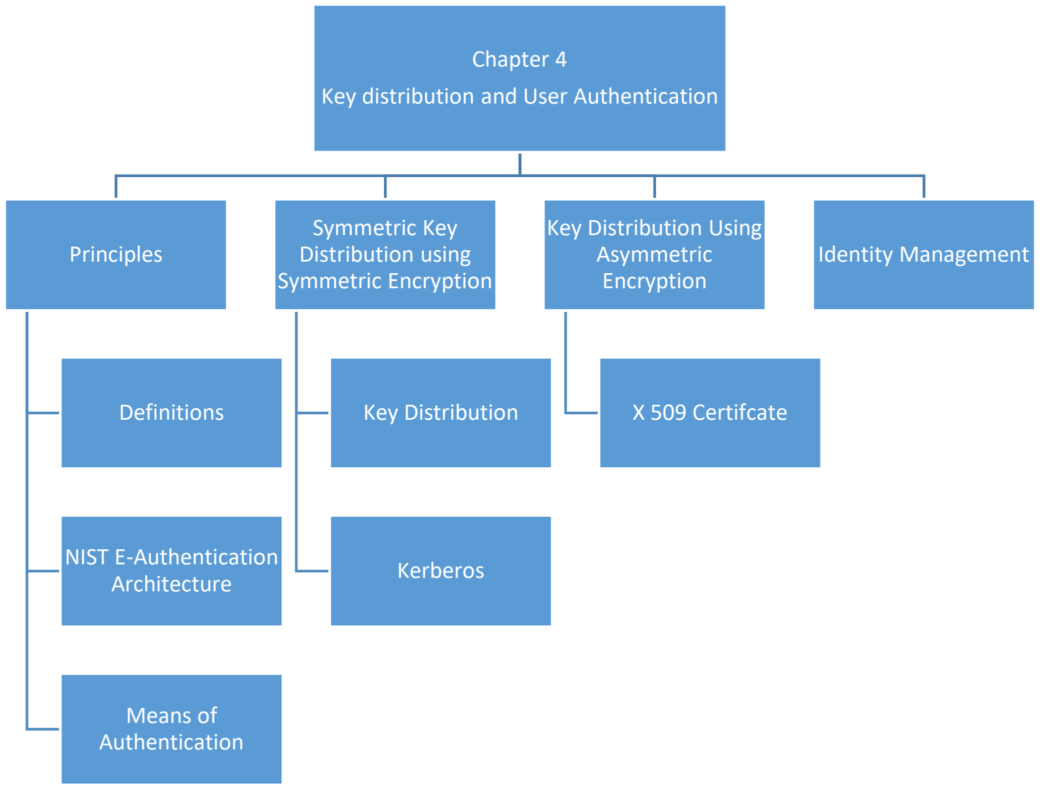
Chapter 4

Key Distribution and User Authentication

Prepared by: Ahmed Badawy



Chapter 4
Key distribution and User Authentication
Principles
Symmetric Key Distribution using Symmetric Encryption
Key Distribution Using Asymmetric Encryption
Identity Management
Definitions
NIST E-Authentication Architecture
Means of Authentication
Key Distribution
Kerberos
X 509 Certifcate

Principles

- **User authentication is**
  - the fundamental building block and the primary line of defense
  - the basis for most types of access control and for user accountability
  - **RFC 4949 definition:**
    - User authentication has two steps
      - Identification step
        - Presenting an identifier to the security system
        - Means by which a user provides a claimed identity to the system
      - Verification step
        - Presenting or generating authentication information that corroborates the binding between the entity and the identifier.
  - **NIST SP 800 definition:**
    - The process of establishing confidence in the user identities that are presented electronically to an information system.
    - Which identity is presented has to do with the required **confidence level**; creating an account with an email provider is different from creating an account with a bank to access online banking.
- **NIST E-User Authentication Model**
  - In the figure below,
    - the objective is for a user to access resources provided through the relying party
    - However, this should be done in a secure way and the relying party (RP) must have a way of confirming the identity of the user for access control
  - Sequence for registration

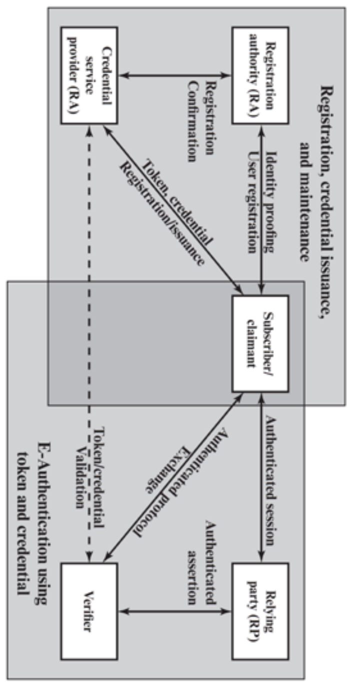
1. Registration, credential issuance and maintenance
2. E-Authentication using token and credential

1. <u>Registration, credential issuance and maintenance</u>

- First step that must be applied before authentication and accessing resources step
- Registration Authority (RA) is a trusted entity that establishes and vouches for the identity of an applicant to the credential service provider (CSP)
    1. Applicant applies to RA to become a subscriber of CSP by providing identity proofing and registration.
    2. RA then communicates with CSP to confirm the registration of the user
    3. The CSP issues credentials, token and/or an encryption key to the subscriber.
- In this step, the status of the user changes from **applicant → subscriber**

2. <u>E-Authentication using token and credential</u>

- In this step, the status of the user changes from **subscriber → claimant**
    4. The claimant demonstrates possession of token to the Verifier
    5. The verifier confirms that the claimant is indeed a subscriber → verifier then communicates with CSP to confirm token or credential. Verifier and CSP can be part of the same system.
    6. Verifier passes on an assertion about the identity of the subscriber to the RP.
    7. The RP makes access control or authorized decision. The user now have an authenticated session with the RP.



- **<u>Means of authentication</u>**

- 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

---

<u>Symmetric Key Distribution Using Symmetric Encryption</u>

- Symmetric encryption implies that the two communicating parties must have shared the same key and that the key must be protected from access by others.
- Frequent key exchange is desirable to limit the amount of data compromised if an attacker learns the key.
- **Strength of cryptographic systems rests with the key distribution technique.**
- Options of key distribution:
    - Manual:
        - A key can be generated by either of the two parties A or B and delivered physically.
        - Or the key can be generated by a third party and delivered physically to the two parties.
    - If A and B have previously shared a key, one party could transmit the new key encrypted using the old key.
    - Through a key distribution center (KDC).

- The KDC has a permanent key with A and B, K_{AC} and K_{BC}.
- The KDC generated a session key to be used for encrypting the data between A and B, K_{AB}.
- The KDC encrypts K_{AB} using K_{AC} and sends it to A.
- The KDC encrypts K_{AB} using K_{BC} and sends it to B.

Kerberos

Motivation:

- Assume a distributed environment in which users at workstations wish to access services on servers distributed throughout the network. Servers then need to restrict access to authorized users and to be able to authenticate requests for service. In this environment, a workstation cannot be trusted to identify its users correctly to network services, because:
  - A user may gain access to a particular workstation and pretend to be another user operating from that workstation.
  - A user may alter the network address of a workstation so that the requests sent from the altered workstation appear to come from the impersonated workstation.
  - A user may eavesdrop on exchanges and use a replay attack to gain entrance to a server or to disrupt operations
  - In any of these cases, an unauthorized user may be able to gain access to services and data that he or she is not authorized to access.

Hence, rather than building elaborate authentication protocols at each server, there was a need for a centralized authentication server whose function is to authenticate users to servers and servers to users. Kerberos fills in this gap.

What is Kerberos?

- A centralized key distribution and authentication service developed at MIT
- Relies exclusively on symmetric encryption.
- Two common versions; version 4 and version 5.
- Version 4 uses DES.

What are the objectives of Kerberos? TWO objectives

- User authentication and Key distribution

How many servers are part of Kerberos infrastructure?
TWO Servers:

- **Authentication Server (AS)**
  - Users negotiate with AS to identify self
  - AS provides TGT to the user
- **Ticket Granting Server (TGS)**
  - The user takes the TGT and sends it to TGS
  - The user requests access to other services from TGS.

What are the prerequisites of Kerberos? TWO

- All users are registered with the Kerberos server
- All servers are registered with the Kerberos server.

What are the types of tickets used in Kerberos? TWO

- **Ticket Granting Ticket (TGT) $\boldsymbol{Ticket}_{tgt}$**
  - Provided from AS to Client (C)
  - Allows C to start communicating with TGS
  - Encrypted with permanent key K_{tgs}
- **Ticket ($\boldsymbol{Ticket}_{v}$):**
  - Provided from TGS to C
  - Allows C to access server V
  - Encrypted with permanent key K_v

What types of keys are used in Kerberos? TWO

- **Permanent Keys**
  - K_c between Client and AS → **used in the authentication step (First Objective)**
  - K_{tgs} between AS and TGS→ used to encrypt $\boldsymbol{Ticket}_{tgt}$
  - K_v between TGS and V → used to encrypt $\boldsymbol{Ticket}_{v}$
- **Session Keys**
  - $K_{c,tgs}$ between Client and TGS → used to encrypt communication in the intermediate step between C and TGS
  - $K_{c,v}$ between Client and TGS → **this is the symmetric session key that will be used for encrypting the data between the client and the server (V) (Second Objective)**

How is the authentication step applied in Kerberos?

- It is assumed that users have already registered with Kerberos servers.

Why do we need to encrypt the ticket?

- To prevent alteration or forgery of the content of the ticket.

- Kerberos servers have a hashed version of the user's password (obtained earlier from the registration step).
- In the authentication step, the client does not send his password.
- The AS sends the client a message (challenge) encrypted with a key (K_c) derived from the client's password.
- If the client is indeed the person he claims to be, he must be able to generate $K_c$from his end, decrypt the message sent from the AS, and retrieve the information inside.

<u>What is a motivation behind the use of tickets?</u>

- To minimize the number of time a user has to enter a password.
- The tickets are reusable during their lifetime.
- The user's workstation can save the server ticket and uses it on behalf of the user for multiple accesses to the mail server.

- Note the ticket is encrypted with a permanent key known only to infrastructure servers (AS, TGS, and Server) and not known to users. Even legitimate users cannot alter the content of the ticket.

<u>Why do we use timestamps in Kerberos?</u>

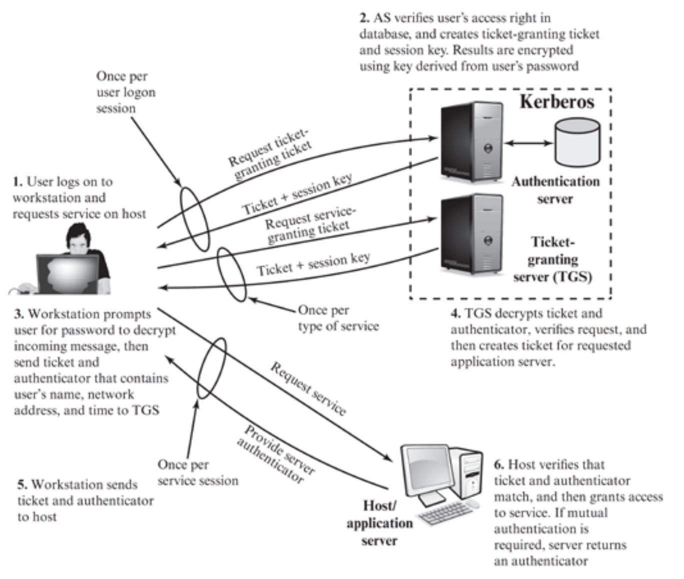
- To prevent a reply attack: an opponent could capture the TGT and wait until the legitimate user logs off and then use the TGS to spoof the TGS.

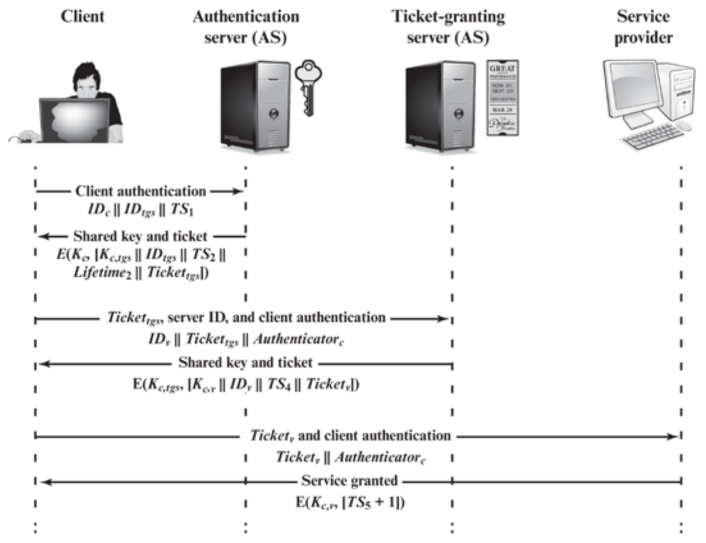
<u>What is an authenticator and what is it used for?</u>

- An identifier that contains user's name, network address and time timestamp.
- It is encrypted using a session key.

<u>How is the session key shared between the AS and TGS and then TGS and V?</u>

- The session key is sent twice, once inside the ticket (which is encrypted using a permanent key) and another outside the ticket (which is encrypted using a session key)


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
Request service-granting ticket
Ticket + session key
Authentication server
Ticket-granting server (TGS)
1. User logs on to workstation and requests service on host
Once per type of service
3. Workstation prompts user for password to decrypt incoming message, 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


Client
Authentication server (AS)
Ticket-granting server (AS)
Service provider
Client authentication
$ID_c \| ID_{tgs} \| TS_1$
Shared key and ticket
$E(K_c, [K_{c,tgs} \| ID_{tgs} \| TS_2 \| Lifetime_2 \| Ticket_{tgs}])$
$Ticket_{tgs}$, server ID, and client authentication
$ID_v \| Ticket_{tgs} \| Authenticator_c$
Shared key and ticket
$E(K_{c,tgs}, [K_{c,v} \| ID_v \| TS_4 \| Ticket_v])$
$Ticket_v$ and client authentication
$Ticket_v \| Authenticator_c$
Service granted
$E(K_{c,v}, [TS_5 + 1])$

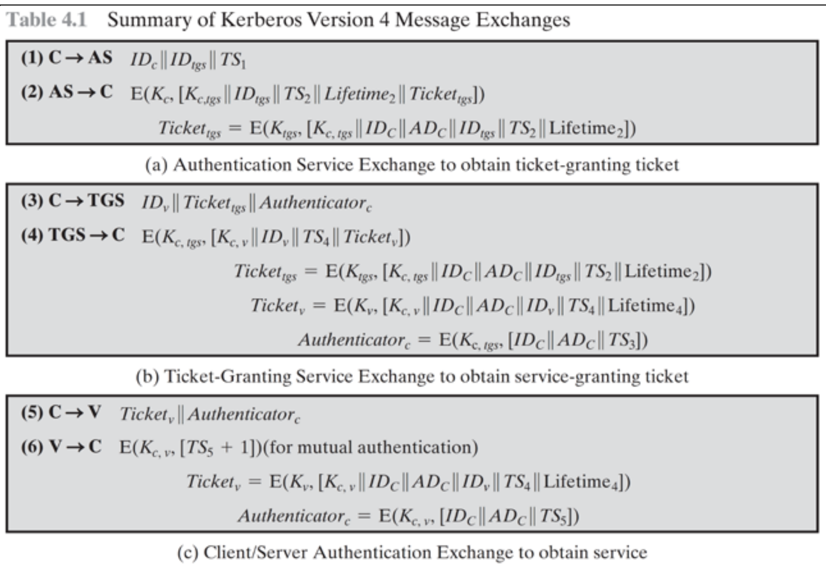


Table 4.1 Summary of Kerberos Version 4 Message Exchanges

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

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

$$Ticket_{tgs} = E(K_{tgs}, [K_{c,tgs} \| ID_C \| AD_C \| ID_{tgs} \| TS_2 \| \text{Lifetime}_2])$$

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

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

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

$$Ticket_{tgs} = E(K_{tgs}, [K_{c,tgs} \| ID_C \| AD_C \| ID_{tgs} \| TS_2 \| \text{Lifetime}_2])$$

$$Ticket_v = E(K_v, [K_{c,v} \| ID_C \| AD_C \| ID_v \| TS_4 \| \text{Lifetime}_4])$$

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

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

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

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

$$Ticket_v = E(K_v, [K_{c,v} \| ID_C \| AD_C \| ID_v \| TS_4 \| \text{Lifetime}_4])$$

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

(c) Client/Server Authentication Exchange to obtain service

By the end of step (a):

- C has shared his ID with the AS
- The AS has replied back with a challenge generated from the hashed version of the user's saved password
- The AS provided TGT to C and a session key $K_{c,tgs}$.

How *ADc* is known to the AS and what is it used for?

- Contains network information about the client's workstation.
- Used to prevent the use of ticket from a different workstation
- AS obtains form the received packet header information.

By the end of step (b):

- C has shared his authenticator with and TGT with TGS
- TGS verfies the *IDc, ADc*, timestamps and lifetime matches the content of the authenticator
- TGS then sends back $Ticket_v$ and, updated timestamp, IDv and session key $K_{c,v}$ to the C.

By the end of step (c):

- C shared $Ticket_v$ and authenticaor with V
- V verfies the *IDc, ADc*, timestamps and lifetime matches the content of the authenticator
- V sends backs to C an encrypted message having only the timestamp incremented encrypted with $K_{c,v}$ to confirm the C's identity has been validated and now V is ready for a session where data will be encrypted using $K_{c,v}$.

What are the differences between Kerveros V4 and V5?

- V5 intended to address limitation of V4 including encryption using DES. V5 uses an encryption identifier to allow any encryption technique to be used.
- V5 allows for different more network addresses to be used.
- Lifetime of ticket in V4 was limited to 21 hours. In V5, tickets have a start and end date to allow for an arbitrary lifetime of the ticket.
- V5 allows for subsession keys (not only one key for session as in V4).
- V5 allows for authentication forwarding between different service servers (was not enabled in V4).

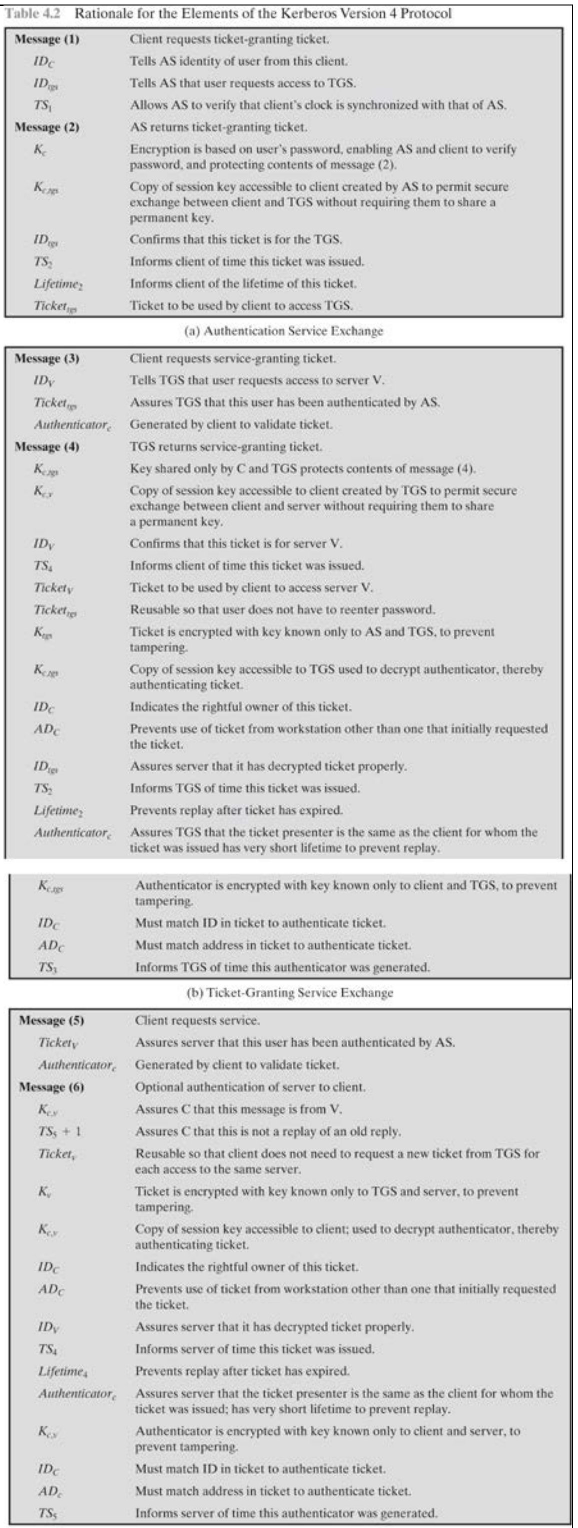


Table 4.2 Rationale for the Elements of the Kerberos Version 4 Protocol

Message (1)	Client requests ticket-granting ticket.
ID_C	Tells AS identity of user from this client.
ID_{tgs}	Tells AS that user requests access to TGS.
TS_1	Allows AS to verify that client's clock is synchronized with that of AS.
Message (2)	AS returns ticket-granting ticket.
K_c	Encryption is based on user's password, enabling AS and client to verify password, and protecting contents of message (2).
$K_{c,tgs}$	Copy of session key accessible to client created by AS to permit secure exchange between client and TGS without requiring them to share a permanent key.
ID_{tgs}	Confirms that this ticket is for the TGS.
TS_2	Informs client of time this ticket was issued.
$Lifetime_2$	Informs client of the lifetime of this ticket.
$Ticket_{tgs}$	Ticket to be used by client to access TGS.

(a) Authentication Service Exchange

Message (3)	Client requests service-granting ticket.
ID_V	Tells TGS that user requests access to server V.
$Ticket_{tgs}$	Assures TGS that this user has been authenticated by AS.
$Authenticator_c$	Generated by client to validate ticket.
Message (4)	TGS returns service-granting ticket.
$K_{c,tgs}$	Key shared only by C and TGS protects contents of message (4).
$K_{c,v}$	Copy of session key accessible to client created by TGS to permit secure exchange between client and server without requiring them to share a permanent key.
ID_V	Confirms that this ticket is for server V.
TS_4	Informs client of time this ticket was issued.
$Ticket_V$	Ticket to be used by client to access server V.
$Ticket_{tgs}$	Reusable so that user does not have to reenter password.
K_{tgs}	Ticket is encrypted with key known only to AS and TGS, to prevent tampering.
$K_{c,tgs}$	Copy of session key accessible to TGS used to decrypt authenticator, thereby authenticating ticket.
ID_C	Indicates the rightful owner of this ticket.
AD_C	Prevents use of ticket from workstation other than one that initially requested the ticket.
ID_{tgs}	Assures server that it has decrypted ticket properly.
TS_2	Informs TGS of time this ticket was issued.
$Lifetime_2$	Prevents replay after ticket has expired.
$Authenticator_c$	Assures TGS that the ticket presenter is the same as the client for whom the ticket was issued has very short lifetime to prevent replay.
$K_{c,tgs}$	Authenticator is encrypted with key known only to client and TGS, to prevent tampering.
ID_C	Must match ID in ticket to authenticate ticket.
AD_C	Must match address in ticket to authenticate ticket.
TS_3	Informs TGS of time this authenticator was generated.

(b) Ticket-Granting Service Exchange

Message (5)	Client requests service.
$Ticket_V$	Assures server that this user has been authenticated by AS.
$Authenticator_c$	Generated by client to validate ticket.
Message (6)	Optional authentication of server to client.
$K_{c,v}$	Assures C that this message is from V.
$TS_5 + 1$	Assures C that this is not a replay of an old reply.
$Ticket_v$	Reusable so that client does not need to request a new ticket from TGS for each access to the same server.
K_v	Ticket is encrypted with key known only to TGS and server, to prevent tampering.
$K_{c,v}$	Copy of session key accessible to client; used to decrypt authenticator, thereby authenticating ticket.
ID_C	Indicates the rightful owner of this ticket.
AD_C	Prevents use of ticket from workstation other than one that initially requested the ticket.
ID_V	Assures server that it has decrypted ticket properly.
TS_4	Informs server of time this ticket was issued.
$Lifetime_4$	Prevents replay after ticket has expired.
$Authenticator_c$	Assures server that the ticket presenter is the same as the client for whom the ticket was issued; has very short lifetime to prevent replay.
$K_{c,v}$	Authenticator is encrypted with key known only to client and server, to prevent tampering.
ID_C	Must match ID in ticket to authenticate ticket.
AD_c	Must match address in ticket to authenticate ticket.
TS_5	Informs server of time this authenticator was generated.

• V4 employs a byte ordering of its own. V5 uses a more conventional well established byte ordering to provide an unambiguous byte ordering.	

Key Distribution using Asymmetric Encryption

Public key encryption is used for:

- The distribution of public keys
- Distribution of secret (symmetric) keys

Public key certificates

- Consists of a public key plus a user ID of the key owner, and the whole block **signed by a trusted third party**.
- This third party is referred to as certificate authority (CA).
- A user presents his or her public key to the CA in a secure way, then obtain a signed certificate.
- The user can then publish the certificate.

Why public keys are shared through a certificate signed by a CA?

- If a user sends his public key directly to other users, anyone can forge such public announcement and pretend to be the user and resend this public key to others.

Public key distribution of secret keys

- The ID info of the user and his public key are concatenated with the CA information. Then all this info is hashed.
- The hashed message is then encrypted using the **private key** of the CA.
- The encrypted hashed message (signature) is appended to the certificate.
- There is a verification step that is conducted at the receiving end.
- The verification step is applied to ensure that the signature is indeed the signature of a CA.
- The verification step is applied by decrypting the hashed message using PK of the CA.
- Note that this implies that receiving party must have acquired the PK of the signing CA in a trusted way.

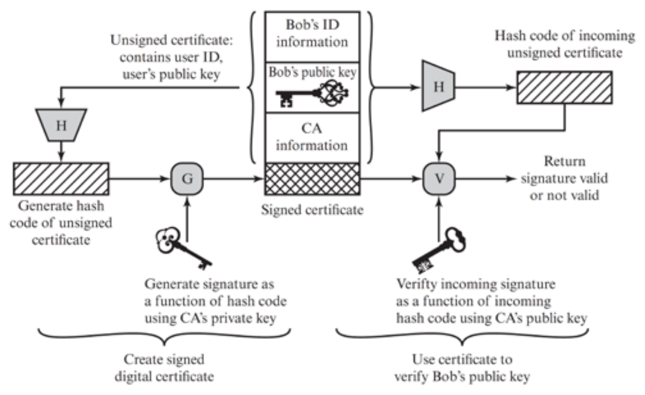

Unsigned certificate: contains user ID, user's public key
Bob's ID information
Bob's public key
CA information
Hash code of incoming unsigned certificate
H
H
G
V
Generate hash code of unsigned certificate
Signed certificate
Return signature valid or not valid
Generate signature as a function of hash code using CA's private key
Verifty incoming signature as a function of incoming hash code using CA's public key
Create signed digital certificate
Use certificate to verify Bob's public key

Figure 4.4 Public-Key Certificate Use

X.509 Certificate

- X.500 is a series of recommendations that define a directory service.
- X509 defines a framework for the provision of authentication services by the X.500 directory to its users.
- The directory may serve as a repository for PK certificates.
- The standard does not specify the use of a specific digital signature algorithm nor a specific hash function.
- The certificate may be placed in a directory by the CA or by the user.
- The directory server itself is not responsible for the creation of the PK or the certification functions.
- The directory provides an easily accessible location for users to obtain certificates.
- No party other than the CA can modify the certificate without being detected.

General format of X.509 Certificates

- **Version**: differentiates among successive versions of the certificate format

- **Serial number**: integer value, unique within the CA
- **Signature algorithm identifier**: identifies which algorithm used to sign the certificate
- **Issuer name**: name of CA
- Period of validity
- **Subject name**: name of the user
- **Subject PK info**: the PK of the user plus the identifier of the PK algorithm
- **Issuer unique identifier**: used to uniquely identify the issuing CA.
- **Subject unique identifier**: used to uniquely identify the issuing subject in the case the name has been used for different entities.
- **Extension**: based on the version
- **Signature**: the digital signature of the CA and info about the algorithm used in the digital signature.

Revocation of certificates:

- A certificate could be revoked for one of the following reasons:
  - The user PK is compromises
  - The user is no longer certified by the CA
  - The certificate is compromised
- The certificate revocation list is signed by the CA and posted on the directory.
- The certificate revocation list contains the serial number of the certificate and the revocation date.

Obtaining a User's Certificate:

- There are many CA. This implies that one user A could be certified from a CA named X, and another user B could be certified from another CA named Z.
- A would like to communicate with B. A receives B's certificate but needs to verify the signature of the CA that signed B's certificate, i.e., verify the signature of Z.
- Different CA's can certify one another and create certificates of each other → this creates a chain of certificates.
- Forward certificate: Certificate of X generated by other CA's
- Reverse Certificate: certificated generated by X that are the certificates of other CAs.
- A can acquire the following certificates from the directory to establish a certification path to B:
  X<<W>> W<<V>> V<<Y>> Y<<Z>> Z<<B>>

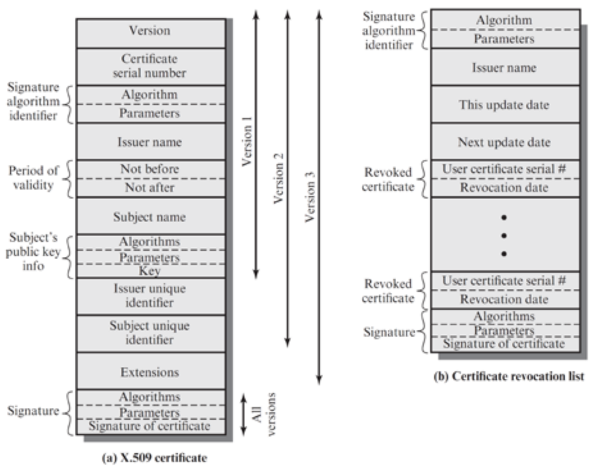


Figure 4.5 X.509 Formats

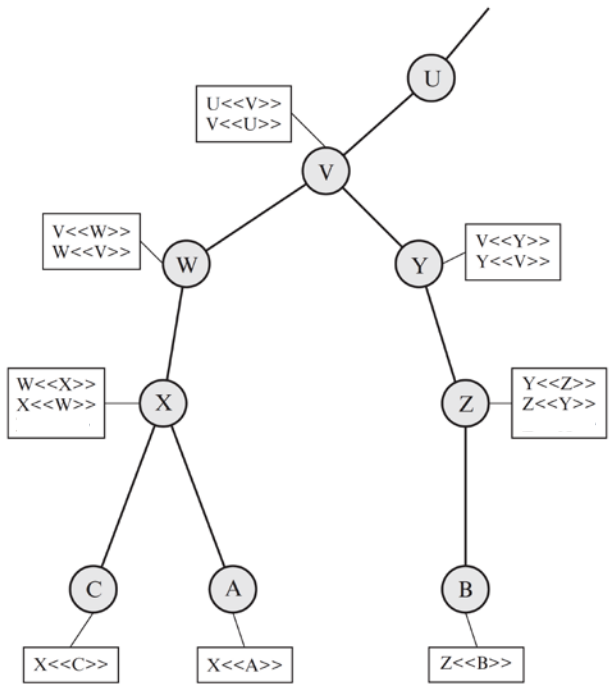


Figure 4.6 X.509 Hierarchy: A Hypothetical Example

Identity management

- A centralized automated approach to provide enterprise-wide access to resources by employees and other authorized individuals.
- The central concept of identity management is the use of single sign-on (SSO).
- SSO enables a user to access all network resources after a single authentication.

Federated Identity management:

- The use of a common identity management scheme across multiple enterprises and numerous applications.
- Refers to agreements, standards and technologies that enable portability of identities, attributes, and entitlement across multiple enterprises.

Typical services provided by a federated identity management system include the following:

- **Point of contact:** Includes authentication that a user corresponds to the user name provided, and management of user/server sessions.
- **SSO protocol services:** Provides a vendor-neutral security token service for supporting a single sign on to federated services.
- **Trust services:** Federation relationships require a trust relationship-based federation between business partners. A trust relationship is represented by the combination of the security tokens used to exchange information about a user, the cryptographic information used to protect these security tokens, and optionally the identity mapping rules applied to the information contained within this token.
- **Key services:** Management of keys and certificates.
- **Identity services:** Services that provide the interface to local data stores, including user registries and databases, for identity-related information management.

- **Authorization:** Granting access to specific services and/or resources based on the authentication.
- **Provisioning:** Includes creating an account in each target system for the user, enrollment or registration of user in accounts, establishment of access rights or credentials to ensure the privacy and integrity of account data.
- **Management:** Services related to runtime configuration and deployment.

<u>Generic Identity management architecture includes:</u>

- **Principal**: identity holder
- **identity provider**: associates authentication information with a principal.
- **Attribute Service**: manages the creation and maintenance of attributes (attributes include identifier, passwords, biometric information)
- **Administrators**: may also assign attributes to users such as roles and access permission.
- **Data Consumers**: an entities that obtain and employ data maintained and provided by identity and attribute provider, for example data server or file server.

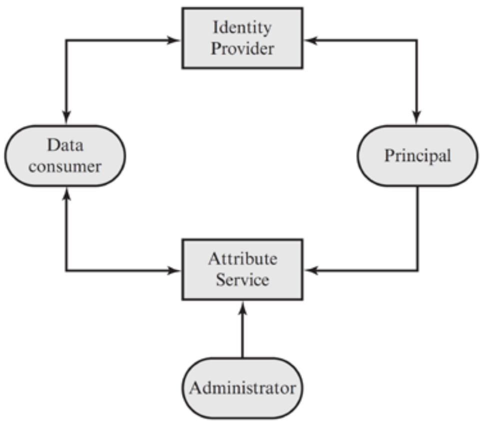
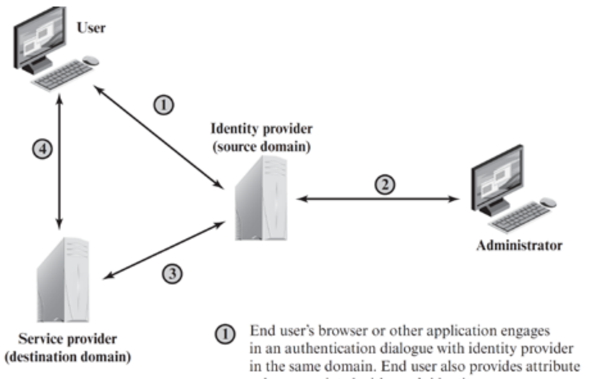


Figure 4.8 Generic Identity Management Architecture


User
Identity provider (source domain)
Administrator
Service provider (destination domain)
1
2
3
4

1. End user's browser or other application engages in an authentication dialogue with identity provider in the same domain. End user also provides attribute values associated with user's identity.
2. Some attributes associated with an identity, such as allowable roles, may be provided by an administrator in the same domain.
3. A service provider in a remote domain, which the user wishes to access, obtains identity information, authentication information, and associated attributes from the identity provider in the source domain.
4. Service provider opens session with remote user and enforces access control restrictions based on user's identity and attributes.

Figure 4.9 Federated Identity Operation

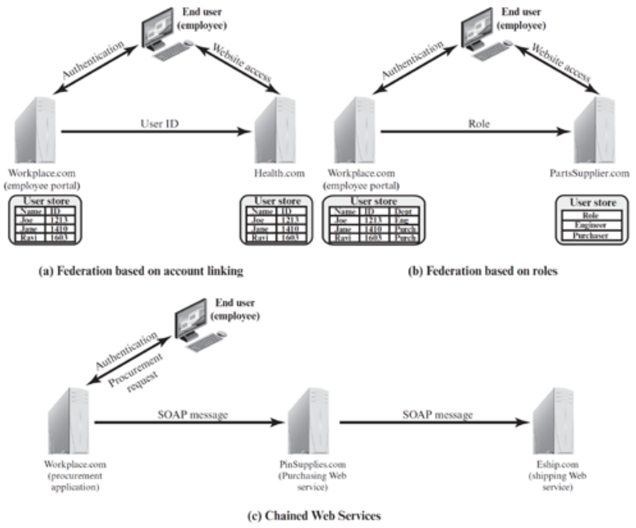


Figure 4.10 Federated Identity Scenarios