Chapter 4

Authentication Applications
OUTLINE

• Kerberos
• X.509 Authentication Service
Authentication Applications

• authentication functions developed to support application-level authentication & digital signatures
• Kerberos – a private-key authentication service
• X.509 - a public-key directory authentication service
Security Concerns

• key concerns are **confidentiality** and **timeliness**
• to provide confidentiality must encrypt identification and session key info
• which requires the use of previously shared private or public keys
• need timeliness to prevent **replay attacks**
• provided by using sequence numbers or timestamps or challenge/response
KERBEROS

In Greek mythology, a many headed dog, the guardian of the entrance of Hades
KERBEROS

• Users wish to access services on servers.
• Three threats exist:
  – User pretend to be another user.
  – User alter the network address of a workstation.
  – User eavesdrop on exchanges and use a replay attack.
Kerberos Requirements

• its first report identified requirements as:
  – secure
  – reliable
  – transparent
  – scalable
Kerberos

• trusted key server system from MIT
• provides centralized secret-key third-party authentication in a distributed network
  – allows users access to services distributed through network
  – without needing to trust all workstations
  – rather all trust a central authentication server
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
A Simple Authentication Dialogue

(1) C → AS: \( \text{ID}_c \parallel \text{P}_c \parallel \text{ID}_v \)

(2) AS → C: Ticket

(3) C → V: \( \text{ID}_c \parallel \text{Ticket} \)

Ticket = \( E_{K_v}[\text{ID}_c \parallel \text{AD}_c \parallel \text{ID}_v] \)

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 and V
A Simple Authentication Dialogue

• Problems:
  – Ticket nonusable
    • different servers for different services
  – Plaintext transmission of password

• Solution:
  – Ticket-Granting Server (TGS)
  – No plaintext transmission of password
A More Secure Authentication Dialogue

Once per user logon session:

(1)  \( C \to AS: \quad \text{ID}_c \| \text{ID}_{tgs} \)
(2)  \( AS \to C: \quad E_{K_c} [\text{Ticket}_{tgs}] \)

Once per type of service:

(3)  \( C \to TGS: \quad \text{ID}_c \| \text{ID}_v \| \text{Ticket}_{tgs} \)
(4)  \( TGS \to C: \quad \text{Ticket}_v \)

Once per service session:

(5)  \( C \to V: \quad \text{ID}_c \| \text{Ticket}_v \)

\[ \text{Ticket}_{tgs} = E_{K_{tgs}} [\text{ID}_c \| \text{AD}_c \| \text{ID}_{tgs} \| \text{TS}_1 \| \text{lifetime}_1] \]
\[ \text{TS} = \text{timestamp} \]
\[ \text{Ticket}_v = E_{K_v} [\text{ID}_c \| \text{AD}_c \| \text{ID}_v \| \text{TS}_2 \| \text{lifetime}_2] \]

\( K_c = \) a key derived from user password and already stored in at AS
\( K_{tgs} = \) a key shared only by AS and TGS
A More Secure Authentication Dialogue

• Problems:
  – Lifetime associated with the ticket-granting ticket
    • If to short → repeatedly asked for password
    • If to long → greater opportunity to replay. The threat is that an opponent will steal the ticket and use it before it expires
  – No authentication from server to user

• Solution
  – Provide a proof that the person using a ticket is the person to whom that ticket was issued (use authenticator)
  – One more step: authentication from server to user (mutual authentication)
Kerberos v4 Overview

- a basic third-party authentication scheme
- have an Authentication Server (AS)
  - users initially negotiate with AS to identify self
  - AS provides a non-corruptible authentication credential (ticket granting ticket TGT)
- have a Ticket Granting server (TGS)
  - users subsequently request access to other services from TGS on basis of users TGT
Kerberos v4 Dialogue

1. obtain ticket granting ticket from AS
   • once per session
2. obtain service granting ticket from TGT
   • for each distinct service required
3. client/server exchange to obtain service
   • on every service request
## Table 4.1
**Summary of Kerberos Version 4 Message Exchanges**

1. **C → AS**  
   \[ ID_c \| ID_{tgs} \| TS_1 \]

2. **AS → C**  
   \[ E(K_{c}, [K_{c,tgs} \| ID_{tgs} \| TS_2 \| Lifetime_2 \| Ticket_{tgs}]) \]
   \[ Ticket_{tgs} = E(K_{tgs}, [K_{c,tgs} \| ID_C \| AD_C \| ID_{tgs} \| TS_2 \| Lifetime_2]) \]

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

3. **C → TGS**  
   \[ ID_v \| Ticket_{tgs} \| Authenticator_c \]

4. **TGS → C**  
   \[ E(K_{c,tgs}, [K_{c,v} \| ID_v \| TS_4 \| Ticket_v]) \]
   \[ Ticket_{tgs} = E(K_{tgs}, [K_{c,tgs} \| ID_C \| AD_C \| ID_{tgs} \| TS_2 \| Lifetime_2]) \]
   \[ Ticket_v = E(K_{v}, [K_{c,v} \| ID_C \| AD_C \| ID_v \| TS_4 \| Lifetime_4]) \]
   \[ Authenticator_c = E(K_{c,tgs}, [ID_C \| AD_C \| TS_3]) \]

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

5. **C → V**  
   \[ Ticket_v \| Authenticator_c \]

6. **V → C**  
   \[ E(K_{c,v}, [TS_5 + 1]) \] (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 Exchange to obtain service
1. User logs on to workstation and requests service on host.

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.

5. Workstation sends ticket and authenticator to host.

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 application server.

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

Figure 4.1 Overview of Kerberos
Figure 4.2 Kerberos Exchanges
<table>
<thead>
<tr>
<th>Message (1)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$ID_C$</td>
<td>Client requests ticket-granting ticket.</td>
</tr>
<tr>
<td>$ID_{tgs}$</td>
<td>Tells AS identity of user from this client.</td>
</tr>
<tr>
<td>$TS_1$</td>
<td>Tells AS that user requests access to TGS.</td>
</tr>
<tr>
<td></td>
<td>Allows AS to verify that client's clock is</td>
</tr>
<tr>
<td></td>
<td>synchronized with that of AS.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Message (2)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$K_c$</td>
<td>AS returns ticket-granting ticket.</td>
</tr>
<tr>
<td></td>
<td>Encryption is based on user's password, enabling</td>
</tr>
<tr>
<td></td>
<td>AS and client to verify password, and protecting</td>
</tr>
<tr>
<td></td>
<td>contents of message (2).</td>
</tr>
<tr>
<td>$K_{c,tgs}$</td>
<td>Copy of session key accessible to client created</td>
</tr>
<tr>
<td></td>
<td>by AS to permit secure exchange between client</td>
</tr>
<tr>
<td></td>
<td>and TGS without requiring them to share a</td>
</tr>
<tr>
<td></td>
<td>permanent key.</td>
</tr>
<tr>
<td>$ID_{tgs}$</td>
<td>Confirms that this ticket is for the TGS.</td>
</tr>
<tr>
<td>$TS_2$</td>
<td>Informs client of time this ticket was issued.</td>
</tr>
<tr>
<td>$Lifetime_2$</td>
<td>Informs client of the lifetime of this ticket.</td>
</tr>
<tr>
<td>$Ticket_{tgs}$</td>
<td>Ticket to be used by client to access TGS.</td>
</tr>
</tbody>
</table>

(a) Authentication Service Exchange
Table 4.2 Rationale for the Elements of the Kerberos Version 4 Protocol
(page 2 of 3)

<table>
<thead>
<tr>
<th>Message (3)</th>
<th>Client requests service-granting ticket.</th>
</tr>
</thead>
<tbody>
<tr>
<td>$ID_V$</td>
<td>Tells TGS that user requests access to server $V$.</td>
</tr>
<tr>
<td>$Ticket_{tgs}$</td>
<td>Assures TGS that this user has been authenticated by AS.</td>
</tr>
<tr>
<td>$Authenticator_c$</td>
<td>Generated by client to validate ticket.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Message (4)</th>
<th>TGS returns service-granting ticket.</th>
</tr>
</thead>
<tbody>
<tr>
<td>$K_{c,jgs}$</td>
<td>Key shared only by $C$ and TGS protects contents of message (4).</td>
</tr>
<tr>
<td>$K_{c,v}$</td>
<td>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.</td>
</tr>
<tr>
<td>$ID_V$</td>
<td>Confirms that this ticket is for server $V$.</td>
</tr>
<tr>
<td>$TS_A$</td>
<td>Informs client of time this ticket was issued.</td>
</tr>
<tr>
<td>$Ticket_V$</td>
<td>Ticket to be used by client to access server $V$.</td>
</tr>
<tr>
<td>$Ticket_{tgs}$</td>
<td>Reusable so that user does not have to reenter password.</td>
</tr>
<tr>
<td>$K_{tgs}$</td>
<td>Ticket is encrypted with key known only to AS and TGS, to prevent Tampering.</td>
</tr>
<tr>
<td>$K_{c,jgs}$</td>
<td>Copy of session key accessible to TGS used to decrypt authenticator, thereby authenticating ticket.</td>
</tr>
<tr>
<td>$ID_C$</td>
<td>Indicates the rightful owner of this ticket.</td>
</tr>
<tr>
<td>$AD_C$</td>
<td>Prevents use of ticket from workstation other than one that initially requested the ticket.</td>
</tr>
<tr>
<td>$ID_{tgs}$</td>
<td>Assures server that it has decrypted ticket properly.</td>
</tr>
<tr>
<td>$TS_2$</td>
<td>Informs TGS of time this ticket was issued.</td>
</tr>
<tr>
<td>$Lifetime_2$</td>
<td>Prevents replay after ticket has expired.</td>
</tr>
<tr>
<td>$Authenticator_c$</td>
<td>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.</td>
</tr>
<tr>
<td>$K_{c,jgs}$</td>
<td>Authenticator is encrypted with key known only to client and TGS, to prevent tampering.</td>
</tr>
<tr>
<td>$ID_C$</td>
<td>Must match ID in ticket to authenticate ticket.</td>
</tr>
<tr>
<td>$AD_C$</td>
<td>Must match address in ticket to authenticate ticket.</td>
</tr>
<tr>
<td>$TS_3$</td>
<td>Informs TGS of time this authenticator was generated.</td>
</tr>
</tbody>
</table>

(b) Ticket-Granting Service Exchange
### Table 4.2 Rationale for the Elements of the Kerberos Version 4 Protocol (page 3 of 3)

<table>
<thead>
<tr>
<th>Message (5)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Ticket_v$</td>
<td>Assures server that this user has been authenticated by AS.</td>
</tr>
<tr>
<td>$Authenticator_c$</td>
<td>Generated by client to validate ticket.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Message (6)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$K_{c,v}$</td>
<td>Assures C that this message is from V.</td>
</tr>
<tr>
<td>$TS_5 + 1$</td>
<td>Assures C that this is not a replay of an old reply.</td>
</tr>
</tbody>
</table>

- $Ticket_v$: 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.
- $TS_A$: Informs server of time this ticket was issued.
- $Lifetime_A$: 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.

(c) Client/Server Authentication Exchange
Kerberos Realms

• a Kerberos environment consists of:
  – a Kerberos server
  – a number of clients, all registered with server
  – application servers, sharing keys with server

• this is termed a realm
  – typically a single administrative domain

• if have multiple realms, their Kerberos servers must share keys and trust
Kerberos Realms

1. request ticket for local TGS
2. ticket for local TGS
3. request ticket for remote TGS
4. ticket for remote TGS
5. request ticket for remote service
6. ticket for remote service
7. request remote service
Kerberos Version 5

• developed in mid 1990’s
• specified as Internet standard RFC 1510
• provides improvements over v4
  – addresses environmental shortcomings
    • encryption alg, network protocol, byte order, ticket lifetime, authentication forwarding, interrealm auth
  – and technical deficiencies
    • double encryption, non-std mode of use, session keys, password attacks
Difference Between Version 4 and 5

- Encryption system dependence (V.4 DES)
- Internet protocol dependence
- Message byte ordering
- Ticket lifetime
- Authentication forwarding
- Interrealm authentication
### Table 4.3 Summary of Kerberos Version 5 Message Exchanges

<table>
<thead>
<tr>
<th>Step</th>
<th>Exchange Description</th>
<th>Message</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>C → AS</td>
<td>Options, IDc, Realmc, IDtg, Times, Nonce1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>AS → C</td>
<td>Realmc, IDC, Tickettg, E(Kc, [Kc,tgs, Times, Nonce1, Realmtg, IDtg])</td>
<td>Tickettg = E(Ktgs, [Flags, Kc,tgs, Realmc, IDC, ADC, Times])</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(a) Authentication Service Exchange to obtain ticket-granting ticket</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>C → TGS</td>
<td>Options, IDv, Times, Nonce2, Tickettg, Authenticatorc</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>TGS → C</td>
<td>Realmc, IDC, Ticketv, E(Kc,tgs, [Kc,v, Times, Nonce2, Realmv, IDv])</td>
<td>Ticketv = E(Kv, [Flags, Kc,v, Realmc, IDC, ADC, Times])</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Authenticatorc = E(Kc,tgs, [IDC, Realmc, TS1])</td>
<td>(b) Ticket-Granting Service Exchange to obtain service-granting ticket</td>
</tr>
<tr>
<td>5</td>
<td>C → V</td>
<td>Options, Ticketv, Authenticatorc</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>V → C</td>
<td>E_{Kc,v} [TS2, Subkey, Seq#]</td>
<td>Ticketv = E(Kv, [Flags, Kc,v, Realmc, IDC, ADC, Times])</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Authenticatorc = E(Kc,v, [IDC, Realmc, TS2, Subkey, Seq#])</td>
<td>(c) Client/Server Authentication Exchange to obtain service</td>
</tr>
</tbody>
</table>
Kerberos - in practice

• **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 reimplemented locally
One of the major roles of public-key encryption is to address the problem of key distribution.

There are two distinct aspects to the use of public-key encryption in this regard:

- The distribution of public keys
- The use of public-key encryption to distribute secret keys

Public-key certificate

- Consists of a public key plus a user ID of the key owner, with the whole block signed by a trusted third party
- Typically, the third party is a certificate authority (CA) that is trusted by the user community, such as a government agency or a financial institution
- 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
- Anyone needing this user’s public key can obtain the certificate and verify that it is valid by way of the attached trusted signature
Figure 4.5  Simplified Depiction of Essential Elements of Digital Signature Process
Figure 3.13 Man-in-the-Middle Attack
Figure 4.3 Public-Key Certificate Use

- **Unsigned certificate**: contains user ID, user's public key
  - Generate hash code of unsigned certificate
  - Encrypt hash code with CA's private key to form signature
  - Create signed digital certificate

- **Signed certificate**
  - Bob's ID information
  - Bob's public key
  - CA information

- **Recipient can verify signature by comparing hash code values**
  - Decrypt signature with CA's public key to recover hash code
  - Use certificate to verify Bob's public key
X.509 Authentication Service

• part of CCITT X.500 directory service standards
  – distributed servers maintaining user info database
• defines framework for authentication services
  – directory may store public-key certificates
  – with public key of user signed by certification authority
• also defines authentication protocols
• uses public-key crypto & digital signatures
  – algorithms not standardized, but RSA recommended
• X.509 certificates are widely used
X.509 Authentication Service

- Distributed set of servers that maintains a database about users.
- Each certificate contains the public key of a user and is signed with the private key of a CA.
- Is used in S/MIME, IP Security, SSL/TLS and SET.
- RSA is recommended to use.
X.509 Certificates

• issued by a Certification Authority (CA), containing:
  – version (1, 2, or 3)
  – serial number (unique within CA) identifying certificate
  – signature algorithm identifier
  – issuer X.500 name (CA)
  – period of validity (from - to dates)
  – subject X.500 name (name of owner)
  – subject public-key info (algorithm, parameters, key)
  – issuer unique identifier (v2+)
  – subject unique identifier (v2+)
  – extension fields (v3)
  – signature (of hash of all fields in certificate)

• notation CA<<A>> denotes certificate for A signed by CA
X.509 Certificates
Obtaining a Certificate

- any user with access to CA can get any certificate from it
- only the CA can modify a certificate
- because cannot be forged, certificates can be placed in a public directory
CA Hierarchy

• if both users share a common CA then they are assumed to know its public key
• otherwise CA's must form a hierarchy
• use certificates linking members of hierarchy to validate other CA's
  – each CA has certificates for clients (forward) and parent (backward)
• each client trusts parents certificates
• enable verification of any certificate from one CA by users of all other CAs in hierarchy
A acquires B certificate using chain: $X<<W>>W<<V>>V<<Y>>Y<<Z>>Z<<B>>$
B acquires A certificate using chain: $Z<<Y>>Y<<V>>V<<W>>W<<X>>X<<A>>$
Certificate Revocation

• certificates have a period of validity
• may need to revoke before expiry, eg:
  1. user's private key is compromised
  2. user is no longer certified by this CA
  3. CA's certificate is compromised
• CA’s maintain list of revoked certificates
  – the Certificate Revocation List (CRL)
• users should check certificates with CA’s CRL
Authentication Procedures

• X.509 includes three alternative authentication procedures:
  • One-Way Authentication
  • Two-Way Authentication
  • Three-Way Authentication
  • all use public-key signatures
One-Way Authentication

• 1 message (A->B) used to establish
  – the identity of A and that message is from A
  – message was intended for B
  – integrity & originality of message

• message must include timestamp, nonce, B's identity and is signed by A

• may include additional info for B
  – eg session key
Two-Way Authentication

• 2 messages (A->B, B->A) which also establishes in addition:
  – the identity of B and that reply is from B
  – that reply is intended for A
  – integrity & originality of reply

• reply includes original nonce from A, also timestamp and nonce from B

• may include additional info for A
Three-Way Authentication

- 3 messages (A->B, B->A, A->B) which enables above authentication without synchronized clocks
- has reply from A back to B containing signed copy of nonce from B
- means that timestamps need not be checked or relied upon
Authentication Procedures

(a) One-way authentication

(b) Two-way authentication

(c) Three-way authentication

Figure 4.5  X.509 Strong Authentication Procedures
X.509 Version 3

- has been recognized that additional information is needed in a certificate
  - email/URL, policy details, usage constraints
- rather than explicitly naming new fields, defined a general extension method
- extensions consist of:
  - extension identifier
  - criticality indicator
  - extension value
Certificate Extensions

- key and policy information
  - convey info about subject & issuer keys, plus indicators of certificate policy
- certificate subject and issuer attributes
  - support alternative names, in alternative formats for certificate subject and/or issuer
- certificate path constraints
  - allow constraints on use of certificates by other CA’s
Public Key Infrastructure
Summary

• have considered:
  – Kerberos trusted key server system
  – X.509 authentication and certificates