

## CS 5523 Lecture 24: Introduction to Computer Security

---

- *Policy versus methods*
- *Types of attacks*
- *Levels of reliability*
- *Shared key scenarios*
- *Trusted server scenarios*
- *Public key scenarios*

### Policies versus mechanisms:

---

- *A security policy specifies limits for accessing and sharing resources*
- *A security mechanism enforces or implements a security policy*
- *Cryptography – study of processes that encode and decode messages in order to hide their information. Many security mechanism are based on encryption.*

Figure 7.1  
 Historical context: the evolution of security needs

	1965-75	1975-89	1990-99	Current
<i>Platforms</i>	Multi-user timesharing computers	Distributed systems based on local networks	The Internet, wide-area services	The Internet + mobile devices
<i>Shared resources</i>	Memory, files	Local services (e.g. NFS), local networks	Email, web sites, Internet commerce	Distributed objects, mobile code
<i>Security requirements</i>	User identification and authentication	Protection of services	Strong security for commercial transactions	Access control for individual objects, secure mobile code
<i>Security management environment</i>	Single authority, single authorization database (e.g. /etc/passwd)	Single authority, delegation, replicated authorization databases (e.g. NIS)	Many authorities, no network-wide authorities	Per-activity authorities, groups with shared responsibilities

Instructor's Guide for Coulouris, Dollimore and Kindberg Distributed Systems: Concepts and Design Edn. 3  
 © Addison-Wesley Publishers 2000

## Threats:

- *Leakage – acquisition of information by an authorized recipient*
- *Tampering – unauthorized alteration of information*
- *Vandalism – interference of proper system information without gain to the perp*

## Methods of attack:

---

- *Eavesdropping – obtaining a copy of a message without authorization*
- *Masquerading – sending or receiving messages using someone else's identity*
- *Message tampering – intercepting messages and altering them*
- *Replaying – storing intercepted messages and sending them at a later date*
- *Denial of service – flooding a channel with messages to deny access to others*

*How might these attacks be used to cause damage?*

## Mobile code and security:

---

- *Download code and run it locally*
- *Two common mobile code systems:*
  - ┆ *Java Virtual Machine*
  - ┆ *Microsoft Active X*
- *Java Virtual Machine security:*
  - ┆ *Environment maintains a security manager that can't be replaced*
  - ┆ *Security managers can be set to prevent access to local resources*
  - ┆ *Downloaded classes are stored separately from local classes*
  - ┆ *Byte code is checked for validity*

## Information can leak in unexpected ways:

---

- *Mere existence of a connection*
- *Power used in computation*

## Levels of reliability:

---

*A sends a message to B*

- *Authentication of A to B (B can be sure that A is A)*
- *Tamperproof (B can be sure that the message has not altered)*
- *Secret (A and B can be sure that no one else received the information)*
- *Nonrepudiation (B can hold A legally responsible for the message)*

*What levels of reliability do each of the following transactions need?*

*email, purchase of goods and services, banking transactions, micro-transactions*

## Designing a secure system:

---

■ *Trade-off between cost, inconvenience and threat*

■ *Must assume:*

- *Interfaces are exposed*
- *Networks are insecure*
- *Algorithms and code are available*
- *Attackers may have large resources*
- *Need to minimize the trusted base*

Figure 7.2  
Familiar names for the protagonists in security protocols

---

Alice	First participant
Bob	Second participant
Carol	Participant in three- and four-party protocols
Dave	Participant in four-party protocols
Eve	Eavesdropper
Mallory	Malicious attacker
Sara	A server

Figure 7.3  
Cryptography notations

---

$K_A$	Alice's secret key
$K_B$	Bob's secret key
$K_{AB}$	Secret key shared between Alice and Bob
$K_{Apriv}$	Alice's private key (known only to Alice)
$K_{Apub}$	Alice's public key (published by Alice for all to read)
$\{M\}_K$	Message $M$ encrypted with key $K$
$[M]_K$	Message $M$ signed with key $K$

Instructor's Guide for Coulouris, Dollimore and Kindberg Distributed Systems: Concepts and Design Edn. 3  
© Addison-Wesley Publishers 2000

### Simplified overview of secret key encryption:

---

*Encrypted message:*

$$E(K, M) = \{M\}_K$$

*Decrypted message:*

$$D(K, E(K, M)) = D(K, \{M\}_K) = M$$

*It is hard to get  $M$  from  $\{M\}_K$  without knowing  $K$*

## Scenario 1.

### Secret communication with shared secret key

*Alice and Bob share a secret key  $K_{AB}$ . Alice wants to send a secret message  $M$  to Bob.*

- 1. Alice uses  $K_{AB}$  and an agreed encryption function  $E(K_{AB}, M)$  to encrypt and send message  $M$  to Bob*
- 2. Bob reads the encrypted messages using the corresponding decryption function  $D(K_{AB}, M)$*

*How can Bob and Alice safely get the shared key  $K_{AB}$ ?*

*How can Bob know that  $M$  wasn't a replay?*

## Scenario 2.

### Authenticated communication with a server

*Alice wants to access Bob's files on a local file server. Sara is a trusted authentication server that holds passwords and current secret keys.*

- 1. Alice sends a message to Sara asking for a ticket to access Bob*
  - 2. Sara sends Alice a response encrypted with  $K_A$  that is a ticket encrypted with  $K_B$  and a new secret key  $K_{AB}$  for communication:  
 $\{\{ticket\}_{K_B}, K_{AB}\}_{K_A}$*
  - 3. Alice decrypts response with  $K_A$*
  - 4. Alice sends ticket, her ID and request  $R$  to Bob:  $\{ticket\}_{K_B}, Alice, R$*
  - 5. Bob decrypts ticket using  $K_B$  (the ticket was  $\{K_{AB}, Alice\}_{K_B}$ )*
- This is the simplified scenario for Kerberos.  $K_{AB}$  is the session key.*

## Simplified overview of public key encryption:

Keys come in pairs  $K_1$  and  $K_2$ . Keep one public and one private.  
If you encrypt with  $K_1$ , you can decrypt with  $K_2$  and vice versa:

$$D(K_2, E(K_1, M)) = M$$

and

$$D(K_1, E(K_2, M)) = M$$

## Scenario 3.

### Authenticated communication with public keys

Bob has generated a public/private key pair. There is a trusted authority that gives out key certificates

1. Alice accesses a key distribution center to obtain a public key certificate with Bob's public key. Alice extracts Bob's public key  $K_{Bpub}$
2. Alice creates a new secret key  $K_{AB}$  and encrypts  $\{K_{AB}, \text{known string}\}$  with  $K_{Bpub}$
3. Alice sends  $\{\{\text{unique keyname}\}, \{K_{AB}, \text{known string}\}_{K_{Bpub}}\}$  to Bob.
4. Bob decrypts  $\{K_{AB}, \text{known string}\}_{K_{Bpub}}$  using  $K_{Bpriv}$
5. Bob and Alice now communicate with  $K_{AB}$

This is the scenario for the widely used hybrid cryptographic protocol.

## Scenario 4.

### Digital signatures with a secure digest function

Alice wants to sign document  $M$  so that any recipient can verify it came from Alice. This assumes that Alice has a private-public key pair. A digest is like a checksum.

1. Alice computes a fixed-length digest  $Digest(M)$ .
2. Alice encrypts  $Digest(M)$  with her private key certificate with Bob's public key and makes  $\{M, \{Digest(M)\}_{K_{Apriv}}\}$  available.
3. Bob reads  $\{M, \{Digest(M)\}_{K_{Apriv}}\}$ , extracts  $M$  and computes  $Digest(M)$ .
4. Bob applies  $K_{Apub}$  to  $\{Digest(M)\}_{K_{Apriv}}$  to obtain  $Digest(M)$  and compares the value with his computed value.

Figure 7.4  
Alice's bank account certificate

1. <i>Certificate type</i>	Account number
2. <i>Name</i>	Alice
3. <i>Account</i>	6262626
4. <i>Certifying authority</i>	Bob's Bank
5. <i>Signature</i>	$\{Digest(\text{field 2} + \text{field 3})\}_{K_{Bpriv}}$

Figure 7.5  
Public-key certificate for Bob's Bank

---

1. <i>Certificate type</i>	Public key
2. <i>Name</i>	Bob's Bank
3. <i>Public key</i> :	$K_{Bpub}$
4. <i>Certifying authority</i>	Fred – The Bankers Federation
5. <i>Signature</i>	$\{Digest(field\ 2 + field\ 3)\}_{K_{Fpriv}}$

Instructor's Guide for Coulouris, Dollimore and Kindberg Distributed Systems: Concepts and Design Edn. 3  
© Addison-Wesley Publishers 2000

For next time:

---

■ *Read CDK 8.1-8.2*