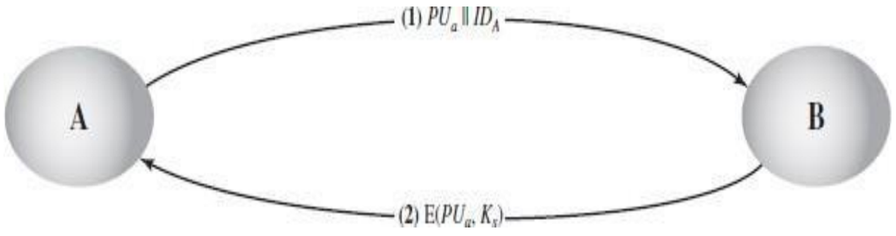
value f(N1), and another nonce, N2.

3. Using the new session key, A returns f(N2) to B.

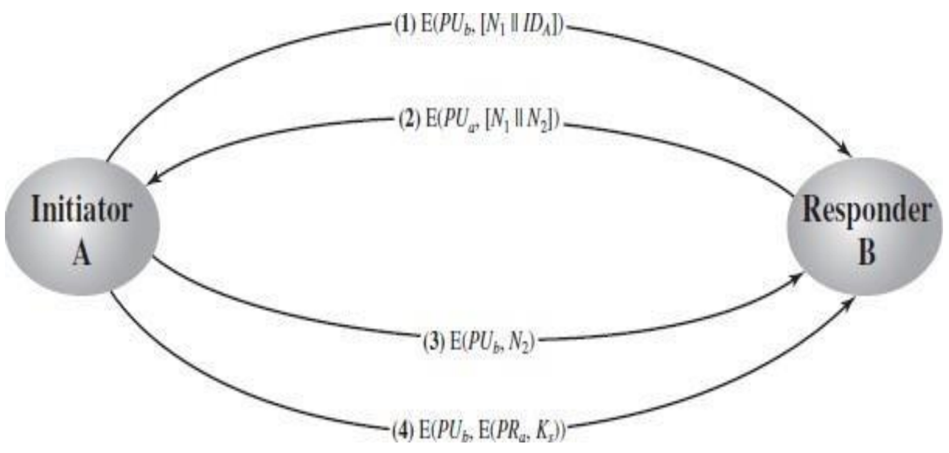
## SYMMETRIC KEY DISTRIBUTION USING ASYMMETRIC ENCRYPTION

**Simple Secret Key Distribution**

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



**Public-Key Distribution of Secret Keys**


(1) $E(PU_b, [N_1 \| ID_A])$
(2) $E(PU_a, [N_1 \| N_2])$
Initiator A
Responder B
(3) $E(PU_b, N_2)$
(4) $E(PU_b, E(PR_a, K_s))$

1. A uses B's public key to encrypt a message to B containing an identifier of A(IDA) and a nonce (N1), which is used to identify this transaction uniquely.
2. B sends a message to A encrypted with PUa and containing A's nonce (N1)

as well as a new nonce generated by B ($N_2$). Because only B could have decrypted message (1), the presence of $N_1$ in message (2) assures A that the correspondent is B.

3. A returns $N_2$, encrypted using B’s public key, to assure B that its correspondent is A.
4. A selects a secret key $K_s$ and sends $M = E(PU_b, E(PR_a, K_s))$ to B. Encryption of this message with B’s public key ensures that only B can read it; encryption with A’s private key ensures that only A could havesent it.
5. B computes $D(PU_a, D(PR_b, M))$ to recover the secret key.

## DISTRIBUTION OF PUBLIC KEYS

- Public announcement
- Publicly available directory
- Public-key authority
- Public-key certificates

### Public Announcement of Public Keys

A
$PU_a$
$PU_a$
$PU_a$
$PU_a$
B
$PU_b$
$PU_b$
$PU_b$
$PU_b$

- Any participant can send his or her public key to any other participant or broadcast the key to the community at large.
- Although this approach is convenient, it has a major weakness. Anyone can forge such a public announcement.
- That is, some user could pretend to be user A and send a public key to another participant or broadcast such a public key. Until such time as user A discovers the forgery and alerts other participants, the forger is able to read all encrypted messages intended for A and can use the forged keys for authentication

**Publicly Available Directory**

1. The authority maintains a directory with a {name, public key} entry for each participant.
2. Each participant registers a public key with the directory authority. Registration would have to be in person or by some form of secure authenticated communication.
3. A participant may replace the existing key with a new one at any time, either because of the desire to replace a public key that has already been used for a large amount of data, or because the corresponding private key has been compromised in some way.
4. Participants could also access the directory electronically. For this purpose, secure, authenticated communication from the authority to the participant is mandatory.

**Public-Key Authority**

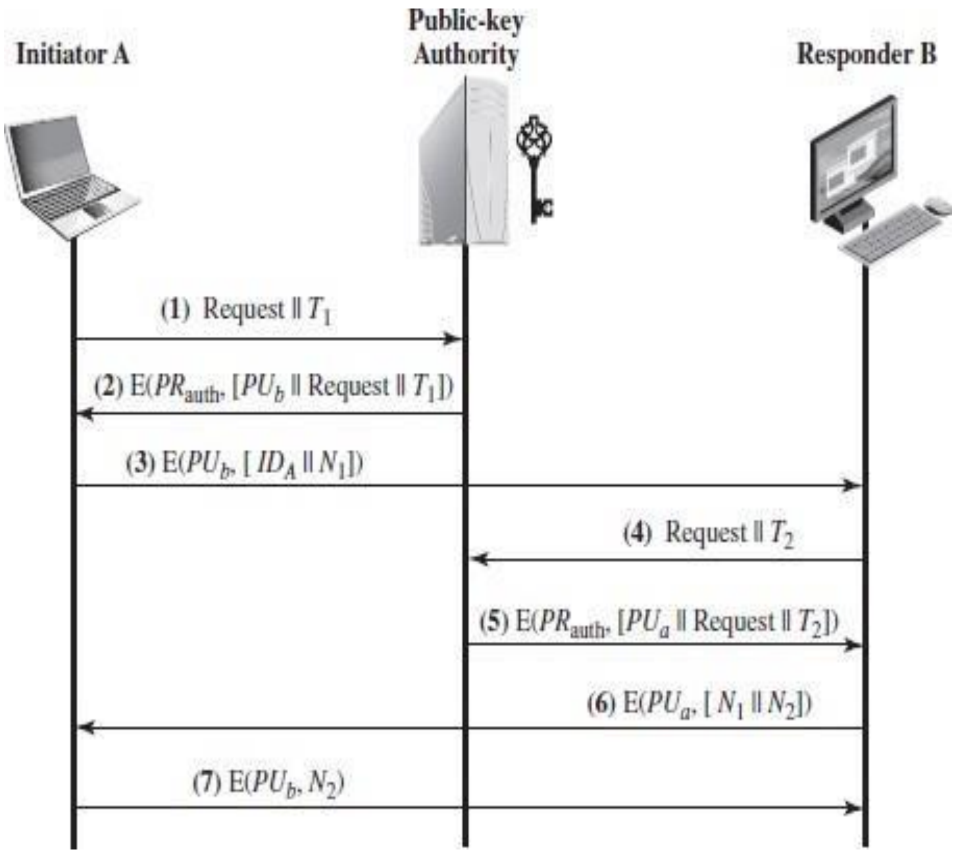
1. A sends a timestamped message to the public-key authority containing a Request for the current public key of B.

2. The authority responds with a message that is encrypted using the authority's private key, $PR_{auth}$. Thus, A is able to decrypt the message using the authority's public key. Therefore, A is assured that the message originated with the authority. The message includes the following:

• B's public key, $PU_b$, which A can use to encrypt messages destined for B

• The original request used to enable A to match this response with the corresponding earlier request and toverify that the original request was not altered before reception by the authority

• The original timestamp given so A can determine that this is not an old message from the authority containing a key other than B's current public key.



Initiator A
Public-key Authority
Responder B
(1) Request ‖ $T_1$
(2) E($PR_{auth}$, [$PU_b$ ‖ Request ‖ $T_1$])
(3) E($PU_b$, [ $ID_A$ ‖ $N_1$])
(4) Request ‖ $T_2$
(5) E($PR_{auth}$, [$PU_a$ ‖ Request ‖ $T_2$])
(6) E($PU_a$, [ $N_1$ ‖ $N_2$])
(7) E($PU_b$, $N_2$)

3. A stores B’s public key and also uses it to encrypt a message to B containing an identifier of A (IDA) anda nonce (N1), which is used to identify this transaction uniquely.

**4, 5.** B retrieves A’s public key from the authority in the same manner as A Retrieved B’s public key.

At this point, public keys have been securely delivered to A and B, and they may begin their protected exchange. However, two additional steps are desirable:

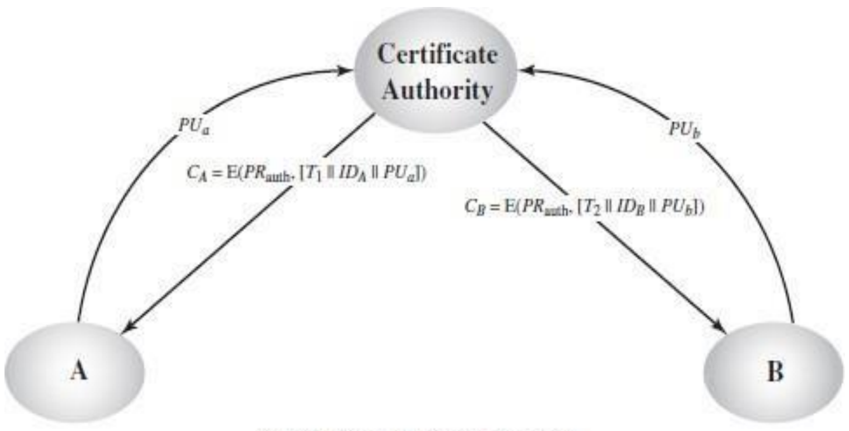
6. B sends a message to A encrypted with PUa and containing A’s nonce (N1) as well as a new nonce generated by B (N2). Because only B could have decrypted message (3), the presence of N1 in message (6) assures A that the correspondent is B.

7. A returns N2, which is encrypted using B’s public key, to assure B that its Correspondent is A.

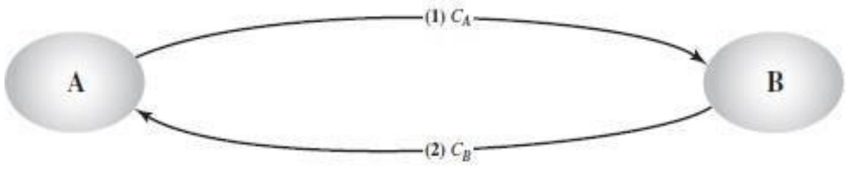
**Public-Key Certificates**

1. Any participant can read a certificate to determine the name and public key of the certificate's owner.
2. Any participant can verify that the certificate originated from the certificate authority and is notcounterfeit.
3. Only the certificate authority can create and update certificates.
4. Any participant can verify the currency of the certificate.

For participant A, the authority provides a certificate of the form$C_A = E(PR_{auth}, [T \| ID_A \| PU_a])$ where $PR_{auth}$ is the private key used by the authority and T is a timestamp. $D(PU_{auth}, C_A) = D(PU_{auth}, E(PR_{auth}, [T \| ID_A \| PU_a])) = (T \| ID_A$


Certificate Authority
$PU_a$
$PU_b$
$C_A = E(PR_{auth}, [T_1 \| ID_A \| PU_a])$
$C_B = E(PR_{auth}, [T_2 \| ID_B \| PU_b])$
A
B

(a) Obtaining certificates from CA

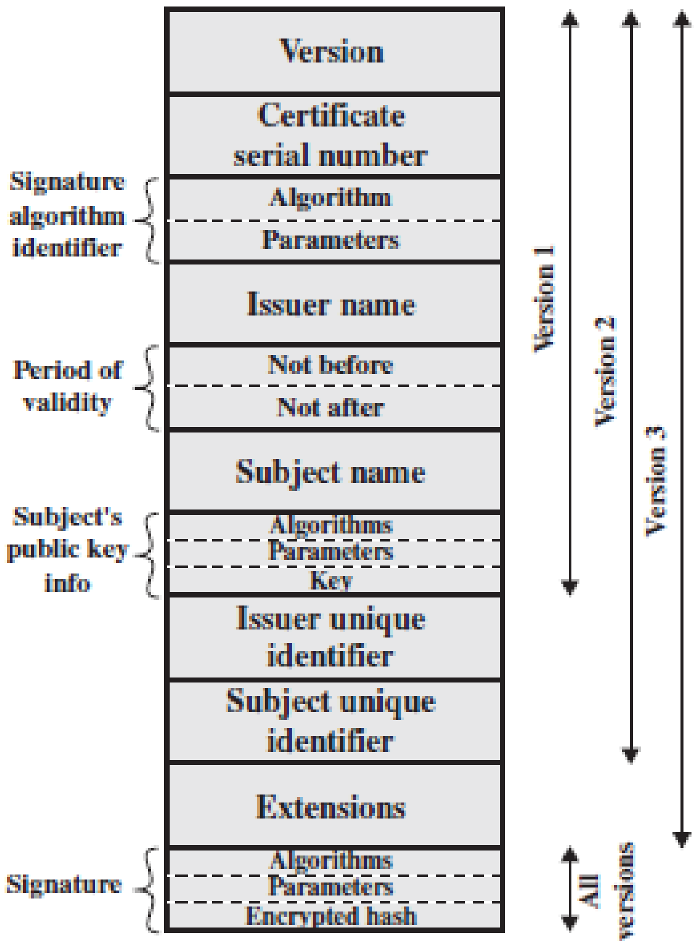


$\|PU_a)$

(b) Exchanging certificates

## X.509 CERTIFICATE

- The public-key certificate associated with each user. These user certificates are assumed to be created by some trusted certification authority (CA) and placed in the directory by the CA or by the user.
- The directory server itself is not responsible for the creation of public keys or for the certification function; it merely provides an easily accessible location for users to obtain certificates.



(a) X.509 certificate

• **Version:** Differentiates among successive versions of the certificate format; the default is version 1

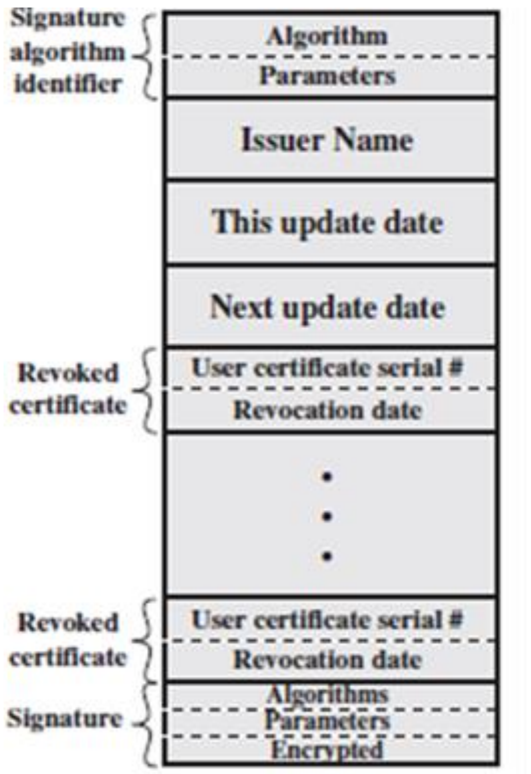
1. If the issuer unique identifier or subject unique identifier are present, the value must be version

2. If one or more extensions are present, the version must be version 3.

• **Serial number**: An integer value unique within the issuing CA that is unambiguously associated with this certificate.

- **Signature algorithm identifier**: The algorithm used to sign the certificate Together with any associated parameters. Because this information is repeated in the signature field at the end of the certificate, this field has little, if any, utility.
- **Issuer name**: X.500 name of the CA that created and signed this certificate.
- **Period of validity**: Consists of two dates: the first and last on which the certificate is valid.
- **Subject name**: The name of the user to whom this certificate refers. That is, this certificate certifies the public key of the subject who holds the corresponding private key.
- **Subject's public-key information**: The public key of the subject, plus an identifier of the algorithm for which this key is to be used, together with any associated parameters.
- **Issuer unique identifier**: An optional-bit string field used to identify uniquely the issuing CA in the event the X.500 name has been reused for different entities.
- **Subject unique identifier**: An optional-bit string field used to identify uniquely the subject in the event the X.500 name has been reused for different entities.
- **Extensions**: A set of one or more extension fields. Extensions were added in version 3 and are discussed later in this section.
- **Signature**: Covers all of the other fields of the certificate; it contains the hash code of the other fields encrypted with the CA's private key. This field includes the signature algorithm identifier.

**Certificate Revocation List:**



(b) Certificate revocation list

**Obtaining a User's Certificate**

User certificates generated by a CA have the following characteristics:

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

A has obtained a certificate from certification authority X1 and B has obtained a certificate from CA X2. If A does not securely know the public key of X2, then B's certificate, issued by X2, is useless to A. A can read B's certificate, but A cannot verify the signature. However, if the two CAs have securely exchanged their own public keys,

The following procedure will enable A to obtain B's public key.

Step 1 A obtains from the directory the certificate of X2 signed by X1. Because A securely knows X1's public key, A can obtain X2's public key from its certificate and verify it by means of X1's signature on the certificate.

Step 2 A then goes back to the directory and obtains the certificate of B signed by X2. Because A now has a trusted copy of X2's public key, A can verify the signature and securely obtain B's public key.

X.509 suggests that CAs be arranged in a hierarchy so that navigation is straightforward.

- **Forward certificates**: Certificates of X generated by other CAs
- **Reverse certificates**: Certificates generated by X that are the certificates of other CAs

## PKIX INFRASTRUCTURE

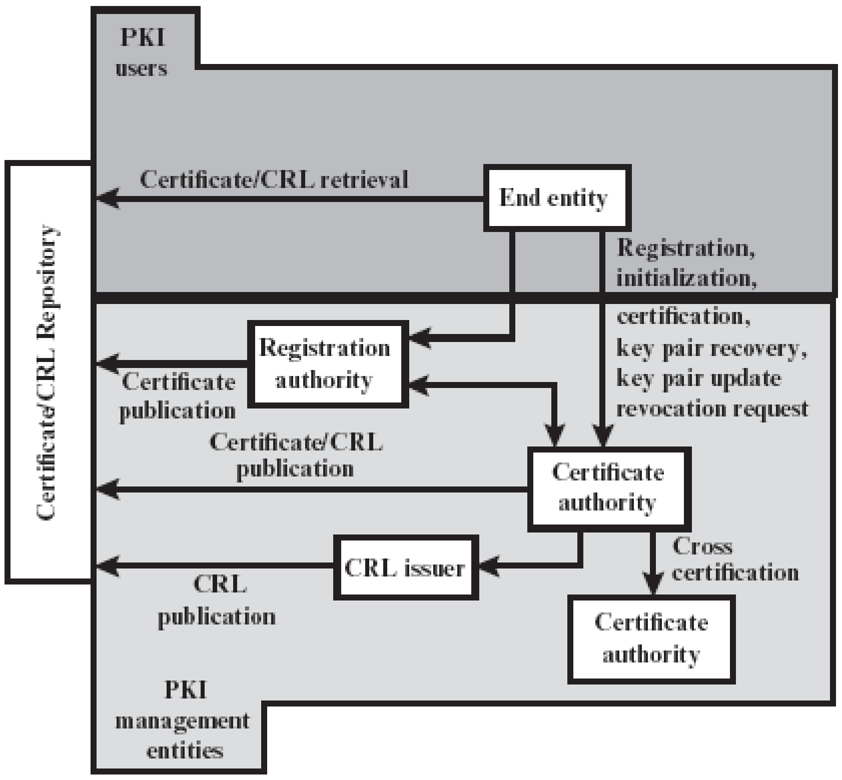
The principal objective for developing a PKI is to enable secure, convenient, and efficient acquisition of public keys.

**The Elements of PKIX Model are:**

- **End entity:** A generic term used to denote end users, devices or any other entity that can be identified in the subject field of a public-key certificate.
- **Certification authority (CA):** The issuer of certificates and (usually) certificate revocation lists (CRLs).
- **Registration authority (RA):** The RA is often associated with the end entity registration process.
- **CRL issuer:** An optional component that a CA can delegate to publish CRLs.
- **Repository:** A generic term used to denote any method for storing certificates and CRLs so that they can be retrieved by end entities.

**The PKIX Management Functions are:**

- **Registration:** This is the process whereby a user first makes itself known to a CA (directly or through an RA), prior to that CA issuing a certificate or certificates for that user.
- **Initialization:** Before a client system can operate securely, it is necessary to install key materials that have the appropriate relationship with keys stored elsewhere in the infrastructure.


PKI users
Certificate/CRL Repository
Certificate/CRL retrieval
End entity
Registration, initialization, certification, key pair recovery, key pair update revocation request
Registration authority
Certificate publication
Certificate/CRL publication
Certificate authority
CRL issuer
CRL publication
Cross certification
Certificate authority
PKI management entities

- **Certification:** This is the process in which a CA issues a certificate for a user's public key.
- **Key pair recovery**: Key pairs can be used to support digital signature creation and verification, encryption and decryption, or both. Key pair recovery allows end entities to restore their encryption/decryption key pair from an authorized key backup facility.
- **Key pair update**: All key pairs need to be updated regularly and new certificates issued. Update is required when the certificate lifetime expires and as a result of certificate revocation.
- **Revocation request**: An authorized person advises a CA of an abnormal situation requiring certificate revocation. Reasons for revocation include private key compromise, change in affiliation, and name change.
- **Cross certification**: A cross- certificate is a certificate issued by one CA to another CA that contains a CA signature key used for issuing certificates.

## USER AUTHENTICATION: REMOTE USER-AUTHENTICATION PRINCIPLES

In most computer security contexts, user authentication is the fundamental building block and the primary line of defense. User authentication is the basis for most types of access control and for user accountability.

An Authentication process consists of two steps:

- **Identification step**
- **Verification step**.

**Four general means of authenticating a user's identity:**

1. **Something the individual knows**: Examples include a password, a personal identification number (PIN), or answers to a prearranged set of questions.
2. **Something the individual possesses**: Examples include cryptographic keys, electronic keycards, smart cards, and physical keys. This type of authenticator is referred to as a token.
3. **Something the individual is** (static biometrics): Examples include recognition by fingerprint, retina, and face.
4. **Something the individual does** (dynamic biometrics): Examples include recognition by voice pattern, handwriting characteristics, and typing rhythm.

**Mutual Authentication**

Protocols enable communicating parties to satisfy themselves mutually about each other's identity and to exchange session keys.

**Examples of replay attacks:**

- **Simple replay**
- **Repetition that can be logged**
- **Repetition that cannot be detected**
- **Backward replay without modification**

One approach to coping with replay attacks is to attach a sequence number to each message used in an authentication exchange. A new message is accepted only if its sequence number is in the proper order.

The difficulty with this approach is that it requires each party to keep track of the last sequence number for each claimant it has dealt with.

Because of this overhead, sequence numbers are generally not used for authentication and key exchange. Instead, one of the following two general approaches is used:

• **Timestamps:** Party A accepts a message as fresh only if the message contains a timestamp that, in A’s judgment, is close enough to A’s knowledge of current time. This approach requires that clocks among the various participants be synchronized.

• **Challenge/response**: Party A, expecting a fresh message from B, first sends B a nonce (challenge) and requires that the subsequent message (response) received from B contain the correct nonce value.

## <u>REMOTE USER-AUTHENTICATION USING SYMMETRIC ENCRYPTION</u>

### Mutual Authentication

This strategy involves the use of a trusted key distribution center (KDC). Each party in the network shares a secret key, known as a master key, with the KDC. The KDC is responsible for generating keys to be used for a short time over a connection between two parties, known as session keys, and for distributing those keys using the master keys to protect the distribution.

Figure below illustrates a proposal initially put forth by **Needham and Schroeder** for secret key distribution using a KDC. The protocol can be summarized as follows.

1. $\mathrm{A} \rightarrow \mathrm{KDC}$: $ID_A \| ID_B \| N_1$
2. $\mathrm{KDC} \rightarrow \mathrm{A}$: $\mathrm{E}(K_a, [K_s \| ID_B \| N_1 \| \mathrm{E}(K_b, [K_s \| ID_A])])$
3. $\mathrm{A} \rightarrow \mathrm{B}$: $\mathrm{E}(K_b, [K_s \| ID_A])$
4. $\mathrm{B} \rightarrow \mathrm{A}$: $\mathrm{E}(K_s, N_2)$
5. $\mathrm{A} \rightarrow \mathrm{B}$: $\mathrm{E}(K_s, \mathrm{f}(N_2))$

Secret keys Ka and Kb are shared between A and the KDC and B and the KDC, respectively.

The purpose of the protocol is to distribute securely a session key Ks to A and B. A securely acquires a new session key in step 2.

The message in step 3 can be decrypted, and hence understood, only by B. Step 4 reflects B's knowledge of Ks, and step 5 assures B of A's knowledge of Ks and assures B that this is a fresh message because of the use of the nonce N2.

Despite the handshake of steps 4 and 5, the protocol is still vulnerable to a form of replay attack.

**Denning** proposes to overcome this weakness by a modification to the Needham/Schroeder protocol that includes the addition of a time stamp to steps 2 and 3. The proposal assumes that the master keys, Ka and Kb, are secure, and it consists of the following steps.

1. $\mathrm{A} \rightarrow \mathrm{KDC}$: $ID_A \| ID_B$
2. $\mathrm{KDC} \rightarrow \mathrm{A}$: $\mathrm{E}(K_a, [K_s \| ID_B \| T \| \mathrm{E}(K_b, [K_s \| ID_A \| T])])$
3. $\mathrm{A} \rightarrow \mathrm{B}$: $\mathrm{E}(K_b, [K_s \| ID_A \| T])$
4. $\mathrm{B} \rightarrow \mathrm{A}$: $\mathrm{E}(K_s, \mathrm{N}_1)$
5. $\mathrm{A} \rightarrow \mathrm{B}$: $\mathrm{E}(K_s, \mathrm{f}(N_1))$

T is a timestamp that assures A and B that the session key has only just been generated. Thus, both A and B know that the key distribution is a fresh exchange.

The Denning protocol seems to provide an increased degree of security compared to the Needham/Schroeder protocol.

However, a new concern is raised: namely, that this new scheme requires reliance on clocks that are synchronized throughout the network.

The problem occurs when a sender's clock is ahead of the intended recipient's clock. In this case, an opponent can intercept a message from the sender and replay it later when the timestamp in the message becomes current at the recipient's site. This replay could cause unexpected results. Gong refers to such attacks as suppress replay attacks.

One way to counter suppress-replay attacks is to enforce the requirement that parties regularly check their clocks against the KDC's clock.

An attempt is made to respond to the concerns about suppress- replay attacks and at the same time fix the problems in the Needham/Schroeder protocol. Subsequently, an inconsistency in this latter protocol was noted and an improved strategy was presented in. The protocol is

1. $A \rightarrow B$: $ID_A \| N_a$
2. $B \rightarrow KDC$: $ID_B \| N_b \| E(K_b, [ID_A \| N_a \| T_b])$
3. $KDC \rightarrow A$: $E(K_a, [ID_B \| N_a \| K_s \| T_b]) \| E(K_b, [ID_A \| K_s \| T_b]) \| N_b$
4. $A \rightarrow B$: $E(K_b, [ID_A \| K_s \| T_b]) \| E(K_s, N_b)$

A initiates the authentication exchange by generating a nonce, Na, and sending that plus its identifier to B in plaintext. This nonce will be returned to A in an encrypted message that includes the session key, assuring A of its timeliness.
B alerts the KDC that a session key is needed. Its message to the KDC includes its identifier and a nonce, Nb. This nonce will be returned to B in an encrypted message that includes the session key, assuring B of its timeliness.
B's message to the KDC also includes a block encrypted with the secret key shared by B and the KDC. This block is used to instruct the KDC to issue credentials to A; the block specifies the intended recipient of the credentials, a suggested expiration time for the credentials, and the nonce received from A.
The KDC passes on to A B's nonce and a block encrypted with the secret key that B shares with the KDC. The block serves as a "ticket" that can be used by A for subsequent authentications, as will be seen. The KDC also sends to A a block encrypted with the secret key shared by A and the KDC. This block verifies that B has received A's initial message (IDB) and that this is a timely message and not a replay (Na), and it provides A with a session key (Ks) and the time limit on its use (Tb).
A transmits the ticket to B, together with the B's nonce, the latter encrypted with the session key. The ticket provides B with the secret key that is used to decrypt E(Ks, Nb) to recover the nonce..

**One-Way Authentication**

1. $A \rightarrow KDC$: $ID_A \| ID_B \| N_1$
2. $KDC \rightarrow A$: $E(K_a, [K_s \| ID_B \| N_1 \| E(K_b, [K_s \| ID_A])])$
3. $A \rightarrow B$: $E(K_b, [K_s \| ID_A]) \| E(K_s, M)$

This approach guarantees that only the intended recipient of a message will be able to read it. It also provides a level of authentication that the sender is A.

## KERBEROS SYSTEMS

Kerberos is an authentication service.

**Advantages:**

- **Secure**
- **Reliable**
- **Transparent**
- **Scalable**

**Kerberos Version 4**

**A Simple Authentication Dialogue**

**(1)** $C \rightarrow AS: ID_C \| P_C \| ID_V$

**(2)** $A \rightarrow C$: Ticket

**(3)** $C \rightarrow V: ID_C \|$ Ticket

Ticket = $E(K_V, [ID_C \| AD_C \| ID_V])$

where

C = client

AS = authentication server

V = server

$ID_C$ = identifier of user on CID

$_V$ = identifier of V

$P_C$ = password of user on C

$AD_C$ = network address of C

$K_V$ = secret encryption key shared by AS and V

**A More Secure Authentication Dialogue**

Introduce a scheme for avoiding plaintext passwords and a new server, known as the **ticket-granting server** (TGS).

**Once per user logon session:**

**(1)** $C \rightarrow AS: ID_C \| ID_{tgs}$

**(2)** $AS \rightarrow C: E(K_C, Ticket_{tgs})$

**Once per type of service:**

**(3)** $C \rightarrow TGS: ID_C \| ID_V \| Ticket_{tgs}$

**(4)** $TGS \rightarrow C: Ticket_V$

**Once per service session:**

**(5)** $C \rightarrow V: ID_C \| Ticket_V$

$Ticket_{tgs} = E(K_{tgs}, [ID_C \| AD_C \| ID_{tgs} \| TS_1 \| Lifetime_1])$

$Ticket_V = E(K_V, [ID_C \| AD_C \| ID_V \| TS_2 \| Lifetime_2])$

The new service, TGS, issues tickets to users who have been authenticated to AS. Thus, the user first requests a ticket-granting ticket ($Ticket_{tgs}$) from the AS. The client module in the user workstation saves this ticket. Each time the user requires access to a new service, the client appliesto the TGS, using the ticket to authenticate itself. The TGS then grants a ticket for the particular service. The client saves each service-granting ticket and uses it to authenticate its user to a server each time a particular service is requested.

**1.** The client requests a ticket-granting ticket on behalf of the user by sending its user's ID to theAS, together with the TGS ID, indicating a request to use the TGS service.

**2.** The AS responds with a ticket that is encrypted with a key that is derivedfrom the user's password ($K_C$), which is already stored at the AS. When this response arrives at the client, the client prompts the user for his or her password, generates the key, and attempts to decrypt the incoming message. If the correct password is supplied, the ticket is successfully recovered.

**3.** The client requests a service-granting ticket on behalf of the user. For this purpose, the client transmits a message to the TGS containing the user's ID, the ID of the desired service, and the ticket-granting ticket.

**4.** The TGS decrypts the incoming ticket using a key shared only by the AS and the TGS ($K_{tgs}$) and verifies the success of the decryption by the presence of its ID. It checks to make sure that thelifetime has not expired. Then it compares the user ID and network address with the incoming information to authenticate the user. If the user is permitted access to the server V, the TGS issues ticket to grant access to the requested service.

5. The client requests access to a service on behalf of the user. For this purpose, the client transmitsa message to the server containing the user's ID and the service granting ticket. The server authenticates by using the contents of the ticket.

**Authentication Service Exchange to obtain ticket-granting ticket**

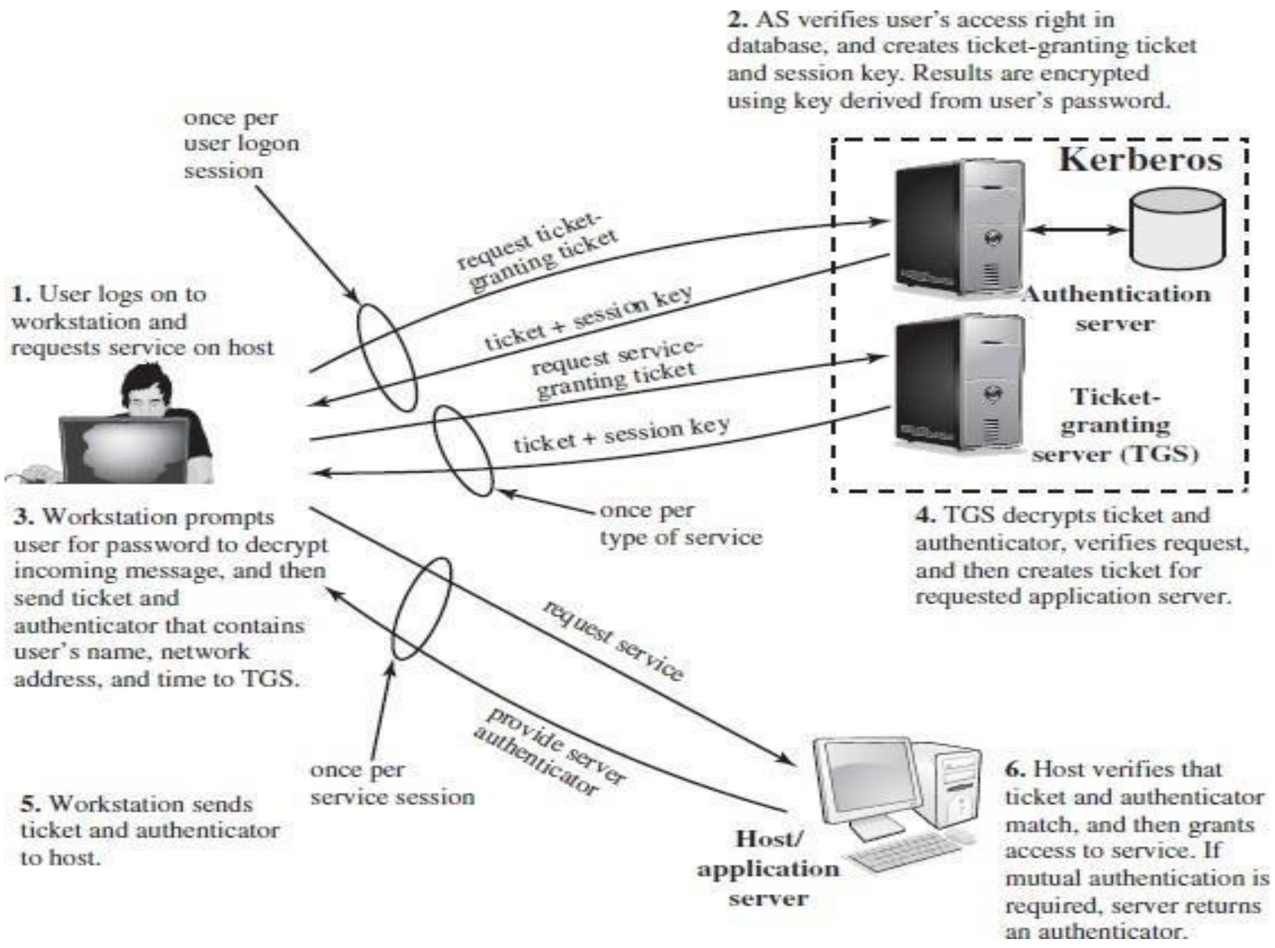
$$
\begin{array}{lll}
(1) & \mathbf{C \rightarrow AS} & ID_c \| ID_{tgs} \| TS_1 \\
(2) & \mathbf{AS \rightarrow C} & \mathrm{E}(K_c, [K_{c,tgs} \| ID_{tgs} \| TS_2 \| Lifetime_2 \| Ticket_{tgs}]) \\
 & & Ticket_{tgs} = \mathrm{E}(\mathrm{K}_{tgs}, [\mathrm{K}_{c,tgs} \| \mathrm{ID}_C \| \mathrm{AD}_C \| \mathrm{ID}_{tgs} \| \mathrm{TS}_2 \| \mathrm{Lifetime}_2])
\end{array}
$$

**Ticket-Granting Service Exchange to obtain service-granting ticket**

$$
\begin{array}{lll}
(3) & \mathbf{C \rightarrow TGS} & ID_v \| Ticket_{tgs} \| Authenticator_c \\
(4) & \mathbf{TGS \rightarrow C} & \mathrm{E}(K_{c,tgs}, [K_{c,v} \| ID_v \| TS_4 \| Ticket_v]) \\
 & & Ticket_{tgs} = \mathrm{E}(\mathrm{K}_{tgs}, [\mathrm{K}_{c,tgs} \| \mathrm{ID}_C \| \mathrm{AD}_C \| \mathrm{ID}_{tgs} \| \mathrm{TS}_2 \| \mathrm{Lifetime}_2]) \\
 & & Ticket_v = \mathrm{E}(\mathrm{K}_v, [\mathrm{K}_{c,v} \| \mathrm{ID}_C \| \mathrm{AD}_C \| \mathrm{ID}_v \| \mathrm{TS}_4 \| \mathrm{Lifetime}_4]) \\
 & & Authenticator_c = \mathrm{E}(\mathrm{K}_{c,tgs}, [\mathrm{ID}_C \| \mathrm{AD}_C \| \mathrm{TS}_3])
\end{array}
$$

**Client/Server Authentication Exchange to obtain service**

$$
\begin{array}{lll}
(5) & \mathbf{C \rightarrow V} & Ticket_v \| Authenticator_c \\
(6) & \mathbf{V \rightarrow C} & \mathrm{E}(K_{c,v}, [TS_5 + 1]) \text{ (for mutual authentication)} \\
 & & Ticket_v = \mathrm{E}(\mathrm{K}_v, [\mathrm{K}_{c,v} \| \mathrm{ID}_C \| \mathrm{AD}_C \| \mathrm{ID}_v \| \mathrm{TS}_4 \| \mathrm{Lifetime}_4]) \\
 & & Authenticator_c = \mathrm{E}\ (\mathrm{K}_{c,v}, [\mathrm{ID}_C \| \mathrm{AD}_C \| \mathrm{TS}_5])
\end{array}
$$


2. AS verifies user's access right in database, and creates ticket-granting ticket and session key. Results are encrypted using key derived from user's password.
once per user logon session
Kerberos
request ticket-granting ticket
ticket + session key
1. User logs on to workstation and requests service on host
Authentication server
request service-granting ticket
ticket + session key
Ticket-granting server (TGS)
once per type of service
3. Workstation prompts user for password to decrypt incoming message, and then send ticket and authenticator that contains user's name, network address, and time to TGS.
4. TGS decrypts ticket and authenticator, verifies request, and then creates ticket for requested application server.
request service
provide server authenticator
once per service session
5. Workstation sends ticket and authenticator to host.
Host/ application server
6. Host verifies that ticket and authenticator match, and then grants access to service. If mutual authentication is required, server returns an authenticator.

**Kerberos Realms and Multiple Kerberi**

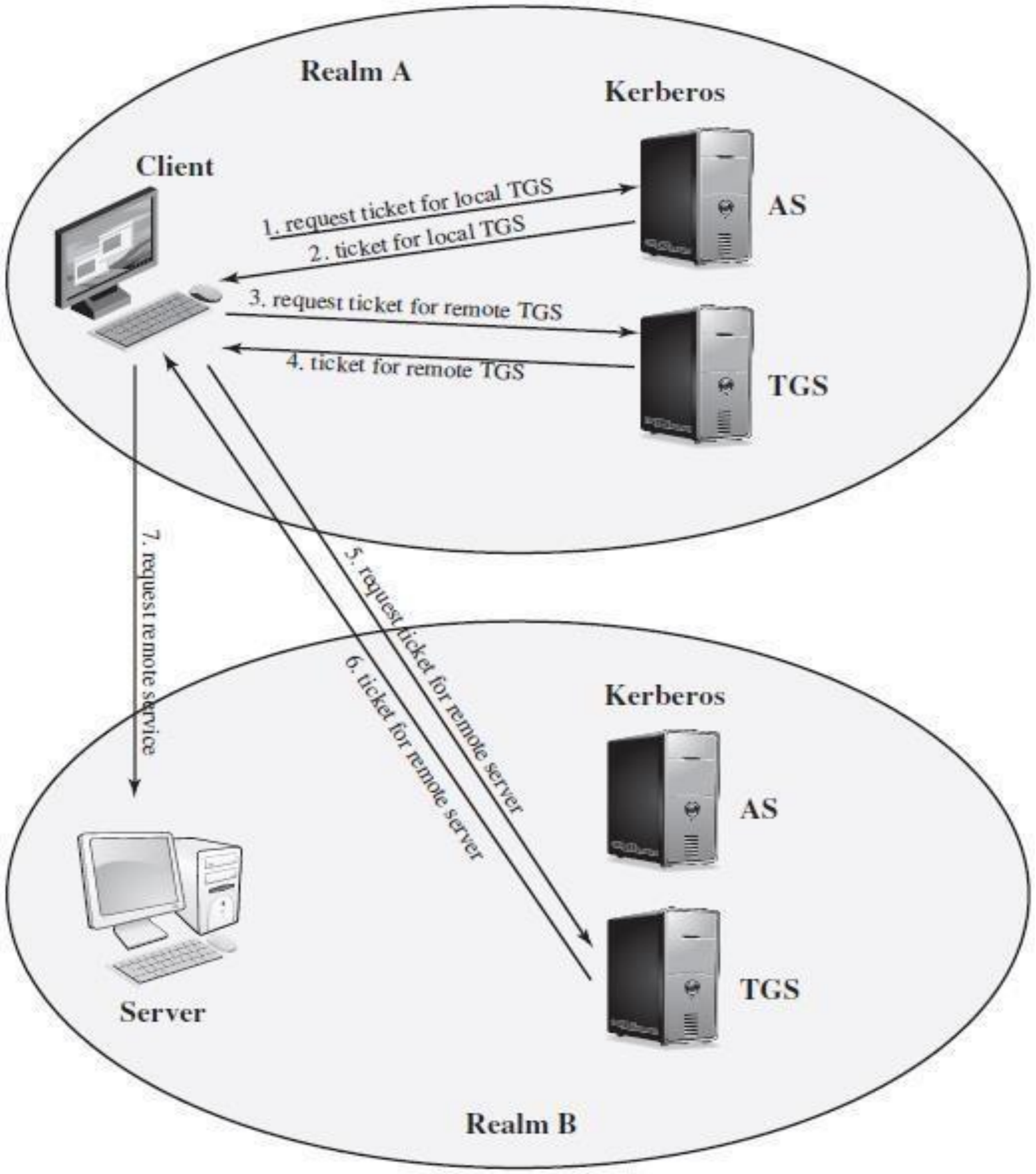
A full-service Kerberos environment consisting of a Kerberos server, a number of clients, and anumber of application servers requires the following:

1. The Kerberos server must have the user ID and hashed passwords of all participating users inits database. All users are registered with the Kerberos server.
2. The Kerberos server must share a secret key with each server. All servers are registered withthe Kerberos server.

Such an environment is referred to as a **Kerberos realm**. The concept of **realm** can be explainedas follows.

- A Kerberos realm is a set of managed nodes that share the same Kerberos database.
- The Kerberos database resides on the Kerberos master computer system, which should bekept in a physically secure room.
- A read only copy of the Kerberos database might also reside on other Kerberos computer systems. However, all changes to the database must be made on the master computer system.

- Changing or accessing the contents of a Kerberos database requires the Kerberos master password. A related concept is that of a **Kerberos principal**, which is a service or user thatis known to the Kerberos system.


Realm A
Kerberos
Client
AS
1. request ticket for local TGS
2. ticket for local TGS
3. request ticket for remote TGS
4. ticket for remote TGS
TGS
7. request remote service
5. request ticket for remote server
6. ticket for remote server
Kerberos
AS
Server
TGS
Realm B

- Each Kerberos principal is identified by its principal name. Principal names consist of three parts: a service or user name, an instance name, and a realm name. Kerberos provides a mechanism for supporting such interrealm authentication. For two realms tosupport interrealm authentication, a third requirement is added:

**3.** The Kerberos server in each interoperating realm shares a secret key with the server in the otherrealm. The two Kerberos servers are registered with each other.

**(1)** C → AS: $ID_c$ || $ID_{tgs}$ || $TS_1$

**(2)** AS → C: $E(K_c$, [$K_{c, tgs}$ || $ID_{tgs}$ || $TS_2$ || $Lifetime_2$ || $Ticket_{tgs}$])

**(3)** C → TGS: $ID_{tgsrem}$ || $Ticket_{tgs}$ || $Authenticator_c$

**(4)** TGS → C: $E(K_{c,tgs}$, [$K_{c, tgsrem}$ || $ID_{tgsrem}$ || $TS_4$ || $Ticket_{tgsrem}$])

**(5)** C → $TGS_{rem}$: $ID_{vrem}$ || $Ticket_{tgsrem}$ || $Authenticator_c$

**(6)** $TGS_{rem}$ → C: $E(K_{c,tgsrem}$, [$K_{c, vrem}$ || $ID_{vrem}$ || $TS_6$ || $Ticket_{vrem}$])

**(7)** C → $V_{rem}$: $Ticket_{vrem}$ || $Authenticator_c$

## REMOTE USER AUTHENTICATION USING ASYMMETRIC ENCRYPTION

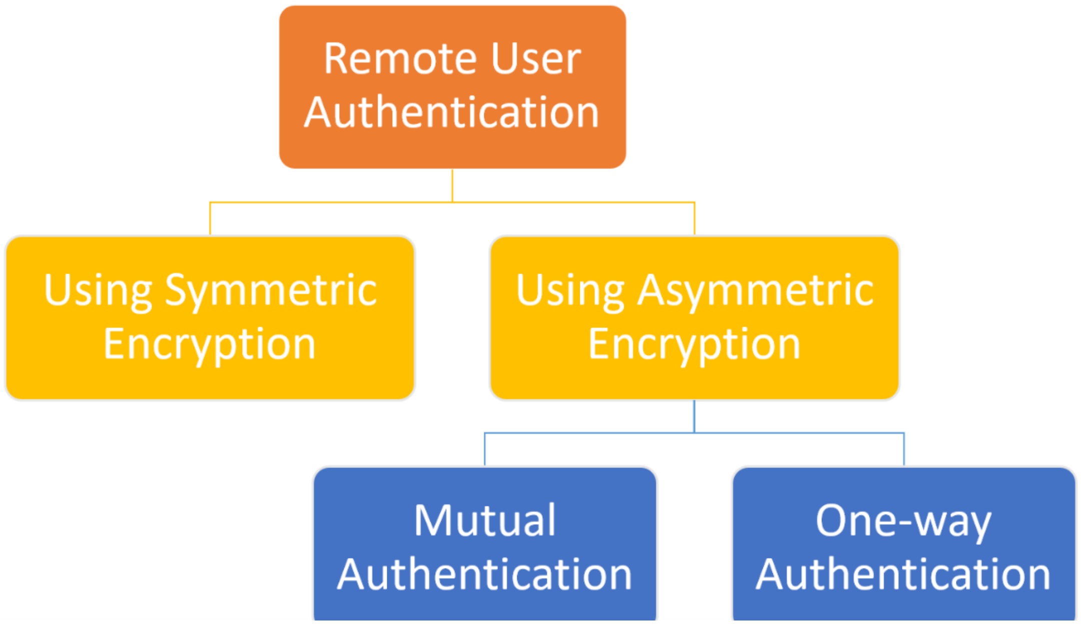
**What is User Authentication?**

User authentication is the process of recognizing a user's identity.

**What is Remote user authentication?**

It is a mechanism in which the remote server verifies the legitimacy of a user over non-secure communication channel.

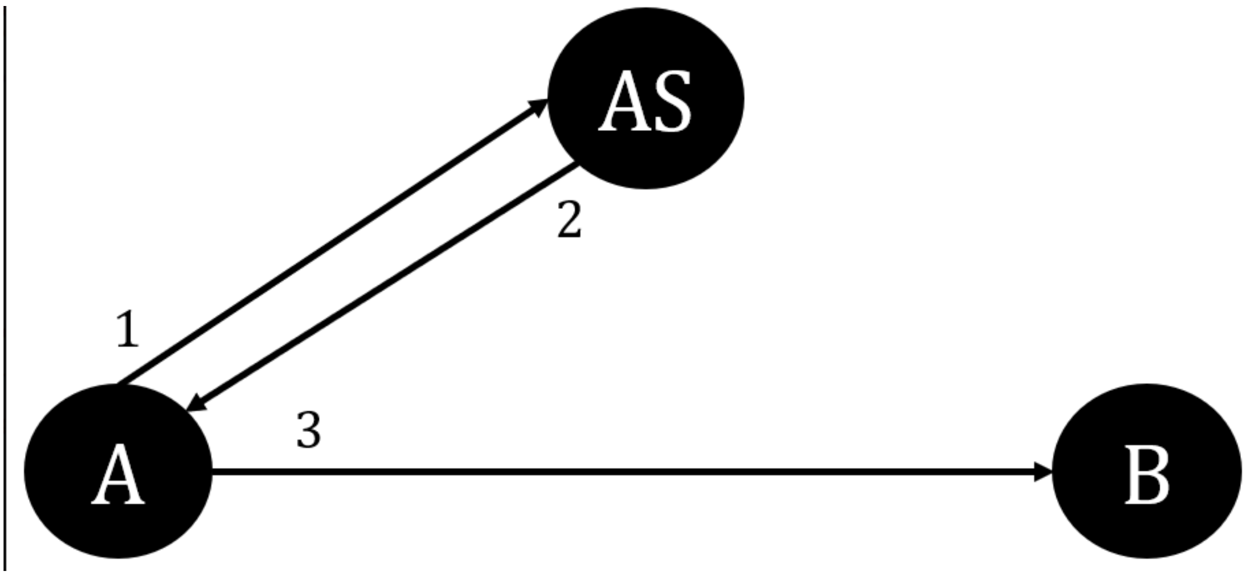
There are two types of remote user authentication:

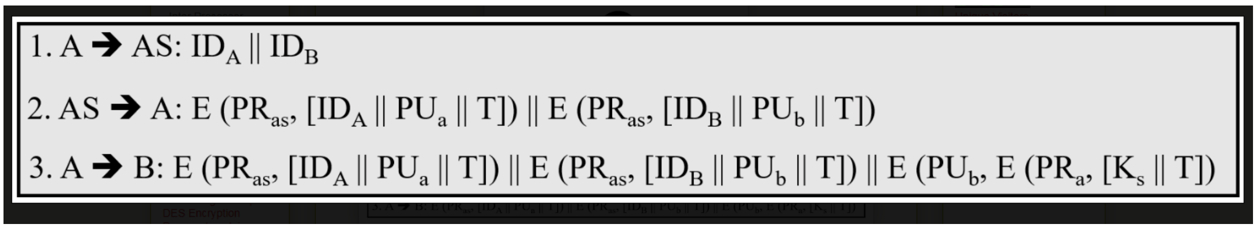

Remote User Authentication
Using Symmetric Encryption
Using Asymmetric Encryption
Mutual Authentication
One-way Authentication

**Mutual Authentication**

One approach to the use of public-key encryption for the purpose of session-key distribution.

**Solution-1:**





1. A ➔ AS: $ID_A \| ID_B$

2. AS ➔ A: $E(PR_{as}, [ID_A \| PU_a \| T]) \| E(PR_{as}, [ID_B \| PU_b \| T])$

3. A ➔ B: $E(PR_{as}, [ID_A \| PU_a \| T]) \| E(PR_{as}, [ID_B \| PU_b \| T]) \| E(PU_b, E(PR_a, [K_s \| T]))$

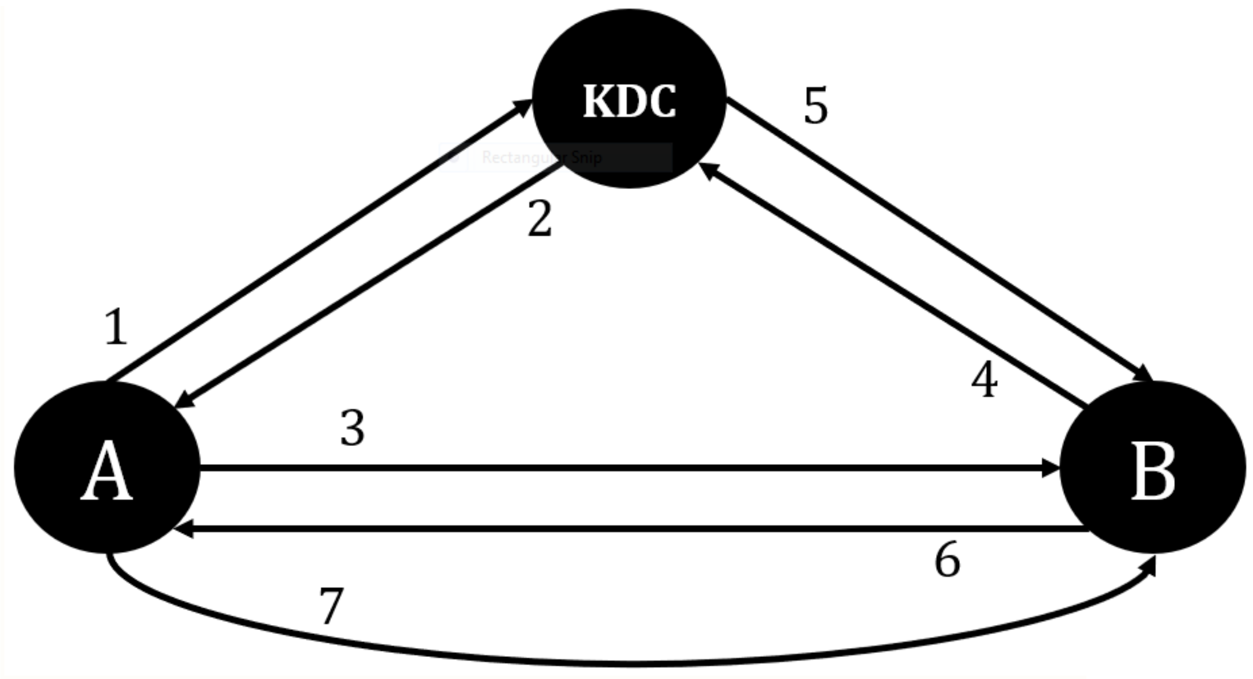
In this case, the central system is referred to as an authentication server (AS), because it is not actually responsible for secret-key distribution. Rather, the AS provides public-key certificates.

The session key is chosen and encrypted by A; hence, there is no risk of exposure by the AS.

The timestamps protect against replays of compromised keys.

This protocol is compact but, as before, requires the synchronization of clocks.

**Solution-2:**



1. A ➔ KDC: $ID_A \,||\, ID_B$
2. KDC ➔ A: E($PR_{\text{auth}}$, [$ID_B \,||\, PU_b$])
3. A ➔ B: E($PU_b$, [$N_a \,||\, ID_A$])
4. B ➔ KDC: $ID_A \,||\, ID_B \,||$ E($PU_{\text{auth}}$, $N_a$ )
5. KDC ➔ B: E($PR_{\text{auth}}$, [$ID_A \,||PU_a$]) || E($PU_b$, E($PR_{\text{auth}}$, [$N_a \,||\, K_s \,||\, ID_A \,||\, ID_B$]))
6. B ➔ A: E($PU_a$, [E($PR_{\text{auth}}$, [$N_a \,||\, K_s \,||\, ID_A \,||\, ID_B$])|| $N_b$])
7. A ➔ B: E($K_s$, $N_b$)

In step-1, A informs the KDC of its intention to establish a secure connection with B.

In step-2, The KDC returns to A the B's public-key certificate

In step-3, using B's public key, A informs B of its desire to communicate and sends a nonce.

In step-4, B asks the KDC for A's public-key certificate and requests a session key; B includes A's nonce so that the KDC can stamp the session key with that nonce. The nonce is protected using the KDC's public key.

In step-5, the KDC returns to B a copy of A's public-key certificate, plus the information Na, Ks, IDB, IDA. This information basically says that Ks is generated by KDC on behalf of B and tied to Na. This information is encrypted using the KDC's private key to allow B to verify that the information is in fact from the KDC. It is also encrypted using B's public key so that no other entity may use the information in an attempt to establish a fraudulent connection with A.

In step-6 and 7, this information encrypted with PRAuth is given to A together with Nb. All the foregoing is encrypted using A’s public key. A retrieve the session key Ks and uses it to encrypt Nb.

**One-way Authentication**

There are focus on two main areas, when communication on non-secure network: **Confidentiality and Authentication**.

$$\begin{tikzcd}
\mathrm{A} \Rightarrow \mathrm{B}: \mathrm{E}(PU_b, [M \,||\, \mathrm{E}(PR_a, H(M))]) \arrow[d] \\
\mathrm{A} \Rightarrow \mathrm{B}: M \,||\, \mathrm{E}(PR_a, \mathrm{H}(M)) \,||\, \mathrm{E}(PR_{as}, [T \,||\, ID_A \,||\, PU_a]) \arrow[d] \\
\mathrm{A} \Rightarrow \mathrm{B}: \mathrm{E}(Pu_b, (M \,||\, \mathrm{E}(PR_a, \mathrm{H}(M)) \,||\, \mathrm{E}(PR_{as}, [T \,||\, ID_A \,||\, PU_a]))
\end{tikzcd}$$

**If confidentiality is the primary concern**, then the following may be more efficient.

In this case, the message is encrypted with a one-time secret key. A also encrypts this one-time key with B’s public key. Only B will be able to use the corresponding private key to recover the one-time key and then use that key to decrypt the message. This scheme is more efficient than simply encrypting the entire message with B’s public key.

**If authentication is the primary concern**, then a digital signature may suit.

This method guarantees that A cannot later deny having sent the message. However, this technique is open to another kind of fraud. Bob composes a message to his boss Alice that contains an idea that will save the company money. He appends his digital signature and sends it into the e-mail system. Eventually, the message will get delivered to Alice’s mailbox. But suppose that Max has heard of Bob’s idea and gains access to the mail queue before delivery. He finds Bob’s message, strips off his signature, appends his, and requeues the message to be delivered to Alice. Max gets credit for Bob’s idea.