CS 556 – Computer Security
Fall 2012

Dr. Indrajit Ray
Email: indrajit@cs.colostate.edu

Department of Computer Science
Colorado State University
Fort Collins, CO 80523, USA
MUTUAL AUTHENTICATION

Secret Key Based Mediated Authentication

KERBEROS

CERTIFICATE-BASED MUTUAL AUTHENTICATION PROTOCOLS

MUTUAL AUTHENTICATION
Mutual Authentication

Secret Key Based Mediated Authentication

Kerberos Certificate-Based Mutual Authentication Protocols
Mutual Authentication

- Problem – How to share key
  - This is more critical than in the case of secret key based encryption schemes because authentication is a more fundamental issue

- Solution – Mediated Authentication
  - Secret key based – Needham-Schroeder and Kerberos
  - Public key based – X.509
Secret Key Based Mediated Authentication
Needham–Schroeder Protocol

- Mediated authentication and key exchange protocol based on trusted third party
- After authentication communicating parties share a secret key that can be used in future secure exchanges
**Needham-Schroeder Protocol**

**Message 1:** Alice → TP  
A, B, N_A

**Message 2:** TP → Alice  
{N_A, B, K_{AB}, \{K_{AB}, A\}K_{BS}\}K_{AS}

**Message 3:** Alice → Bob  
{K_{AB}, A\}K_{BS}

**Message 4:** Bob → Alice  
{N_B\}K_{AB}

**Message 5:** Alice → Bob  
{N_B - 1\}K_{AB}
**Protocol Analysis**

- **Problem** – suffers from replay attack
  - Message 3 can be subject to replay attack with an old compromised session key by an active attacker

- **Solutions**
  - Include a timestamp in messages 1 to 3, which requires synchronised clocks
  - Have Alice ask Bob for a random value $J_B$ to be sent to $S$ for return in $\{K_{AB}, A, J_B\}K_{BS}$
Kerberos Mutual Authentication

Secret Key Based Mediated Authentication

Kerberos Certificate-Based Mutual Authentication Protocols
Kerberos – Improved Needham-Schroeder

- Designed as part of Project Athena at MIT
- Provides the means of authenticating workstation users (clients) to server and sharing a session key
- Uses the ticket approach
  - Client authenticates itself to an authentication server
  - Authentication Server gives ticket to client
  - Client uses ticket to get authenticated
Access control may be provided for

- Each computing resource
- In either local or remote network (realm)

Has a Key Distribution Center (KDC) – or Kerberos Server – containing

- A database of principals (clients and services)
- Secret encryption keys for these shared with KDC
Physical Security

- Client Workstations
  - None, so cannot be trusted
- Servers
  - Moderately secure rooms with moderately diligent administration
- Kerberos
  - Highly secure room with extremely diligent system administration
Trust – Bilateral RHHOSTS Model

A trusts B
A will allow users logged on to B to log on to A without a password
Trust – Consolidated Kerberos Model

Mutual Authentication
Secret Key Based Mediated Authentication
Kerberos Certificate-Based Mutual Authentication Protocols

Dr. Indrajit Ray, Computer Science Department

CS 556 - Computer Security - © 2012 Colorado State University – 14 / 50
Trust – Consolidated Kerberos Model

- Breaking into one host provides a cracker no advantage in breaking into other hosts
- Authentication systems can be viewed as trust propagation systems
  - The Rhosts model is a tangled web model
  - The Kerberos model is a centralized star model
What Kerberos Does Not Do

- Makes no sense on an isolated system
- Does not mean that host security can be allowed to slip
- Does not protect against Trojan Horses
- Does not protect against viruses/worms
Kerberos Design Goals

● Impeccability
  ♦ No cleartext passwords on the network
  ♦ No client passwords on servers (server though must store secret server key)
  ♦ Minimum exposure of client key on workstations

● Containment
  ♦ Compromise affects only one client (or server)
  ♦ Limited authentication lifetime (8 hours, 24 hours or more)
Kerberos Design Goals (continued)

- **Transparency**
  - Password required only at login
  - Minimum modification to existing applications
Kerberos Design Decisions

- Uses timestamps to avoid replays
  - Requires time synchronized within a small window (5 minutes)
- Uses DES-based symmetric key cryptography
- Stateless
## Notation

<table>
<thead>
<tr>
<th>Symbol used</th>
<th>What it means</th>
</tr>
</thead>
<tbody>
<tr>
<td>c</td>
<td>client principal</td>
</tr>
<tr>
<td>s</td>
<td>server principal</td>
</tr>
<tr>
<td>KDC</td>
<td>Kerberos server</td>
</tr>
<tr>
<td>TGS</td>
<td>Ticket granting server</td>
</tr>
<tr>
<td>$K_x$</td>
<td>private key of x</td>
</tr>
<tr>
<td>$K_{c,s}$</td>
<td>session key for c &amp; s</td>
</tr>
<tr>
<td>${\text{info}}K_x$</td>
<td>string info encrypted in $K_x$</td>
</tr>
<tr>
<td>$T_{c,s}$</td>
<td>ticket for c to use s</td>
</tr>
<tr>
<td>$A_c$</td>
<td>authenticator for c</td>
</tr>
<tr>
<td>addr</td>
<td>client's IP address</td>
</tr>
<tr>
<td>Message in network</td>
<td>Structure of message</td>
</tr>
<tr>
<td>--------------------</td>
<td>----------------------</td>
</tr>
<tr>
<td>$T_{c,s}$ – ticket for $c$ to use $s$</td>
<td>${s,c,addr,timestamp,lifetime,K_{c,s}}$</td>
</tr>
<tr>
<td>$A_c$ – authenticator for $c$</td>
<td>${c,addr,timestamp}$</td>
</tr>
</tbody>
</table>
Session Key Distribution

1. Client → KDC: c, s, n
2. KDC → Client: \{K_{c,s}, n, T_{c,s}\}K_c
3. Client → Server: A_c, T_{c,s}
For user to server authentication, client key is the user's password, converted to a DES Key via a publicly known algorithm.
Trust In Workstation

- Untrusted client workstation has $K_c$
- Workstation is expected to delete it after decrypting message in step 2
- Compromised workstation can compromise one user
- Compromise does not propagate to other users
Authentication Failures

- Ticket decryption by server yields garbage
- Ticket timed out
- Wrong source IP address
- Replay attempt
Kerberos Impersonation

- Active intruder on the network can cause denial of service by impersonation of Kerberos IP address
- Network monitoring at multiple points can help detect such an attack by observing IP impersonation
Kerberos Reliability

- Availability enhanced by keeping slave Kerberos servers with replicas of the Kerberos database
- Slave databases are read only
- Simple propagation of updates from master to slaves ensure consistency
Use of the Session Key

- Kerberos establishes a session key $K_{c,s}$
- Session key can be used by the applications for
  - Client to server authentication (no additional step required in the protocol)
  - Mutual authentication (requires fourth message from server to client $\{f(A_c)\}K_{c,s}$, where $f$ is some publicly known function)
  - Message confidentiality using $K_{c,s}$
  - Message integrity using $K_{c,s}$
Ticket Granting Service – Why?

- Problem: Transparency
  - User should provide password once upon initial login and should not be asked for it on every service request

- Solution: Ticket Granting Service
  - Store session key on workstation in lieu of password
Ticket Granting Service

- Kerberos server authenticates client and provides an initial ticket
- $K_c$ deleted from workstation after this exchange (have to trust the workstation)
- Client submits the initial ticket to a Ticket Granting server to get a ticket for a particular service
Getting a Service Ticket

**Kerberos Certificate-Based Mutual Authentication Protocols**

1. Client → KDC: c, tgs, n
2. KDC → Client: \( \{K_{c,tgs},n,\{T_{c,tgs}\}K_{tgs}\}K_c\)
3. Client → TGS: \( \{A_c\}K_{c,tgs},\{T_{c,tgs}\}K_{tgs},s,n\)
4. TGS → Client: \( \{K_{c,s},n,\{T_{c,s}\}K_s\}K_{c,tgs}\)
5. Client → Server: \( \{A_c\}K_{c,s},\{T_{c,s}\}K_s\)
Ticket Lifetime

- Lifetime is minimum of
  - Requested lifetime
  - Maximum lifetime for requesting principal
  - Maximum lifetime for requesting service
  - Maximum lifetime of ticket granting ticket

- Maximum lifetime is 21.5 hours
Kerberos Naming

- Users and servers have same name format:
  - name.instance@realm

- Mapping of Kerberos authentication names to local system names is left up to the service provider
Kerberos V5 Enhancements

- **Naming**
  - Kerberos V5 supports V4 names, but also provides for other naming structures such as X.500

- **Timestamps**
  - V4 timestamps are Unix timestamps (seconds since 1/1/1970). V5 timestamps are in OSI ASN.1 format
Kerberos V5 Enhancements

- **Ticket lifetime**
  - V4 tickets valid from time of issue to expiry time, and limited to 21.5 hours
  - V5 tickets have start and end timestamps; maximum lifetime can be set by realm
- V5 tickets are renewable, so service can be maintained beyond maximum ticket lifetime
V5 Ticket Renewal

- Tickets can be renewed until minimum of:
  - Requested end time
  - Start time + requesting principal’s maximum renewable lifetime
  - Start time + requested server’s maximum renewable lifetime
  - Start time + maximum renewable lifetime of realm
Kerberos V5 Preauthentication

- In Kerberos V4 anybody can request a TGT for any user
  - TGT is encrypted with user’s secret key (password)
  - Perform an offline attack on TGT by trying to decrypt it with different passwords
- Kerberos V5 introduces preauthentication to make sure only the true user requests TGT
  - Simplest preauthentication involves user encrypting a timestamp with her secret key. If server is able to decrypt message and get timestamp within a window, preauthentication is successful.
  - Other types of preauthentication protocols also possible
Kerberos in Practice

- Kerberos server needs to be secured and on-line
  - Server compromise is a major source of vulnerability
- Currently two Kerberos versions
  - V4: restricted to single realm
  - V5: allows inter-realm authentication
**MS Windows Authentication**

- Based on Kerberos V5
  - 128 bit RC4-HMAC
  - 56 bit DES-CBC-CRC
  - 56 bit DES-CBC-MD5
- Has extensions for using public-key certificates
Windows SSPI Architecture

**Secret Key Based Mediated Authentication**
- Kerberos
- Certificate-Based Mutual Authentication Protocols

---

**Application Layer**
- Application
  - RPC
- .NET Application
  - .NET Framework
- Internet Explorer

**SSPI Layer**
- Security Service Provider Interface (GSSAPI – RFC 2743)
  - Kerberos
  - NTLM
  - Digest
  - Schannel
  - Negotiate
  - Other

**SSP Layer**
- Challenge-Response scheme for backward compatibility
- For integration with LDAP protocols
- For negotiating specific protocols
- Implements SSL and TLS protocols
CERTIFICATE-BASED MUTUAL AUTHENTICATION PROTOCOLS
**X.509 – Directory Authentication Service**

- Part of CCITT X.500 directory services
- Defines framework for authentication services
- Directory may store public-key certificates
- Uses public-key cryptography and digital signatures
- Algorithms not standardised but RSA is recommended
- Does not require physically secured on-line servers – Advantage over Kerberos
Symbols Used

- $E_x\{\}$: indicates encryption of a sequence of data values under the public key of party $x$
- $S_x\{\}$: indicates a sequence of data values together with a signature over those values, using the private key of party $x$
- $ts_{xy}$: a current time-stamp generated by party $x$ to assist party $y$ in detecting replayed messages (may contain both generation and expiry date/time for the message conveying it)
- $nrv_{xy}$: a non-repeating value sent by party $x$ to assist party $y$ in detecting replayed messages
- $key_{xy}$ a secret key generated by $x$ to be used in protecting subsequent communications between $x$ and $y$
X.509 Authentication Exchange

Secret Key Based Mediated Authentication

Kerberos Certificate-Based Mutual Authentication Protocols

Public-key certificate distribution

Optional Message 3

Alice Bob

Message 1
Message 2

Optional Message 3
**X.509 Authentication Exchange**

**Message 1:**

\[ A, S_A\{ts_{AB}, nrv_{AB}, B, E_B\{key_{AB}\}\} \]

- The field \( E_B\{key_{AB}\} \) is optional

- B verifies A’s signature, checks that the identifier B in the message is correct, checks that the time-stamp is current and (optionally if an effective non-repeating value procedure is in use) checks the non-repeating value as protection against replay
Message 2:

\[ S_B\{ts_{BA}, nrv_{BA}, A, nrv_{AB}, E_A\{key_{BA}\}\} \]

- The field \( E_A\{key_{BA}\} \) is optional
- A performs the corresponding set of actions

Optional Message 3: The first two messages are identical except that the time-stamps are not conveyed. The third message is

\[ S_A\{B, nrv_{BA}\} \]
LOCKOut Fortezza Authentication Protocol

- Fortezza is a PCMCIA card developed by the NSA to provide general purpose cryptographic capabilities
  - Has built in implementation for SHA and DSS among other features
  - Tamper evident
- Computes hash functions and digital signatures for authentication
LOCKOut Fortezza Authentication

**MUTUAL AUTHENTICATION**
- Secret Key Based Mediated Authentication

**KERBEROS**
- CERTIFICATE-BASED MUTUAL AUTHENTICATION PROTOCOLS

---

**Diagram:**
- **Hello I am Bob**
- **Hello I am HAL, Prove you are Bob**
- **Server ID**
- **80-bit Challenge**
- **Time Stamp**
- **Constant Prefix**
- **Challenge**
- **Time Stamp**
- **Response**
- **SHA Message Digest**
- **DSS Digital Signature**
- **Bob’s private key**

---

Dr. Indrajit Ray, Computer Science Department
Authenticated Diffie-Hellman

- Three way authentication exchange that combines public-key based mutual authentication with a Diffie-Hellman key derivation exchange
- Has an advantage over Kerberos and X.509 in that compromise of a signature key will not compromise the secrecy of the session key derived previously in conjunction with the authentication
Authenticated Diffie-Hellman

1: Chooses $x_a$

2: $y_B = \alpha^{x_a} \mod p$

3: $y_A = \alpha^{x_b} \mod p$, $E_{K_{AB}}\{S_B\{y_B, y_A\}\}$

4: $E_{K_{AB}}\{S_A\{y_B, y_A\}\}$

$K_{AB} = \alpha^{x_a.x_b} \mod p$

Alice Computates

Bob Computates

Secret Key Based Mediated Authentication

Kerberos Certificate-Based Mutual Authentication Protocols