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.

2.1 Key Management and Distribution: Symmetric Key Distribution

Symmetric key Distribution using Symmetric Encryption

For symmetric encryption to work, the two parties to an exchange must share the same key, and that key must be protected from access by others. Therefore, the term that refers to the means of delivering a key to two parties who wish to exchange data, without allowing others to see the key. 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 encrypted links to A and B.

Key distribution Centre:

- The use of a key distribution center is based on the use of a hierarchy of keys. At a minimum, two levels of keys are used.
- Communication between end systems is encrypted using a temporary key, often referred to as a Session key.
- Typically, the session key is used for the duration of a logical connection and then discarded
- Master key is shared by the key distribution center and an end system or user and used to encrypt the session key.

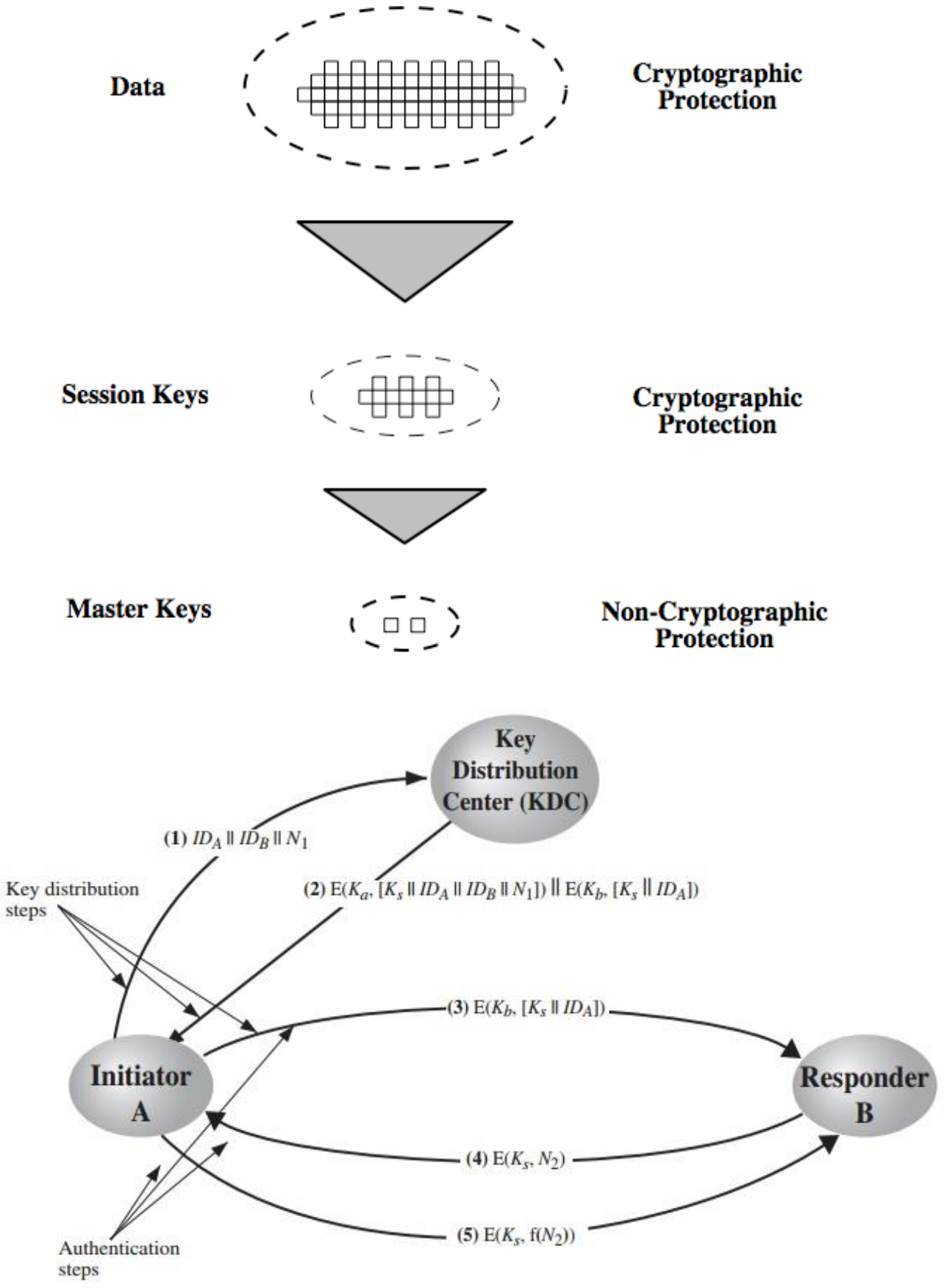


Figure Key Distribution Scenario

Let us assume that 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, Ka, 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, N1, 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 Ka 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, Ks, to be used for the session
- The original request message

Thus, A can verify that its original request. In addition, the message includes two items intended for B:

- The one-time session key, Ks to be used for the session
- An identifier of A (e.g., its network address), IDA

These last two items are encrypted with Kb (the master key that the KDC shares with B).

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, $E(K_b, [K_s \| ID_A])$. At this point, a session key has been securely delivered to A and B, and they may begin their protected exchange. However, two additional steps are desirable:

4. Using the newly minted session key for encryption, B sends a nonce, N2, to A.

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

Major Issues with KDC:

Hierarchical Key Control

- It is not necessary to limit the key distribution function to a single KDC. Indeed, for very large networks, it may not be practical to do so. As an alternative, a hierarchy of KDCs can be established.

Session Key Lifetime

- The distribution of session keys delays the start of any exchange and places a burden on network capacity. A security manager must try to balance these competing considerations in determining the lifetime of a particular session key.

A Transparent Key Control Scheme

- The approach suggested in Figure is useful for providing end-to-end encryption at a network or transport level in a way that is transparent to the end users.
- The approach assumes that communication makes use of a connection-oriented end-to- end protocol, such as TCP.
- The noteworthy element of this approach is a session security module (SSM), which may consist of functionality at one protocol layer, that performs end-to-end encryption and obtains session keys on behalf of its host or terminal.

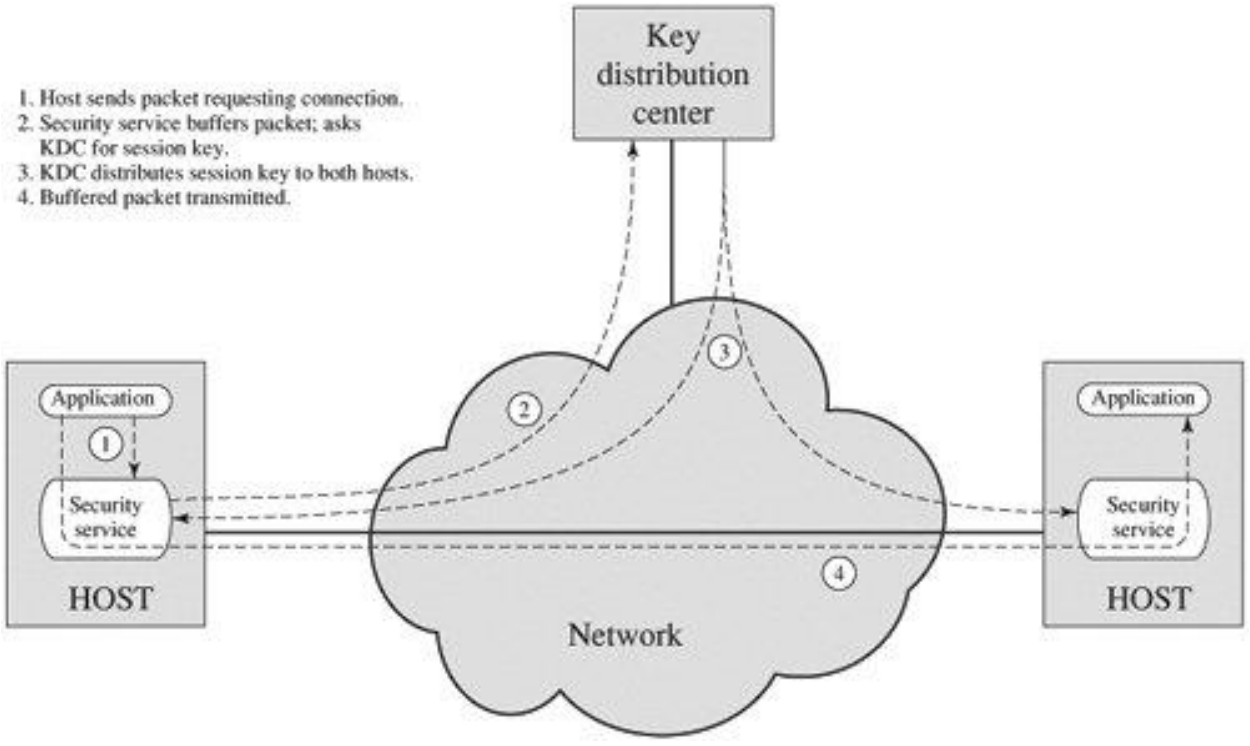


Fig: Automatic key distribution for connection oriented Protocol

Decentralized Key Control

- The use of a key distribution center imposes the requirement that the KDC be trusted and be protected from subversion. This requirement can be avoided if key distribution is fully decentralized.

Controlling Key Usage

The concept of a key hierarchy and the use of automated key distribution techniques greatly reduce the number of keys that must be manually managed and distributed. It also may be desirable to impose some control on the way in which automatically distributed keys are used. For example, in addition to separating master keys from session keys, we may wish to define different types of session keys on the basis of use, such as

• Data-encrypting key, for general communication across a network

• PIN-encrypting key, for personal identification numbers (PINs) used in electronic funds transfer and point-of-sale applications

• File-encrypting key, for encrypting files stored in publicly accessible locations

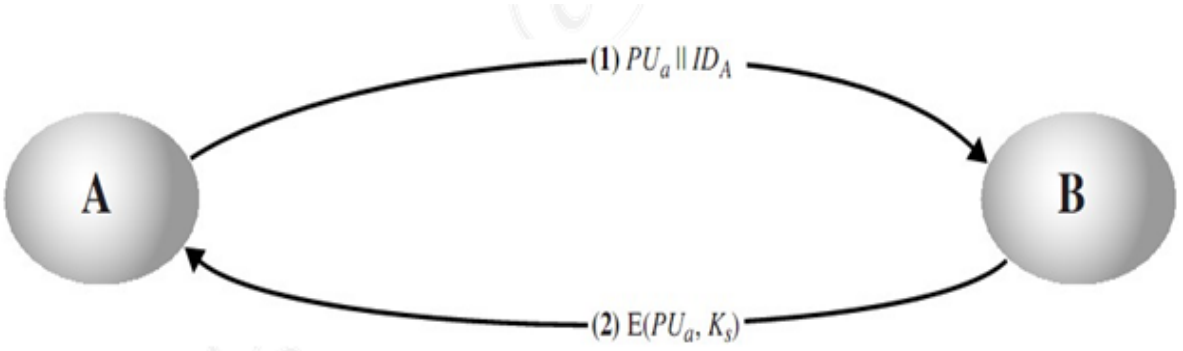
SYMMETRIC KEY DISTRIBUTION USING ASYMMETRIC ENCRYPTION

• Once public keys have been distributed or have become accessible, secure communication that thwarts eavesdropping, tampering, or both, is possible.

• Public-key encryption provides for the distribution of secret keys to be used for conventional encryption.

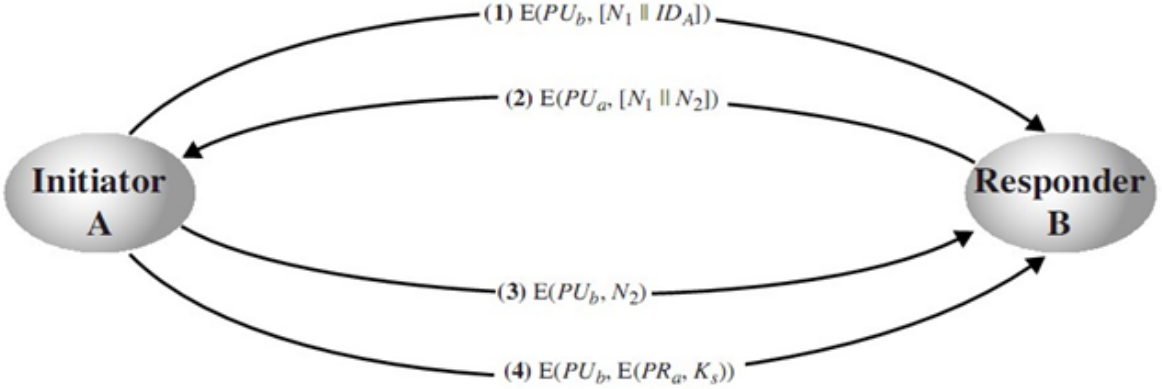
Simple Secret Key Distribution

- A generates a public/private key pair {PUa, PRa} and transmits a message to B consisting of PUa and an identifier of A, IDA
- B generates a secret key, Ks, and transmits it to A, encrypted with A's public key.
- 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.
- A discards PUa and PRa and B discards PUa.



Here third party can intercept messages and then either relay the intercepted message or substitute another message Such an attack is known as a man-in-the-middle attack.

Secret Key Distribution with Confidentiality and Authentication:



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

- B computes $D(PU_a, D(PR_b, M))$ to recover the secret key.

A Hybrid Scheme:

Yet another way to use public-key encryption to distribute secret keys is a hybrid approach.

- This scheme retains the use of a key distribution center (KDC) that shares a secret master key with each user and distributes secret session keys encrypted with the master key.
- A public key scheme is used to distribute the master keys.
- The addition of a public-key layer provides a secure, efficient means of distributing master keys.

2.2 DISTRIBUTION OF PUBLIC KEYS

There are four different schemes

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

Public announcement

Any participant can send his or her public key to any other participant or **broadcast** the key to the community.

Uncontrolled Public-Key Distribution

Limitatıons:

- 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.

- Authentication is needed to avoid this problem.

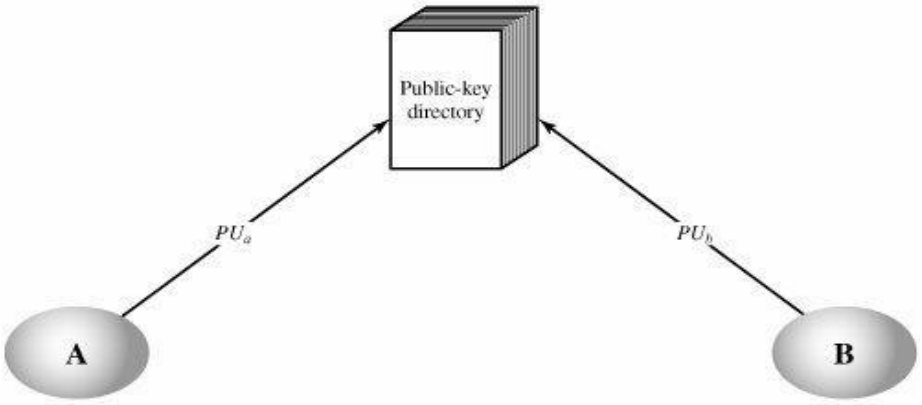
Publicly Available Directory

A greater degree of security can be achieved by maintaining a publicly available dynamic directory of public keys. Maintenance and distribution of the public directory would have to be the responsibility of some trusted entity or organization.

i)The authority maintains a directory with a {name, public key} entry for each participant.

ii)Each participant registers a public key with the directory authority.

iii) Participants could also access the directory electronically.

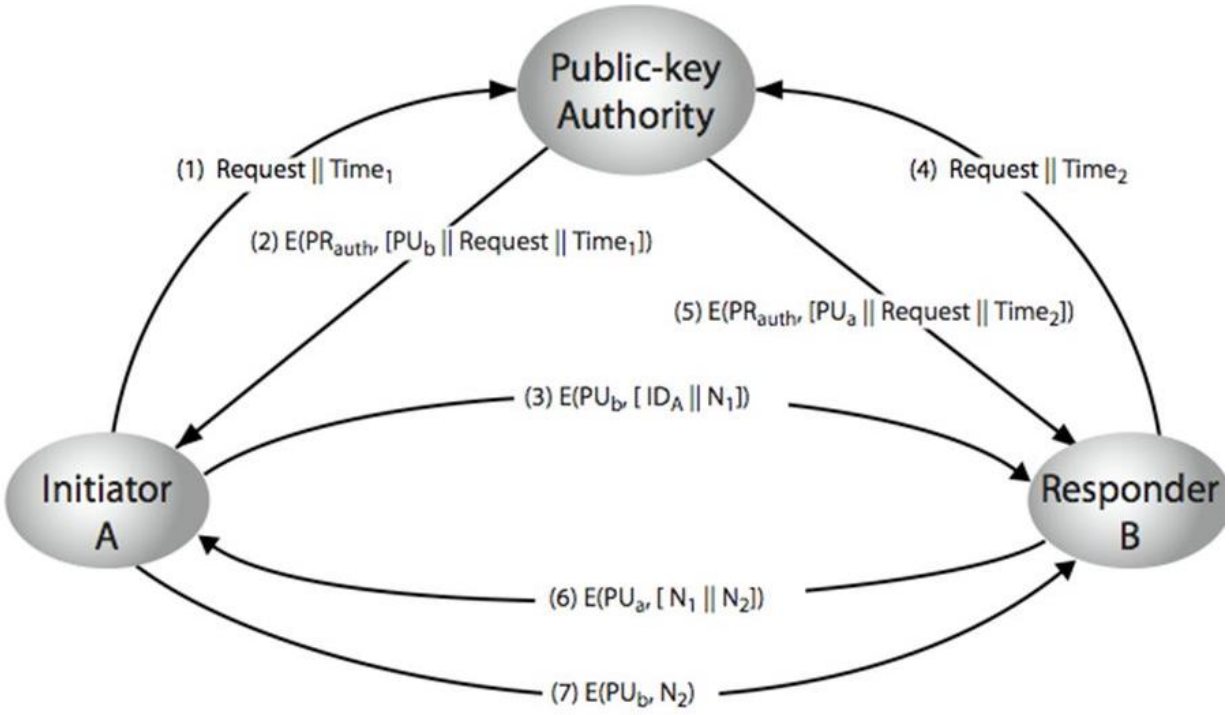


Limitations

- An Adversary may impersonate by stealing the private key of public key directory a n d falsely send the public key details.
- An attacker may attack the records stored in the directory.

Public key Authority

- Stronger security for public-key distribution can be achieved by providing tighter control over the distribution of public keys from the directory.
- Each participant reliably knows a public key for the authority, with only the authority knowing the corresponding private key.



i) A sends a timestamped message to the public-key authority containing a request for the current public key of B.

ii) The authority responds with a message that is encrypted using the authority's private key, PRauth

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, PUb which A can use to encrypt messages destined for B

● The original request, to enable A to match this response with the corresponding earlier request and to verify that the original request was not altered before reception by the authority

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

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

iv) B retrieves A's public key from the authority in the same manner as A retrieved B's public key. v)At this point, public keys have been securely delivered to A and B, and they may begin their protected exchange.

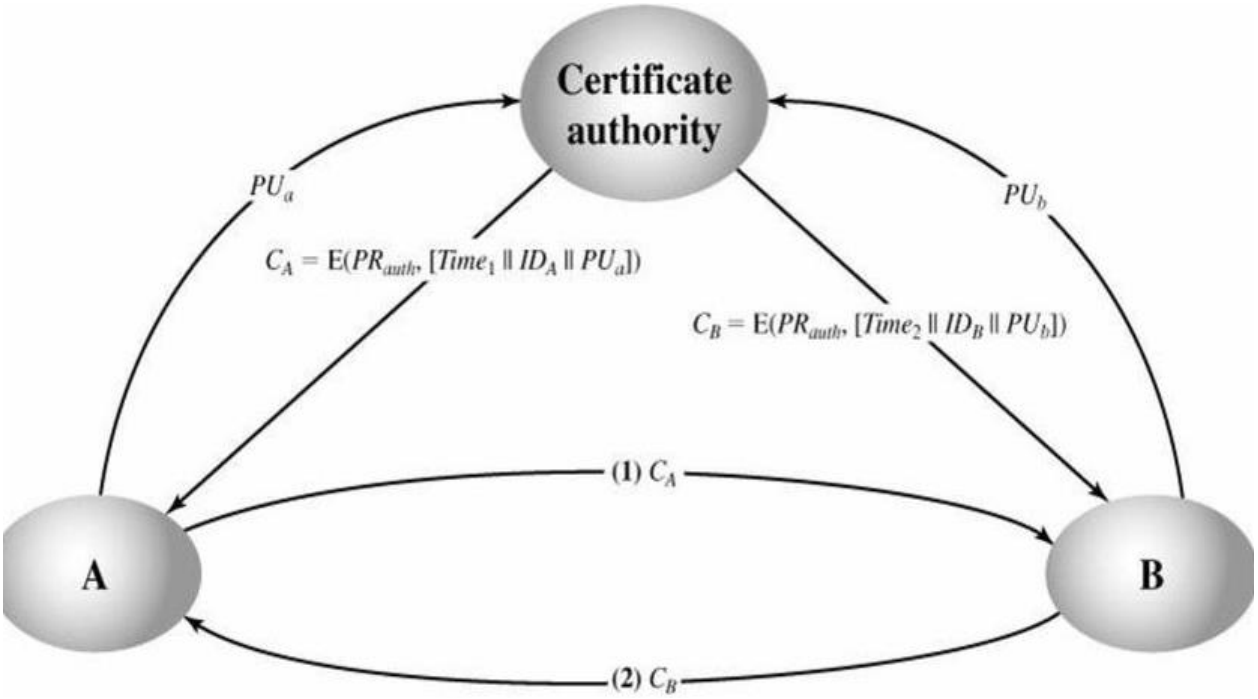
vi) B sends a message to A encrypted with PUa and containing A's nonce (N1) as well as a new nonce generated by B (N2) Because only B could have decrypted message (3), the presence of N1 in message (6) assures A that the correspondent is B.

vii) A returns N2, encrypted using B's public key, to assure B that its correspondent is A. Limitations:

- Bottleneck may occur in public authority.
- Tampering of records stored by the authority may take place.

Public key certificate

- A certificate consists of a public key plus an identifier of the key owner, with the whole block signed by a trusted third party.
- Typically, the third party is a certificate authority, such as a government agency or a financial institution, that is trusted by the user community.
- A user can present his or her public key to the authority in a secure manner, and obtain a certificate. The user can then publish the certificate.



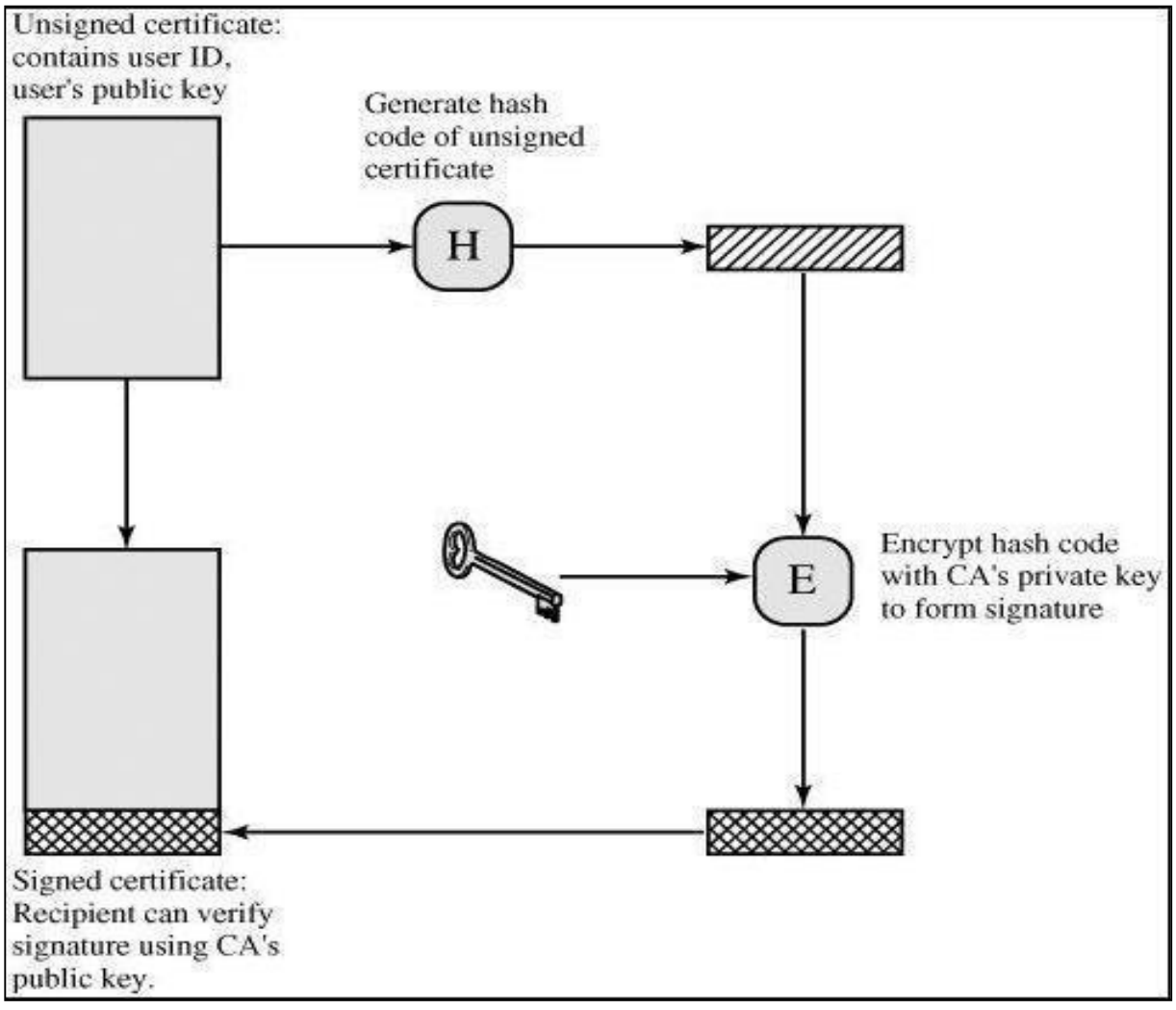
2.3 X.509 AUTHENTICATION SERVICE

ITU-T recommendation X.509 is part of the X.500 series of

recommendations that define a directory service. X.509 defines a framework for the provision of authentication services by the X.500 directory to its users. The directory may serve as a repository of public-key certificates of the type.

Each certificate contains the public key of a user and is signed with the private key of a trusted certification authority. X.509 certificate format is used in S/MIME, IP Security, and SSL/TLS and SET.

X.509 is based on the use of public-key cryptography and digital signature algorithms. Figure illustrates the generation of public key.



Certificates

Figure shows the general format of a certificate, which includes the following elements:

Version:

Differentiates among successive versions of the certificate format; the default is version 1.

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

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.

Signature: This field includes the signature algorithm identifier.

The standard uses the following notation to define a certificate:

CA<<A>> = CA {V, SN, AI, CA, TA, A, Ap}

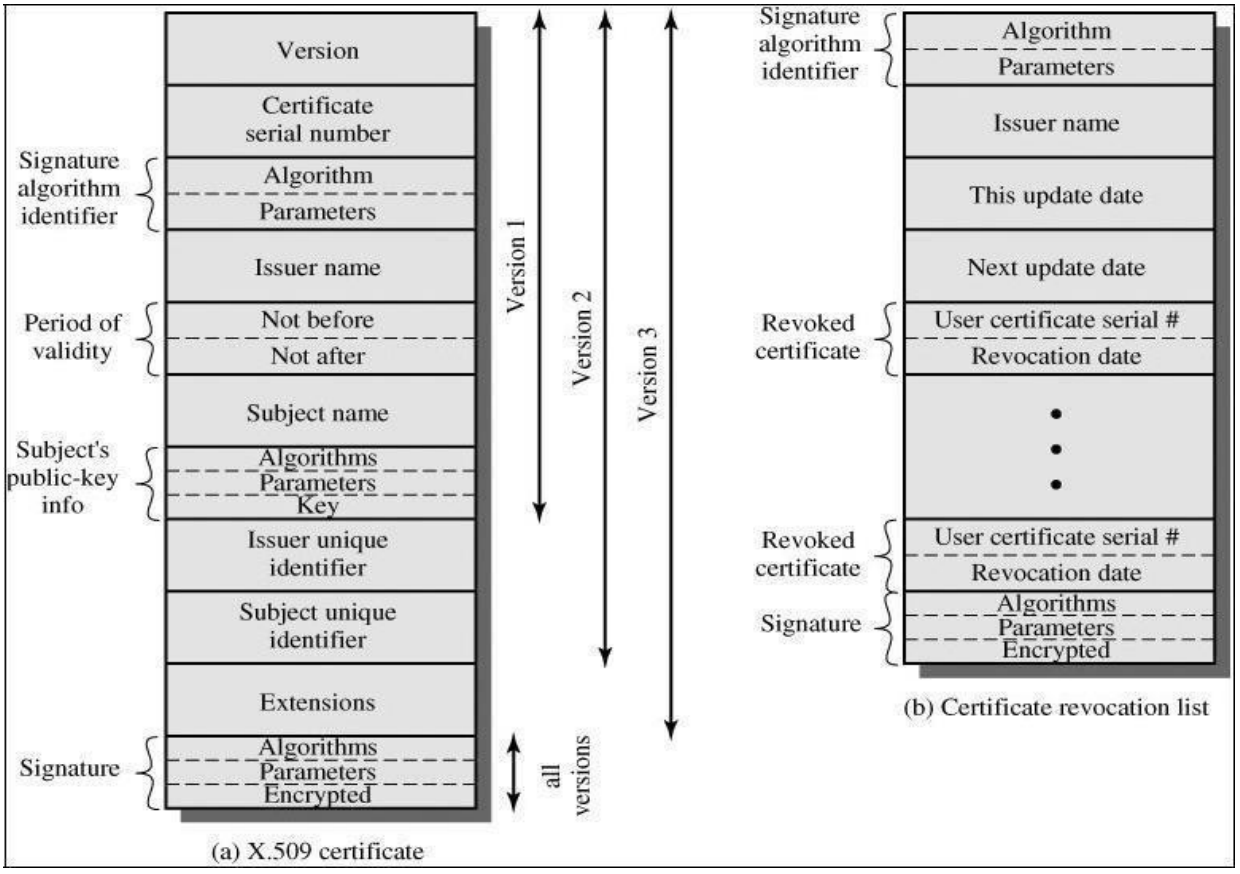
The CA signs the certificate with its private key. If the corresponding public key is known to a user, then that user can verify that a certificate signed by the CA is valid.

Obtaining a 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. Because certificates are unforgeable, they can be placed in a directory without the need for the directory to make special efforts to protect them.



Certificate Revocation

Certificates have a period of validity.

May need to revoke before expiry, eg:

- User's private key is compromised
- User is no longer certified by this CA
- CA's certificate is compromised

CA maintain a list consisting of all revoked but not expired certificates issued by that CA, including both those issued to users and to other CAs. Each certificate revocation list (CRL) posted to the directory is signed by the issuer. When a user receives a certificate in a message, the user must determine whether the certificate has been revoked. The user could check the directory each time a certificate is received.

2.4 PUBLIC-KEY INFRASTRUCTURE (PKI)

- RFC 2822 (Internet Security Glossary) defines public-key infrastructure (PKI) as the set of hardware, software, people, policies, and procedures needed to create, manage, store, distribute, and revoke digital certificates based on asymmetric cryptography.
- The principal objective for developing a PKI is to enable secure, convenient, and efficient acquisition of public keys. The Internet Engineering Task Force (IETF) Public Key Infrastructure X.509 (PKIX) working group has been the driving force behind setting up a formal (and generic) model based on X.509 that is suitable for deploying a certificate-based architecture on the Internet. This section describes the PKIX model.

Key elements of the PKIX model. These elements are

End entity:

A generic term used to denote end users, devices (e.g., servers, routers), or any other entity that can be identified in the subject field of a public key certificate. End entities typically consume and/or support PKI-related services.

Certification authority (CA):

The issuer of certificates and (usually) certificate revocation lists (CRLs). It may also support a variety of administrative functions, although these are often delegated to one or more Registration Authorities.

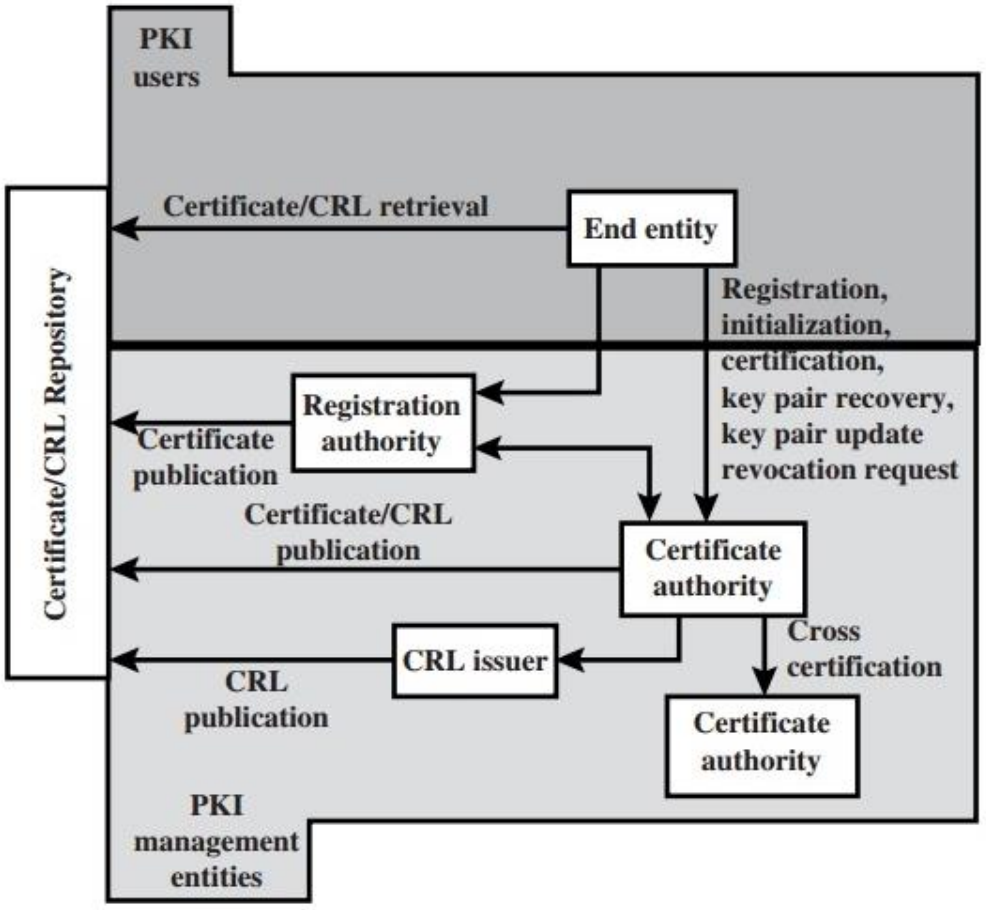
Registration authority (RA): An optional component that can assume a number of ad ministrative functions from the CA. The RA is often associated with the end entity regi stration process but can assist in a number of other areas as well.

CRL issuer: An optional component that a CA can delegate to publish CRLs.

Repository: A generic term used to denote any method for storing certificates and CRLs so that they can be retrieved by end entities.

PKIX Management Functions

PKIX identifies a number of management functions that potentially need to be supported by management protocols. These are indicated in Figure and include the following:



PKIX Architectural Model

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. Registration begins the process of enrolling in a PKI.
- Registration usually involves some offline or online procedure for mutual authentication. Typically, the end entity is issued one or more shared secret keys used for subsequent authentication.

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. For example, the client needs to be securely initialized with the public key and other assured information of the trusted CA(s), to be used in validating certificate paths.

Certification: This is the process in which a CA issues a certificate for a user's public key, returns that certificate to the user's client system, and/or posts that certificate in a repository.

Key pair recovery:

- Key pairs can be used to support digital signature creation and verification, encryption and decryption, or both. When a key pair is used for encryption/decryption, it is important to provide a mechanism to recover the necessary decryption keys when normal access to the keying material is no longer possible, otherwise it will not be possible to recover the encrypted data.
- Loss of access to the decryption key can result from forgotten passwords/PINs, corrupted disk drives, damage to hardware tokens, and so on. Key pair recovery allows end entities to restore their encryption/decryption key pair from an authorized key backup facility (typically, the CA that issued the end entity's certificate).

Key pair update: All key pairs need to be updated regularly (i.e., replaced with a new key pair) and new certificates issued. Update is required when the cer- tificate lifetime expires and as a result of certificate revocation.

Revocation request: An authorized person advises a CA of an abnormal situa- tion requiring certificate revocation. Reasons for revocation include private- key compromise, change in affiliation, and name change.

Cross certification: Two CAs exchange information used in establishing a cross-certificate. A cross-certificate is a certificate issued by one CA to another CA that contains a CA signature key used for issuing certificates.

PKIX Management Protocols

- The PKIX working group has defined two alternative management protocols between PKIX entities that support the management functions listed in the preceding subsection.
- RFC 2510 defines the certificate management protocols (CMP). Within CMP, each of the management functions is explicitly identified by specific protocol

exchanges. CMP is designed to be a flexible protocol able to accommodate a variety of technical, operational, and business models.

- RFC 2797 defines certificate management messages over CMS (CMC), where CMS refers to RFC 2630, cryptographic message syntax.
- CMC is built on earlier work and is intended to leverage existing implementations. Although all of the PKIX func- tions are supported, the functions do not all map into specific protocol exchanges.

2.5 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. RFC 2828 defines user authentication.

REMOTE USER-AUTHENTICATION PRINCIPLES

The process of verifying an identity claimed by or for a system entity.

An authentication process consists of two steps:

Identification step: Presenting an identifier to the security system.

Verification step: Presenting or generating authentication information that corroborates the binding between the entity and the identifier.

There are four general means of authenticating a user's identity, which can be used alone or in combination:

Something the individual knows: Examples include a password, a personal identification number (PIN), or answers to a prearranged set of questions.

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.

Something the individual is (static biometrics): Examples include recognition by fingerprint, retina, and face.

Something the individual does (dynamic biometrics): Examples include recognition by voice pattern, handwriting characteristics, and typing rhythm.

For network-based user authentication, the most important methods involve cryptographic keys and some- thing the individual knows, such as a password.

Mutual Authentication

An important application area is that of mutual authentication protocols. Such protocols enable communicating parties to satisfy themselves mutually about each other's identity and to exchange session keys.

Central to the problem of authenticated key exchange are two issues:

Confidentiality and timeliness. To prevent masquerade and to prevent compromise of session keys, essential identification and session-key information must be communicated in encrypted form. The second issue, timeliness, is important because of the threat of message replays.

Examples of replay attacks:

Simple replay: The opponent simply copies a message and replays it later.

Repetition that can be logged: An opponent can replay a timestamped message within the valid time window.

Repetition that cannot be detected: This situation could arise because the original message could have been suppressed and thus did not arrive at its destination; only the replay message arrives.

Backward replay without modification: This is a replay back to the message sender. This attack is possible if symmetric encryption is used and the sender cannot easily recognize the difference between messages sent and messages received on the basis of content.

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.

One-Way Authentication

- One application for which encryption is growing in popularity is electronic mail (e-mail). The very nature of electronic mail, and its chief benefit, is that it is not necessary for the sender and receiver to be online at the same time.
- Instead, the e-mail message is forwarded to the receiver's electronic mailbox, where it is buffered until the receiver is available to read it.
- The "envelope" or header of the e-mail message must be in the clear, so that the message can be handled by the store-and-forward e-mail protocol, such as the Simple Mail Transfer Protocol (SMTP) or X.400.
- However, it is often desirable that the mail-handling protocol not require access to the plaintext form of the message, because that would require trusting the mail-handling mechanism.
- Accordingly, the e-mail message should be encrypted such that the mail-handling system is not in possession of the decryption key.
- A second requirement is that of authentication. Typically, the recipient wants some assurance that the message is from the alleged sender.

2.6 Remote User-Authentication Using Symmetric Encryption

Mutual Authentication

- A two-level hierarchy of symmetric keys can be used to provide confidentiality for communication in a distributed environment
- Strategy involves the use of a trusted key distribution center (KDC)
- Each party shares a secret key, known as a master key, with the KDC
- KDC is responsible for generating keys(Session keys) to be used for a short time over a connection between two parties and for distributing those keys using the master keys to protect the distribution
- Example: Kerberos

Needham and Schroeder protocol

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

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.

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

Suppress-Replay Attacks

- The Denning protocol requires reliance on clocks that are synchronized throughout the network
- A risk involved is based on the fact that the distributed clocks can become unsynchronized as a result of interruption on or faults in the clocks or the synchronization mechanism
- The problem occurs when a sender's clock is fast of the intended recipient's clock
    - 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
    - Such attacks are referred to as *suppress-replay attacks*

Countermeasures

- Parties regularly check their clocks against the KDC's clock
- To rely on handshaking protocols using nonces

One-Way Authentication

Using symmetric encryption, the decentralized key distribution scenario illustrated is impractical. This scheme requires the sender to issue a request to the intended recipient, await a response that includes a session key, and only then send the message.

With some refinement, the KDC strategy is a candidate for encrypted electronic mail. Because we wish to avoid requiring that the recipient (B) be on line at the same time as the sender (A), steps 4 and 5 must be eliminated.

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

2.7 KERBEROS

- Kerberos is an authentication service developed by MIT and is one of the best known and most widely implemented **trusted third party** key distribution systems.
- Provides a centralized authentication server whose function is to authenticate users to servers and servers to users.
- Kerberos relies exclusively on symmetric encryption, making no use of public-key encryption.

Kerberos Requirements

Secure: A network eavesdropper should not be able to obtain the necessary information to impersonate a user.

Reliable: Kerberos should be highly reliable and should employ a distributed server architecture, with one system able to back up another.

Transparent: The user should not be aware that authentication is taking place, beyond the requirement to enter a password.

Scalable: The system should be capable of supporting large numbers of clients and servers. This suggests a modular, distributed architecture.

Kerberos is a basic third-party authentication scheme.

Authentication Server (AS)

- Knows the passwords of all users and stores these in a centralized database.
- AS shares a unique secret key with each server.
- These keys have been distributed physically or in some other secure manner
- users initially negotiate with AS to identify self
- AS provides a non-corruptible authentication credential (ticket granting ticket TGT)

Ticket Granting server (TGS)

- issues tickets to users who have been authenticated to AS
- users subsequently request access to other services from TGS on basis of users TGT

<u>Simple Authentication Dialogue</u>

$$\begin{aligned}
&\textbf{(1)}\ \mathrm{C} \rightarrow \mathrm{AS}: \quad ID_C \| P_C \| ID_V \\
&\textbf{(2)}\ \mathrm{AS} \rightarrow \mathrm{C}: \quad Ticket \\
&\textbf{(3)}\ \mathrm{C} \rightarrow \mathrm{V}: \quad ID_C \| Ticket \\
&Ticket = E(K_v, [ID_C \| AD_C \| ID_V])
\end{aligned}$$

where

C = client
AS = authentication server
V = server
ID_C = identifier of user on C
ID_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

Drawback of simple authentication dialogue

- The password Pc is transmitted as a simple plain text. So, there is a possibility of capturing by the attacker.

More secure authentication Dialogue

Once per user logon session:

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

(2) $AS \rightarrow C: \quad E(K_c, Ticket_{tgs})$

Once per type of service:

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

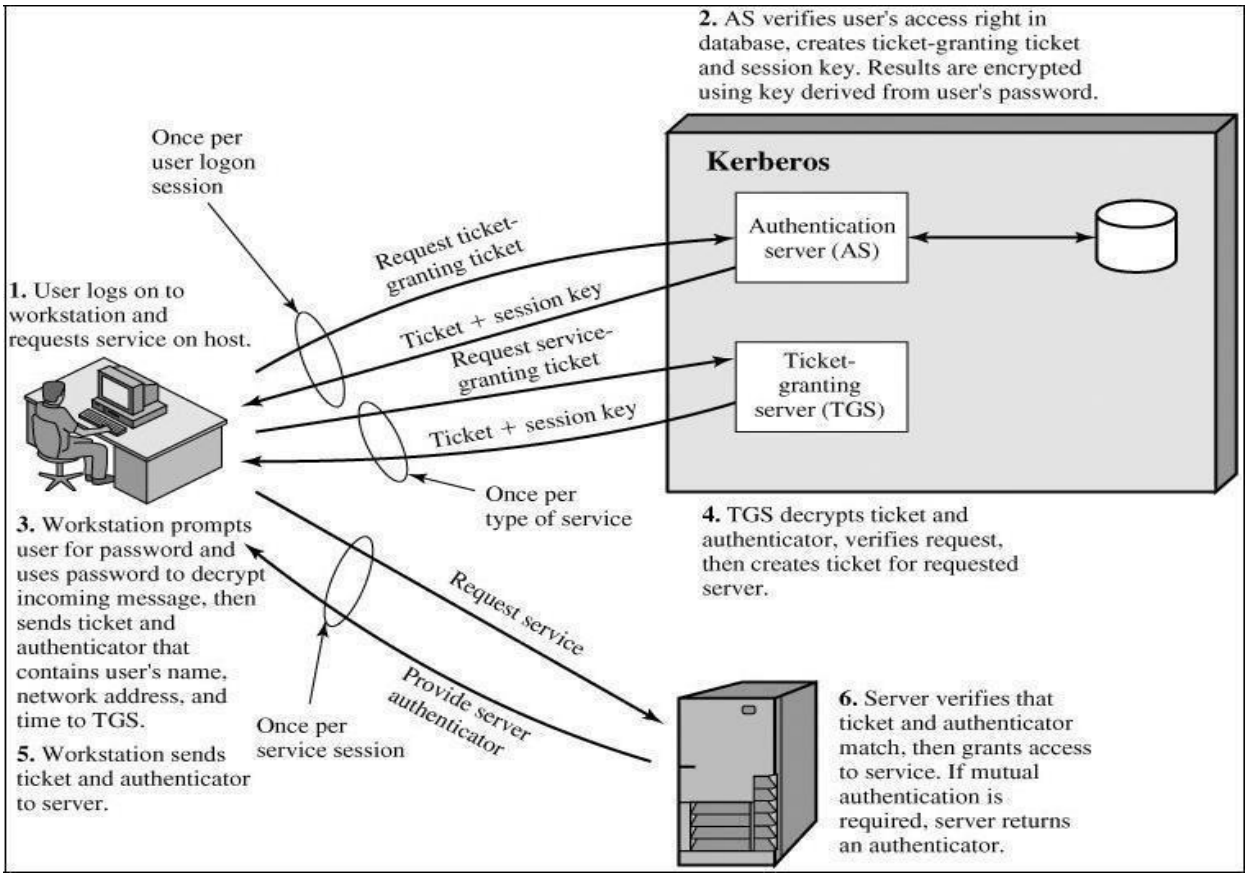
(4) $TGS \rightarrow C: \quad Ticket_v$

Once per service session:

(5) $C \rightarrow V: \quad ID_C \| Ticket_v$

$$Ticket_{tgs} = E(K_{tgs}, [ID_C \| AD_C \| ID_{tgs} \| TS_1 \| Lifetime_1])$$
$$Ticket_v = E(K_v, [ID_C \| AD_C \| ID_v \| TS_2 \| Lifetime_2])$$

Summary of Kerberos Version 4 Message Exchanges



Overview of Kerberos

- Client sends a message to the AS requesting access to the TGS.
- AS responds with a message, encrypted with a key derived from the user's password (Kc) that contains the ticket.

- Encrypted message also contains a copy of the session key, $K_{C,tgs}$, where the subscripts indicate that this is a session key for C and TGS.
- Session key is inside the message encrypted with K_C, only the user's client can read it.
- Same session key is included in the ticket, which can be read only by the TGS.
- Thus, the session key has been securely delivered to both C and the TGS.
- Message (1) includes a timestamp, so that the AS knows that the message is timely.
- Message (2) includes several elements of the ticket in a form accessible to C. This enables C to confirm that this ticket is for the T

THE VERSION 4 AUTHENTICATION DIALOGUE

Table 15.1 Summary of Kerberos Version 4 Message Exchanges

(1) $C \rightarrow AS \quad ID_c \| ID_{tgs} \| TS_1$

(2) $AS \rightarrow C \quad E(K_c, [K_{c,tgs} \| ID_{tgs} \| TS_2 \| Lifetime_2 \| Ticket_{tgs}])$

$$Ticket_{tgs} = E(K_{tgs}, [K_{c,tgs} \| ID_C \| AD_C \| ID_{tgs} \| TS_2 \| Lifetime_2])$$

(a) Authentication Service Exchange to obtain ticket-granting ticket

(3) $C \rightarrow TGS \quad ID_v \| Ticket_{tgs} \| Authenticator_c$

(4) $TGS \rightarrow C \quad E(K_{c,tgs}, [K_{c,v} \| ID_v \| TS_4 \| Ticket_v])$

$$Ticket_{tgs} = E(K_{tgs}, [K_{c,tgs} \| ID_C \| AD_C \| ID_{tgs} \| TS_2 \| Lifetime_2])$$

$$Ticket_v = E(K_v, [K_{c,v} \| ID_C \| AD_C \| ID_v \| TS_4 \| Lifetime_4])$$

$$Authenticator_c = E(K_{c,tgs}, [ID_C \| AD_C \| TS_3])$$

(b) Ticket-Granting Service Exchange to obtain service-granting ticket

(5) $C \rightarrow V \quad Ticket_v \| Authenticator_c$

(6) $V \rightarrow C \quad E(K_{c,v}, [TS_5 + 1])$ (for mutual authentication)

$$Ticket_v = E(K_v, [K_{c,v} \| ID_C \| AD_C \| ID_v \| TS_4 \| Lifetime_4])$$

$$Authenticator_c = E(K_{c,v}, [ID_C \| AD_C \| TS_5])$$

(c) Client/Server Authentication Exchange to obtain service

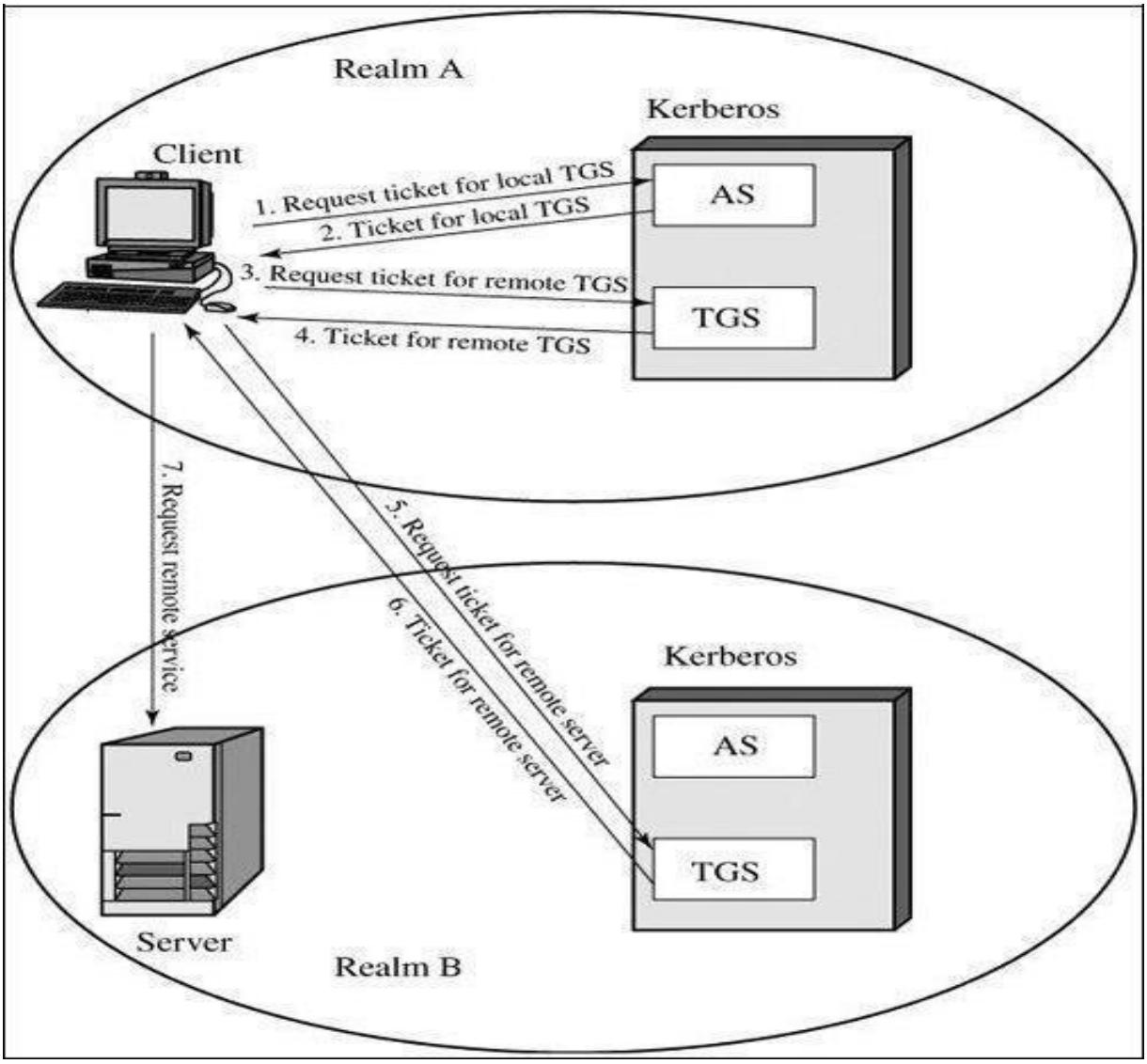
Kerberos Realms

Kerberos environment consisting of a Kerberos server, a number of clients, and a number of application servers requires the following:

1. The Kerberos server must have the user ID and hashed passwords of all participating users in its 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 with the Kerberos server.
3. The Kerberos server in each interoperating realm shares a secret key with the server in the other realm. The two Kerberos servers are registered with each other.



Such an environment is referred to as a **Kerberos realm**. The concept of *realm* can be explained as follows. A Kerberos realm is a set of managed nodes that share the same Kerberos database.

Kerberos principal, which is a service or user that is known to the Kerberos system. 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

A user wishing service on a server in another realm needs a ticket for that server. The user's client follows the usual procedures to gain access to the local TGS and then requests a ticket-granting ticket for a remote TGS (TGS in another realm). The client can then apply to the remote TGS for a service-granting ticket for the desired server in the realm of the remote TGS.

DIFFERENCES BETWEEN VERSIONS 4 AND 5

Version 5 is intended to address the limitations of version 4 in two areas: environmental shortcomings and technical deficiencies. Let us briefly summarize the improvements in each area.

1. Internet protocol dependence: Version 4 requires the use of Internet Protocol (IP) addresses. Version 5 network addresses are tagged with type and length, allowing any network address type to be used.

2.Message byte ordering: In version 4, the sender of a message employs a byte ordering of its own choosing and tags the message to indicate least significant byte in lowest address or most significant byte in lowest address. In version 5, all message structures are defined using Abstract Syntax Notation One (ASN.1) and Basic Encoding Rules (BER), which provide an unambiguous byte ordering.

3. Ticket lifetime: Lifetime values in version 4 are encoded in an 8-bit quantity in units of five minutes. Thus, the maximum lifetime that can be expressed is $28 * 5 = 1280$ minutes (a little over 21 hours). In version 5, tickets include an explicit start time and end time, allowing tickets with arbitrary lifetimes.

4. Authentication forwarding: Version 4 does not allow credentials issued to one client to be forwarded to some other host and used by some other client. Version 5 provides this capability.

5. Inter-realm authentication: In version 4, interoperability among N realms requires on the order of N 2 Kerberos-to-Kerberos relationships, as described earlier. Version 5 supports a method that requires fewer relationships, as described shortly.

Table 15.3 Summary of Kerberos Version 5 Message Exchanges

(1)	C → AS	$\text{Options} \| ID_c \| Realm_c \| ID_{tgs} \| Times \| Nonce_1$
(2)	AS → C	$Realm_C \| ID_C \| Ticket_{tgs} \| \text{E}(K_c, [K_{c,tgs} \| Times \| Nonce_1 \| Realm_{tgs} \| ID_{tgs}])$
		$Ticket_{tgs} = \text{E}(K_{tgs}, [Flags \| K_{c,tgs} \| Realm_c \| ID_C \| AD_C \| Times])$

(a) Authentication Service Exchange to obtain ticket-granting ticket

(3)	C → TGS	$\text{Options} \| ID_v \| Times \| Nonce_2 \| Ticket_{tgs} \| Authenticator_c$
(4)	TGS → C	$Realm_c \| ID_C \| Ticket_v \| \text{E}(K_{c,tgs}, [K_{c,v} \| Times \| Nonce_2 \| Realm_v \| ID_v])$
		$Ticket_{tgs} = \text{E}(K_{tgs}, [Flags \| K_{c,tgs} \| Realm_c \| ID_C \| AD_C \| Times])$
		$Ticket_v = \text{E}(K_v, [\text{Flags} \| K_{c,v} \| \text{Realm}_c \| \text{ID}_C \| \text{AD}_C \| \text{Times}])$
		$Authenticator_c = \text{E}(K_{c,tgs}, [ID_C \| Realm_c \| TS_1])$

(b) Ticket-Granting Service Exchange to obtain service-granting ticket

(5)	C → V	$\text{Options} \| \text{Ticket}_v \| \text{Authenticator}_c$
(6)	V → C	$E_{K_{c,v}}[\text{TS}_2 \| \text{Subkey} \| \text{Seq}\neq]$
		$\text{Ticket}_v = \text{E}(K_v, [\text{Flag} \| K_{c,v} \| \text{Realm}_c \| \text{ID}_C \| \text{AD}_C \| \text{Times}])$
		$Authenticator_c = \text{E}(K_{c,v}, [ID_C \| Relam_c \| TS_2 \| Subkey \| Seq\neq])$

(c) Client/Server Authentication Exchange to obtain service

2.8 Remote User Authentication Using Asymmetric Encryption

Mutual Authentication

- Public-key encryption for session key distribution
  - Assumes each of the two parties is in possession of the current public key of the other
  - May not be practical to require this assumption
- Denning protocol using timestamps
  - Uses an authentication server (AS) to provide public-key certificates
  - Requires the synchronization of clocks

1. $\text{A} \rightarrow \text{AS}: \quad ID_A \| ID_B$
2. $\text{AS} \rightarrow \text{A}: \quad \text{E}(PR_{as}, [ID_A \| PU_a \| T]) \| \text{E}(PR_{as}, [ID_B \| PU_b \| T])$
3. $\text{A} \rightarrow \text{B}: \quad \text{E}(PR_{as}, [ID_A \| PU_a \| T]) \| \text{E}(PR_{as}, [ID_B \| PU_b \| T]) \| \text{E}(PU_b, \text{E}(PR_a, [K_s \| T]))$

- Woo and Lam makes use of nonces

Care needed to ensure no protocol flaws

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

One-Way Authentication

- Have public-key approaches for e-mail
    - Encryption of message for confidentiality, authentication, or both
    - The public-key algorithm must be applied once or twice to what may be a long message
- For confidentiality encrypt message with one-time secret key, public-key encrypted
- If authentication is the primary concern, a digital signature may suffice