Chapter 4

Authentication Applications

By: Bhargavi H Goswami,
Assistant Professor,
Sunshine Group of Institutes,
Rajkot.
Email: bhargavigoswami@gmail.com
Cell No: 9426669020
Outline

• Kerberos
  - Kerberos Version 4
  - Kerberos Version 5

• X.509 Certificates
  - Certificates
  - X.509 Versions

Email: bhargavigoswami@gmail.com,
Mob: 9426669020
Security Concerns

• Key concerns are
  - confidentiality and
  - timeliness.

• Confidentiality:
  - To provide confidentiality must encrypt identification and session key info
  - which requires the use of previously shared private or public keys

• Timeliness:
  - need timeliness to prevent replay attacks
  - provided by using sequence numbers or timestamps or challenge/response

Email: bhargavigoswami@gmail.com,
Mob: 9426669020
In Greek mythology, a many headed dog, the guardian of the entrance of Hades (under world).

Email: bhargavigoswami@gmail.com,
Mob: 9426669020
Requirement: Kerberos

• It is an authentication service developed as part of Project Athena at MIT.
• So far we have studied secure communication between two or more peer nodes.
• What about client server applications and its security?
• Here comes Kerberos into the picture.

Email: bhargavigoswami@gmail.com,
Mob: 9426669020
Kerberos

- Users wish to access services on servers but workstations and servers cannot identify the trusted users.
- Three threats exist:
  1. User pretend to be another user.
  2. User alter the network address of a workstation.
  3. User eavesdrop on exchanges and use a replay attack.

Email: bhargavigoswami@gmail.com,
Mob: 9426669020
KERBEROS

• Provides a centralized authentication server to authenticate users to servers and servers to users.
• Relies on conventional encryption, making no use of public-key encryption
• Two versions: version 4 and 5
• Version 4 makes use of DES.
• **Motivation:**

• In today’s scenario, distributed architecture consisting of dedicated user workstations (clients) and distributed or centralized servers. We require:

• 1. Enforce security using ID Identification by any party in communication.

• 2. Required that clients authenticate themselves to servers before requesting for service.

• 3. Required that user provide identity for each service invoked and servers provide their identity to clients.

Email: bhargavigoswami@gmail.com, Mob: 9426669020
First published Kerberos listed following requirements:

- **Secure**: From eavesdroppers so that they do not impersonate other users.
- **Reliability**: Kerberos means availability of service and for access control.
- **Transparent**: User should not know that authentication is taking place at any stage.
- **Scalable**: System should support large number of clients and servers that support modular, distributed architecture.

Email: bhargavigoswami@gmail.com,
Mob: 9426669020
Kerberos: Version 4

- **Authentication Server (AS):** that knows the passwords of all users and stores these in a centralized database.
- AS shares a unique secret key with each server.
- **Step 1:** The client module C in the user’s workstation requests the user’s password and then sends a message to the AS that includes the user’s ID, the server’s ID, and the user’s password.
- **Step 2:** The AS checks its database to see if the user has supplied the proper password for this user ID and whether this user is permitted access to server V.
- **Step 3:** If both tests are passed, the AS accepts the user as authentic and must now convince the server that this user is authentic. To do so, the AS creates a ticket that contains the user’s ID and network address and the server’s ID.
- **Step 4:** This ticket is encrypted using the secret key shared by the AS and this server.
- **Step 5:** This ticket is then sent back to C.
- **Step 6:** With this ticket, C can now apply to V for service. C sends a message to V containing C’s ID and the ticket.
- **Step 7:** V decrypts the ticket and verifies that the user ID in the ticket is the same as the unencrypted user ID in the message.
- **Step 8:** If these two match, the server considers the user authenticated and grants the requested service.

Email: bhargavigoswami@gmail.com,  
Mob: 9426669020
Kerberos Version 4

- Terms:
  - C = Client
  - AS = authentication server
  - V = server
  - IDc = identifier of user on C
  - IDv = identifier of V
  - Pc = password of user on C
  - ADc = network address of C
  - Kv = secret encryption key shared by AS an V
  - TS = timestamp
  - || = concatenation
A Simple Authentication Dialogue

(1) C → AS:
   IDc || Pc || IDv

(2) AS → C:
   Ticket

(3) C → V:
   IDc || Ticket

Ticket = E_{Kv}[IDc || Pc || IDv]

Email: bhargavigoswami@gmail.com,
Mob: 9426669020
• **Significance of each ingredients:**

• The **ticket** is encrypted to prevent alteration or forgery.

• The server’s ID \((\text{ID}_V)\) is included in the ticket so that the server can verify that it has decrypted the ticket properly.

• \(\text{ID}_C\) is included in the ticket to indicate that this ticket has been issued on behalf of \(C\).

• Finally, \(\text{AD}_C\) serves to counter the threat of using stolen ticket to send message creating problem of impersonation.

• **AS** includes in the ticket the **network address** from which the original request came.
Version 4 Limitations: Improvements

- Problems:
  - A user would need a new ticket for every different service thus, each time password entry. Then
    - Minimize the number of times that a user has to enter a password.
    - If too short → repeatedly asked for password
    - If too long → greater opportunity to replay
  - The threat is that an opponent will steal the ticket and use it before it expires.

- Need improvements? Solution:
  - Applying TGS (Ticket Granting Server)

Email: bhargavigoswami@gmail.com,
Mob: 9426669020
**TGS: Ticket Granting Server**

- **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 applies to 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.

---

**Diagram:**

```
C → AS → TGS → V
```

- **C**: Client
- **AS**: Authentication Server
- **TGS**: Ticket Granting Server
- **V**: Service Provider

**Email:** bhargavigoswami@gmail.com,  
**Mob:** 9426669020
Version 4 Authentication Dialogue

Authentication Service Exchange: To obtain Ticket-Granting Ticket

1. C → AS: IDc || IDtgs || TS1
2. AS → C: $E_{K_c}[K_{c,tgs}|| IDtgs || TS_2 || \text{Lifetime}_2 || \text{Ticket}_{tgs}]$

Ticket-Granting Service Exchange: To obtain Service-Granting Ticket

3. C → TGS: IDv || Ticket_{tgs} || Authenticator_c
4. TGS → C: $E_{K_c}[K_{c,v}|| IDv || TS_4 || \text{Ticket}_{v}]$

Client/Server Authentication Exchange: To Obtain Service

5. C → V: Ticket_v || Authenticator_c
6. V → C: $E_{K_c,v}[TS5 +1]$

Email: bhargavigoswami@gmail.com,
Mob: 9426669020
• **Step 1.** The client requests a ticket-granting ticket on behalf of the user by sending its user’s ID to the AS, together with the TGS ID, indicating a request to use the TGS service.

• **Step 2.** The AS responds with a ticket that is encrypted with a key that is derived from 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 password, generates the key, and attempts to decrypt the message. If the correct password is supplied, the ticket is successfully recovered.

• **Step 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.

• **Step 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 the lifetime 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 a ticket to grant access to the requested service.

• **Step 5:** The client requests access to a service on behalf of the user. For this purpose, the client transmits a message to the server containing the user’s ID and the service granting ticket. The server authenticates by using the contents of the ticket.

Email: bhargavgoswami@gmail.com,
Mob: 9426669020
1. User logs on to workstation and requests service on host.

3. Workstation prompts user for password and uses password to decrypt incoming message, then sends ticket and authenticator that contains user's name, network address, and time to TGS.

5. Workstation sends ticket and authenticator to server.

2. AS verifies user's access right in database, creates ticket-granting ticket and session key. Results are encrypted using key derived from user's password.

4. TGS decrypts ticket and authenticator, verifies request, then creates ticket for requested server.

6. Server verifies that ticket and authenticator match, then grants access to service. If mutual authentication is required, server returns an authenticator.
Limitations still remains

- This new scenario satisfies the two requirements:
  - only one password query per user session
  - Protection of the user password
- Two additional problems remains:
  - If this lifetime is very short (e.g., minutes), then the user will be repeatedly asked for a password.
  - If the lifetime is long (e.g., hours), then an opponent has a greater opportunity for replay.
- Solution?
- Kerberos Realms & Multiple Kerberi.

Email: bhargavigoswami@gmail.com,
Mob: 9426669020
Kerberos Realms & Multiple Kerberi

- A full-service 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.

Email: bhargavigoswami@gmail.com, Mob: 9426669020
• **Kerberos realm:**
  - Kerberos realm is a set of managed nodes that share the same Kerberos database.
  - Kerberos database resides on the Kerberos master computer system.
  - Changing or accessing the contents of a Kerberos database requires the Kerberos master password.

• **Kerberos principal:**
  - 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.
• Networks of clients and servers under different administrative organizations typically constitute different realms.

• Users in one realm may need access to servers in other realms, and some servers may be willing to provide service to users from other realms, provided that those users are authenticated.

• Kerberos provides a mechanism for supporting such inter-realm authentication.

• Let us see the mechanism.
Kerberos Realms

(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}: \[ Ticket_{Vrem} \| Authenticator_C \]

Email: bhargavigoswami@gmail.com,
Mob: 9426669020
• **Message 1: Client Request Ticket Granting Ticket:**
  - $ID_c = \text{Tells AS identity of user from this client.}$
  - $ID_{tgs} = \text{Tells AS that user request access to TGS.}$
  - $TS_1 = \text{Allows AS to verify that client's clock is synchronized with that of AS. i.e Timestamp.}$

Email: bhargavigoswami@gmail.com,
Mob: 9426669020
• **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 the time this ticket was issued.
  
  — $\text{Lifetime}_2 = $ Informs client of the lifetime of this ticket.
  
  — $\text{Ticket}_{tgs} = $ Ticket to be used by client to access TGS.

Email: bhargavigoswami@gmail.com, Mob: 9426669020
• **Message 3:** Client requests service-granting ticket:
  - \( ID_v \) = Tells TGS that user requests access to server \( V \).
  - \( Ticket_{tgsrem} \) = Assures TGS that this user has been authenticated by AS of other realm.
  - Authenticator\( _c \) = Generated by client to validate ticket.

Email: bhargavigoswami@gmail.com,
Mob: 9426669020
• **Message 4:** TGS returns service-granting ticket.

• $K_{c,tgs} =$ Key shared only by C and TGS protects contents of message.

• $K_{c,tgsrem} =$ Copy of session key accessible to TGS of current realm used to decrypt authenticator, thereby authenticating ticket.

• $ID_{tgsrem} =$ Confirms that this ticket is for server of other realm and towards TGS of other realm.

• $TS_4 =$ Informs client of time this ticket was issued.

• $Ticket_{tgsrem} =$ Reusable so that user does not have to reenter password.
• **Message 5**: Client requests ticket from other realm’s TGS.

• $ID_{vrem} =$ Confirms that this ticket is for server $V$ of realm.

• $Ticket_{tgsrem} =$ Assures TGS of realm that this user has been authenticated by AS and TGS by previous realm.

• $Authenticator_c =$ Generated by client to validate ticket.

Email: bhargavigoswami@gmail.com,
Mob: 9426669020
• **Message 6**: Ticket issued for remote server.

• $K_{c,tgsrem} = \text{Authenticator is encrypted with key known only to client and TGS of realm B, to prevent tampering.}$

• $K_{c,vrem} = \text{Copy of session key accessible to client; used to decrypt authenticator, thereby authenticating ticket in next realm.}$

• $ID_{vrem} = \text{Assures server that it has decrypted ticket properly in realm B.}$

• $TS_6 = \text{Informs server of time this ticket was issued.}$

• $Ticket_{vrem} = \text{Assures server that this user has been authenticated by AS of realm A and TGS of realm B.}$
• **Message 7: Request Remote Service**
• \( \text{Ticket}_{vrem} = \text{Assures server that this user has been authenticated by AS of realm } A \) and \( \text{TGS of realm } B. \)
• \( \text{Authenticator}_c = \text{Generated by client to validate ticket} \)
Request for Service in Another Realm

Email: bhargavgoswami@gmail.com
Mob: 9426669020

Figure 4.2 Request for Service in Another Realm
Limitations:

• One problem presented by the foregoing approach is that it does not scale well to many realms.

• If there are $N$ realms, then there must be $N(N - 1)/2$ secure key exchanges so that each Kerberos realm can interoperate with all other Kerberos realms.

Email: bhargavigoswami@gmail.com,
Mob: 9426669020
Difference Between Version 4 and 5

Environmental Shortcomings:
1. Encryption system dependence (V.4 DES)
2. Internet protocol dependence
3. Message byte ordering
4. Ticket lifetime
5. Authentication forwarding
6. Inter-realm authentication

Technical Deficiencies:
1. Double Encryption
2. PCBC Encryption
3. Session Keys
4. Password Attacks

Email: bhargavigoswami@gmail.com,
Mob: 9426669020
<table>
<thead>
<tr>
<th>Point: Environmental Shortcomings:</th>
<th>Version 4</th>
<th>Version 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Encryption System Dependence:</td>
<td>Requires use of DES. Limitation of Export restriction and doubts about strength.</td>
<td>Ciphertext is tagged with encryption type. Thus, any encryption key can be used. Keys are tagged with type and length. Same key can be used for any algorithm or any version.</td>
</tr>
<tr>
<td>Internet Protocol Dependence:</td>
<td>Version 4 requires the use of Internet Protocol (IP) addresses only. Other address types, such as the ISO network address, are not accommodated.</td>
<td>Version 5 network addresses are tagged with type and length, allowing any network address type to be used.</td>
</tr>
<tr>
<td>Message Byte Ordering:</td>
<td>The sender of a message employs a byte ordering of its own choice.</td>
<td>All message structures are defined using Abstract Syntax Notation One (ASN.1) and Basic Encoding Rules (BER), which provide an unambiguous byte ordering.</td>
</tr>
</tbody>
</table>

Email: bhargavigoswami@gmail.com, Mob: 9426669020
<table>
<thead>
<tr>
<th>Point: Environmental Shortcomings:</th>
<th>Version 4</th>
<th>Version 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ticket Lifetime:</td>
<td>Encoded in an 8-bit quantity in units of five minutes. Thus, the maximum lifetime is $28 \times 5 = 1280$ minutes (a little over 21 hours). <strong>Inadequate</strong> for long running simulations.</td>
<td><strong>tickets include an explicit start time and end time, allowing tickets with arbitrary lifetimes.</strong></td>
</tr>
<tr>
<td>Authentication Forwarding:</td>
<td>Does not allow credentials issued to one client to be forwarded to some other host and used by some other client.</td>
<td>Provides this capability.</td>
</tr>
<tr>
<td>Inter-realm Authentication:</td>
<td>Interoperability among $N$ realms requires on the order of $N^2$ Kerberos-to-Kerberos relationships.</td>
<td>Supports a method that requires fewer relationships.</td>
</tr>
<tr>
<td>Point: Technical Deficiencies</td>
<td>Version 4</td>
<td>Version 5</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-----------</td>
<td>-----------</td>
</tr>
<tr>
<td><strong>Double Encryption:</strong></td>
<td>messages (2) and (4)] that tickets provided to clients are encrypted twice.</td>
<td>second encryption is not necessary and is computationally wasteful.</td>
</tr>
<tr>
<td><strong>PCBC Encryption:</strong></td>
<td>makes use of a nonstandard mode of DES known as “propagating cipher block chaining (PCBC)”. Limitation: attack involving the interchange of ciphertext blocks.</td>
<td>The standard CBC mode is used for encryption. A checksum or hash code is attached to the message prior to encryption using CBC.</td>
</tr>
<tr>
<td><strong>Session Keys</strong></td>
<td>Used by client to authenticate and encrypt. Thus, protect message passed during that session. Because the same ticket may be used repeatedly, risk of replay attack.</td>
<td>It is possible for a client and server to negotiate a subsession key, which is to be used only for that one connection. A new access by the client would result in the use of a new subsession key.</td>
</tr>
</tbody>
</table>

Email: bhargavigoswami@gmail.com, Mob: 9426669020
<table>
<thead>
<tr>
<th>Point: Technical Deficiency</th>
<th>Version 4</th>
<th>Version 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Password Attack</td>
<td>Both versions are vulnerable to a password attack. The message from the AS to the client includes material encrypted with a key based on the client’s password. An opponent can capture this message and attempt to decrypt it by trying various passwords.</td>
<td>does provide a mechanism known as pre-authentication, which should make password attacks more difficult, but it does not prevent them.</td>
</tr>
</tbody>
</table>

Email: bhargavigoswami@gmail.com,  
Mob: 9426669020
Kerberos - Version 5

- Currently have two Kerberos versions:
  - 4: restricted to a single realm
  - 5: allows inter-realm authentication, in beta test
- Kerberos v5 is an Internet standard
- specified in RFC1510, and used by many utilities
- To use Kerberos:
  - need to have a KDC on your network
  - need to have Kerberised applications running on all participating systems
  - major problem - US export restrictions
  - Kerberos cannot be directly distributed outside the US in source format (& binary versions must obscure crypto routine entry points and have no encryption)
  - Else crypto libraries must be re-implemented locally

Email: bhargavigoswami@gmail.com,
Mob: 9426669020
• **Realm**: Indicates realm of user.
• **Options**: Used to request that certain flags be set in the returned ticket.
• **Times**: Used by the client to request the following time settings in the ticket:
  - from: the desired start time for the requested ticket
  - till: the requested expiration time for the requested ticket
  - rtime: requested renew-till time
• **Nonce**: A random value to be repeated in message (2) to assure that the response is fresh and has not been replayed by an opponent.
• **Subkey**: The client’s choice for an encryption key to be used to protect this specific application session. If this field is omitted, the session key from the ticket ($K_c, V$) is used.
• **Sequence number**: An optional field that specifies the starting sequence number to be used by the server for messages sent to the client during this session. Messages may be sequence numbered to detect replays.
• This message includes the timestamp from the authenticator.
• **Version 4**, timestamp is incremented by one.
• **Version 5**, not necessarily incremented by one plus sub-key is used. More secure.
• **Ticket Flag**: Supports Expanded Functionality.

Email: bhargavigoswami@gmail.com,
Mob: 9426669020
Table 4.3 Summary of Kerberos Version 5 Message Exchanges

<table>
<thead>
<tr>
<th>Step</th>
<th>Message Exchange</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1)</td>
<td>C → AS  Options</td>
</tr>
<tr>
<td>(2)</td>
<td>AS → C  Realm&lt;sub&gt;c&lt;/sub&gt;</td>
</tr>
<tr>
<td></td>
<td>Ticket&lt;sub&gt;tgs&lt;/sub&gt; = E(K&lt;sub&gt;tgs&lt;/sub&gt;, [Flags</td>
</tr>
<tr>
<td>(3)</td>
<td>C → TGS Options</td>
</tr>
<tr>
<td>(4)</td>
<td>TGS → C  Realm&lt;sub&gt;c&lt;/sub&gt;</td>
</tr>
<tr>
<td></td>
<td>Ticket&lt;sub&gt;tgs&lt;/sub&gt; = E(K&lt;sub&gt;tgs&lt;/sub&gt;, [Flags</td>
</tr>
<tr>
<td></td>
<td>Ticket&lt;sub&gt;v&lt;/sub&gt; = E(K&lt;sub&gt;v&lt;/sub&gt;, [Flags</td>
</tr>
<tr>
<td></td>
<td>Authenticator&lt;sub&gt;c&lt;/sub&gt; = E(K&lt;sub&gt;c,tgs&lt;/sub&gt;, [ID&lt;sub&gt;C&lt;/sub&gt;</td>
</tr>
<tr>
<td>(5)</td>
<td>C → V   Options</td>
</tr>
<tr>
<td>(6)</td>
<td>V → C   E&lt;sub&gt;K&lt;sub&gt;c,v&lt;/sub&gt;&lt;/sub&gt; [ TS&lt;sub&gt;2&lt;/sub&gt;</td>
</tr>
<tr>
<td></td>
<td>Ticket&lt;sub&gt;v&lt;/sub&gt; = E(K&lt;sub&gt;v&lt;/sub&gt;, [Flags</td>
</tr>
<tr>
<td></td>
<td>Authenticator&lt;sub&gt;c&lt;/sub&gt; = E(K&lt;sub&gt;c,v&lt;/sub&gt;, [ID&lt;sub&gt;C&lt;/sub&gt;</td>
</tr>
</tbody>
</table>

(c) Client/Server Authentication Exchange to obtain service
<table>
<thead>
<tr>
<th>Flag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>INITIAL</td>
<td>This ticket was issued using the AS protocol, and not issued based on a</td>
</tr>
<tr>
<td></td>
<td>ticket-granting ticket.</td>
</tr>
<tr>
<td>PRE-AUTHENT</td>
<td>During initial authentication, the client was authenticated by the KDC</td>
</tr>
<tr>
<td></td>
<td>before a ticket was issued.</td>
</tr>
<tr>
<td>HW-AUTHENT</td>
<td>The protocol employed for initial authentication required the use of</td>
</tr>
<tr>
<td></td>
<td>hardware expected to be possessed solely by the named client.</td>
</tr>
<tr>
<td>RENEWABLE</td>
<td>Tells TGS that this ticket can be used to obtain a replacement ticket that</td>
</tr>
<tr>
<td></td>
<td>expires at a later date.</td>
</tr>
<tr>
<td>MAY-POSTDATE</td>
<td>Tells TGS that a postdated ticket may be issued based on this ticket-</td>
</tr>
<tr>
<td></td>
<td>granting ticket.</td>
</tr>
<tr>
<td>POSTDATED</td>
<td>Indicates that this ticket has been postdated; the end server can check</td>
</tr>
<tr>
<td></td>
<td>the authtime field to see when the original authentication occurred.</td>
</tr>
<tr>
<td>INVALID</td>
<td>This ticket is invalid and must be validated by the KDC before use.</td>
</tr>
<tr>
<td>PROXIABLE</td>
<td>Tells TGS that a new service-granting ticket with a different network</td>
</tr>
<tr>
<td></td>
<td>address may be issued based on the presented ticket.</td>
</tr>
<tr>
<td>PROXY</td>
<td>Indicates that this ticket is a proxy.</td>
</tr>
<tr>
<td>FORWARDABLE</td>
<td>Tells TGS that a new ticket-granting ticket with a different network</td>
</tr>
<tr>
<td></td>
<td>address may be issued based on this ticket-granting ticket.</td>
</tr>
<tr>
<td>FORWARDED</td>
<td>Indicates that this ticket has either been forwarded or was issued based</td>
</tr>
<tr>
<td></td>
<td>on authentication involving a forwarded ticket-granting ticket.</td>
</tr>
<tr>
<td>Flag:</td>
<td>Function:</td>
</tr>
<tr>
<td>--------------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>INITIAL</td>
<td>This ticket was issued using AS protocol and not based on Ticket Granting Ticket. Version 5 allows granting Service Granting Service directly by checking INITIAL flag indicating that password of client is checked before.</td>
</tr>
<tr>
<td>PRE-AUTHENT</td>
<td>During initial authentication, the client was authenticated by the KDC (Kerberos data centre) before a ticket was issued.</td>
</tr>
<tr>
<td>HW-AUTHENT</td>
<td>Protocol employed for initial authentication required the use of hardware expected to be possessed solely by named client.</td>
</tr>
<tr>
<td>RENEWABLE</td>
<td>Tells TGS that ticket can be used to obtain a replacement ticket that expires at a later date.</td>
</tr>
<tr>
<td>MAY-POSTDATE</td>
<td>Tells TGS that post dated ticket may be issued based on this ticket granting ticket.</td>
</tr>
<tr>
<td>POSTDATED</td>
<td>Indicates that this ticket has been post dated. Checked by server at original authentication. Reduces process.</td>
</tr>
<tr>
<td>INVALID</td>
<td>Ticket is invalid and need to be validated by KDC before use.</td>
</tr>
<tr>
<td>PROXIABLE</td>
<td>Tells TGS that new service granting ticket with a different network address may be issued based on the presented ticket.</td>
</tr>
<tr>
<td>PROXY</td>
<td>Indicates that this ticket is a proxy.</td>
</tr>
<tr>
<td>FORWARDABLE</td>
<td>Tells TGS that new ticket granting ticket with a different network address may be issued based on ticket granting ticket.</td>
</tr>
<tr>
<td>FORWARDED</td>
<td>Indicates that this ticket has either been forwarded or issued based on future scopes of authentication at forwarded</td>
</tr>
</tbody>
</table>

Email: bhargavigoswami@gmail.com, Mob: 9426669020
X.509 Authentication Service

Topic List:

a) Certificates
b) Obtaining User’s Certificates
c) Revocation of Certificates
d) Authentication Procedure
   a) One Way Authentication
   b) Two Way Authentication
   c) Three Way Authentication
e) X.509 Version 3
   a) Key and Policy Information
   b) Certificate Subject and Issuer Attributes
   c) Certification Path Constraints

Email: bhargavigoswami@gmail.com,
Mob: 9426669020
X.509 Authentication Service

- X.509 is part of the X.500 series of recommendations that define a directory service.
- Directory Service: Servers or distributed set of servers that maintains a database about users.
- X.509 is based on the use of public-key cryptography and digital signatures.
- The directory may serve as a repository of public-key certificates.
- Each certificate contains the public key of a user and is signed with the private key of a CA (Certification Authority).
- Is used in S/MIME, IP Security, SSL/TLS and SET which we’ll study in Chapter 5, 7 & 8.
- RSA is recommended to use.
- Digital Signature scheme used is hash function.

Email: bhargavigoswami@gmail.com,
Mob: 9426669020
Certificates

• The **heart** of the X.509 scheme is the public-key certificate associated with each user.

• **Certificates** are created by **Certification Authority (CA)** or user.

• Directory Server **does not create** public keys but just **provide reference** to these certificates.

• Lets have a look over **general format of certificates**:

Email: bhargavigoswami@gmail.com, 
Mob: 9426669020
X.509 Formats

(a) X.509 Certificate

- Signature
  - Signature algorithm identifier
  - Algorithm
  - Parameters

- Serial Number

- Issuer Name
  - Not before
  - Not after

- Subject Name
  - Algorithms
  - Parameters
  - Key

- Issuer Unique Identifier

- Subject Unique Identifier

- Extensions

(b) Certificate Revocation List

- Revoked certificate
  - User certificate serial #
  - Revocation date

- Next Update Date

- This Update Date

- Issuer Name

- Algorithm parameters

- algorithm

Email: bhargavgoswami@gmail.com, Mob: 9426669020
• **Version:**
  - Differentiates among successive versions of the certificate format;
  - The default is version 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.

Email: bhargavigoswami@gmail.com,
Mob: 9426669020
• **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;
  - Contains the hash code of the other fields encrypted with the CA's private key.
  - This field includes the signature algorithm identifier.

Email: bhargavigoswami@gmail.com,
Mob: 9426669020
The unique identifier fields were added in version 2 to handle the possible reuse of subject and/or issuer names over time.

These fields are rarely used.

The standard uses the following notation to define a certificate:

\[ CA \ll A \gg = CA \{V, SN, AI, CA, UCA, A, UA, Ap, T^A\} \]

Where:
- \( V \) = version of the certificate
- \( SN \) = serial number of the certificate
- \( AI \) = identifier of the algorithm used to sign the certificate
- \( CA \) = name of certificate authority
- \( UCA \) = optional unique identifier of the CA
- \( A \) = name of user A
- \( UA \) = optional unique identifier of the user A
- \( Ap \) = public key of user A
- \( T^A \) = period of validity of the certificate

Example:
- \( Y \ll X \gg = \) the certificate of user X issued by certification authority Y
- \( Y \{I\} \) = the signing of I by Y; consists of I with an encrypted hash code appended.

Typical digital signature approach. See next figure.
Typical Digital Signature Approach

(b) Using public-key encryption

Email: bhargavigoswami@gmail.com,
Mob: 9426669020
Figure 4.5  Simplified Depiction of Essential Elements of Digital Signature Process
Obtaining a User's Certificate

- Characteristics of certificates generated by CA:
  - Any user with access to the public key of the CA can recover the user public key that was certified.
  - No part other than the CA can modify the certificate without this being detected.

Email: bhargavigoswami@gmail.com,
Mob: 9426669020
• **Advantage:**
  - Once B is in possession of A’s certificate, B has confidence that messages it encrypts with A’s public key will be (a) **secure from eavesdropping** and that messages signed with A’s private key are (b) **unforgeable**.

• **Disadvantage:**
  - If there is a large community of users, it may not be practical for all users to subscribe to the same CA.
  - **Reason?**
    - Each participating user must have **CA’s public key** to verify signatures.
    - **Secure key distribution** of this public key.

• **Solution?**
• **N number of CAs**, each of which securely provides its public key to some number of the users.

• **Now,**
  - User A obtain certificate from CA, X1.
  - User B obtain 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.

• **Problem**: to enable A to obtain B’s public key.

Email: bhargavigoswami@gmail.com,
Mob: 9426669020
Solution?

• 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.

• A has used a chain of certificates to obtain B’s public key. X1 <<X2>> X2<<B>>

• In the same fashion, B can obtain A’s public key with X2<<X1>> X1<<A>>

• This can be chained like this X1 <<X2>> X2 <<X3>> ... XN <<B>>

• Such example is given in next figure.

• X.509 suggests that CAs be arranged in a hierarchy so that navigation is straightforward.
X.509 CA Hierarchy
• The directory entry for each CA includes two types of certificates:
  - **Forward certificates**: Certificates of X generated by other CAs i.e W.
  - **Reverse certificates**: Certificates generated by X that are the certificates of other CAs i.e A and C.

• In this example, user 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>>

• Similarly B obtains certification list to communicate with A as follows:
  - Z <<Y>> Y <<V>> V <<W>> W <<X>> X <<A>>

Email: bhargavigoswami@gmail.com,
Mob: 9426669020
Revocation of Certificates

• It may be desirable to **revoke** a certificate before it expires.

• Reasons for revocation:
  - The users secret key is assumed to be compromised.
  - The user is no longer certified by this CA.
  - The CA’s certificate is assumed to be compromised.

• Each CA must **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.

• These lists also should be **posted** on the directory.

Email: bhargavigoswami@gmail.com,
Mob: 9426669020
(b) Certificate revocation list

- Signature algorithm identifier
  - Algorithm Parameters
  - Issuer name
  - This update date
  - Next update date
  - User certificate serial #
  - Revocation date
  - ...

- Revoked certificate
  - User certificate serial #
  - Revocation date

- Revoked certificate
  - User certificate serial #
  - Revocation date

- Signature
  - Algorithms Parameters
  - Encrypted
• Each certificate revocation list (CRL) posted to the directory is signed by the issuer.
• CRL includes (Figure 4.4b) the issuer’s name, the date the list was created, the date the next CRL is scheduled to be issued, and an entry for each revoked certificate. Each entry consists of the serial number of a certificate and revocation date for that certificate.
• Because serial numbers are unique within a CA, the serial number is sufficient to identify the certificate.
• To avoid the delays (and possible costs) associated with directory searches, it is likely that the user would maintain a local cache of certificates and lists of revoked certificates.
Authentication Procedure:

- It is assumed that two parties know each other’s public key from
  - Directory or
  - Certificate is included as initial portion of message.
- X.509 includes three alternative authentication Procedure:
  - One Way Authentication
  - Two Way Authentication
  - Three Way Authentication

Email: bhargavigoswami@gmail.com,
Mob: 9426669020
Authentication Procedures:
One Way Authentication

1. $A\{t_A, r_A, ID_B, sgnData, E_{KU_b}[K_{ab}]\}$

(a) One-way authentication

Email: bhargavigoswami@gmail.com,
Mob: 9426669020
Two Way Authentication

1. $A\{t_a, r_A, ID_B, \text{sgnData}, E_{KU_B}[K_{ab}]\}$

2. $B\{t_B, r_B, ID_A, r_A, \text{sgnData}, E_{KU_a}[K_{ba}]\}$

(b) Two-way authentication

Email: bhargavigoswami@gmail.com,
Mob: 9426669020
Three Way Authentication

1. A\{t_A, r_A, ID_B, sgnData, E_{KU_B}[K_{ab}]\}

2. B\{t_B, r_B, ID_A, r_A, sgnData, E_{KU_A}[K_{ba}]\}

3. A\{r_B\}

(c) Three-way authentication

Email: bhargavigoswami@gmail.com,
Mob: 9426669020
• $t_A = \text{Timestamp, generation time and expiration time.}$

• $r_A = \text{used only once uniquely to sign a cryptographic communication and detect replay attack.}$

• $ID_B = \text{Identity of B.}$

• $sgnData = \text{Signed Data, optional, guaranteeing its authenticity and integrity.}$

• $\text{Message is encrypted using Public key of B.}$

• $\text{Message may carry required session keys for further computations.}$
• One Way Authentication:
  - Single transfer of information from user A to B.
  - Step 1: Identity of A and the message was generated by A.
  - Step 2: That message was intended for B.
  - Step 3: The integrity and originality (that msg is not sent multiple times) of message.
  - Used for: Presenting credentials to B. Or presenting Session Key.

• Two Way Authentication:
  - Permits both the parties in communication to verify each other’s identity.
  - Step 4: Identify B and that msg was replied by B.
  - Step 5: Message was intended for A.
  - Step 6: Integrity and originality of reply.

• Three Way Authentication:
  - Applied when synchronized clocks are not available.
  - Timestamp is not checked.
  - Both have echoed back each other, last msg needs only signed copy of nonces.

Email: bhargavigoswami@gmail.com,
Mob: 9426669020
X.509 Version 3

Following Requirements are not satisfied by Version 2:

1. The **Subject** field is inadequate to convey the identity of a key owner to a public key user. X.509 names may be relatively **short** and **lacking** in obvious **identification details** that may be needed by the user.

2. The **Subject** field is also inadequate for many applications, which typically recognize entities by an Internet e-mail address, a URL, or some other **Internet-related identification**.

3. There is a need to indicate security policy information. This enables a security application or function, such as IPSec, to relate an X.509 certificate to a **given policy**.

4. There is a need to limit the damage that can result from a faulty or malicious **CA (Corrupted)** by setting constraints on the applicability of a particular certificate.

5. It is important to be able to identify different keys used by the same owner at different times. This feature supports **key life cycle management**, in particular the ability to update key pairs for users and CAs on a regular basis or under exceptional circumstances.
• Rather than continue to add fields to a fixed format, version 3 includes a number of optional extensions that may be added to the version 2 format.

• Each extension consists of
  - an Extension Identifier,
  - a Criticality Indicator, and
  - an Extension Value.

• Criticality Indicator: indicates whether an extension can be safely ignored. If the indicator has a value of TRUE and an implementation does not recognize the extension, it must treat the certificate as invalid.

• Certificate Extensions fall into three main categories:
  1. key and policy information,
  2. subject and issuer attributes, and
  3. certification path constraints.
(1) **Key and Policy Information:**

- **Function:** Convey additional information about the subject and issuer keys, plus indicators of certificate policy.
- A certificate policy is a named set of rules that indicates the applicability of a certificate to a particular community and/or class of application with common security requirements.
- **Example:** policy might be applicable to the authentication of electronic data interchange (EDI) transactions for the trading of goods within a given price range.
- **This area includes:**

Email: bhargavigoswami@gmail.com,
Mob: 9426669020
• **Authority key identifier:**
  - Identifies the public key to be used to verify the signature on this certificate or CRL.
  - Enables distinct keys of the same CA to be differentiated.
  - Use: handle CA key pair updating.

• **Subject key identifier:**
  - Identifies the public key being certified.
  - Useful for subject key pair updating.
  - e.g., digital signature and encryption key agreement

• **Key usage:**
  - a restriction imposed as to the purposes for which, and the policies under which, the certified public key may be used.
  - Indicate digital signature, non repudiation, key encryption, data encryption, key agreement, CA signature verification on certificates, and CA signature verification on CRLs.

Email: bhargavigoswami@gmail.com, Mob: 9426669020
• **Private-key usage period:**
  - Indicates the period of use of the private key corresponding to the public key.
  - E.g. In Digital Signature, the usage period for the signing private key is typically shorter than that for the verifying public key.

• **Certificate policies:**
  - Used where multiple policies apply.
  - This extension lists policies that the certificate is recognized as supporting, together with optional qualifier information.

• **Policy mappings:**
  - Used only in certificates for CAs issued by other CAs.
  - Allow an issuing CA to indicate that one or more of that issuer’s policies can be considered equivalent to another policy used in the subject CA’s domain.
(2) Certificate Subject & Issuer Attributes

• Support alternative names, in alternative formats, for a certificate subject or certificate issuer and can convey additional information about the certificate subject to increase a certificate user’s confidence.

• Information such as postal address, position within a corporation, or picture image may be required.

• The extension fields in this area include:

   - Email: bhargavigoswami@gmail.com,
   - Mob: 9426669020
• **Subject alternative name:**
  - This field is important for supporting certain applications, such as electronic mail, EDI, and IPSec, which may employ their own name forms.

• **Issuer alternative name:**
  - Contains one or more alternative names, using any of a variety of forms.

• **Subject directory attributes:**
  - Conveys any desired X.500 directory attribute values for the subject of this certificate.
(3) Certification Path Constraints:

- Allow constraint specifications to be included in certificates issued for CAs by other CAs.
- The constraints may restrict the types of certificates that can be issued by the subject CA or that may occur subsequently in a certification chain.
- This area includes:
  - Basic constraints:
    - Indicates if the subject may act as a CA.
    - If so, a certification path length constraint may be specified.
  - Name constraints:
    - Indicates a name space within which all subject names in subsequent certificates in a certification path must be located.
  - Policy constraints:
    - Specifies constraints that may require explicit certificate policy identification or inhibit policy mapping for the remainder of the certification path.
Public Key Infrastructure

- 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.

- **Objective:** 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**:

- Has Three Parts:
  - 1. PKIX Elements
  - 2. PKIX Management Functions

Email: bhargavigoswami@gmail.com,
Mob: 9426669020
Figure 4.7 PKIX Architectural Model
**Elements:**

- **End entity:**
  - 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).
  - Support a variety of administrative functions.
  - Often delegated to number of registration authorities.

- **Registration authority (RA):**
  - optional component.
  - Does number of administrative functions from the CA.
  - 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.
PKIX Management Functions:

- **Registration:**
  - process of enrolling in a PKI.
  - involves some off-line or online procedure for mutual authentication.
  - 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.
  - Clients initialized with public key.

- **Certification:**
  - CA issues a certificate for a user’s public key and returns that certificate to the user’s client system and/or posts that certificate in a repository.

Email: bhargavigoswami@gmail.com,
Mob: 9426669020
• **Key pair recovery:**
  - 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.
  - 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.
PKIX Management Protocol:

1. RFC2510 CMP:
   - 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.

2. RFC2797 CMC:
   - Defines certificate management messages over CMS(Cryptographic Message Syntax) (CMC).
   - Intended to leverage(give power to) existing implementations.
Chapter is over....😊😊😊

• 40% syllabus is over.

• Complete the Review questions given on page number 134.

• Please practice diagrams properly.

• Tomorrow we will start with chapter 5 and unit 3.

• THANK YOU.............

Email: bhargavigoswami@gmail.com,
Mob: 9426669020