Security

Chapter 8
Basic Issues of Security

• Dependability
  – Availability, Reliability, Safety, Maintainability
  – Confidentiality
    • Information is disclosed only to authorized parties
  – Integrity
    • Modification to a system’s assets can be made only in an authorized way
    • System’s assets
      – hardware, software, data, etc
Types of Threats

- **Interception**
  - Unauthorized party has gained access to a service or data
  - (ex) overheard, illegal copy

- **Interruption**
  - Services or data become unavailable, unusable
  - (ex) DoS

- **Modification**
  - Unauthorized changing of data

- **Fabrication**
  - Additional data or activity are generated that would normally not exist
What is first needed?

• Security Policy
  – Description of security requirements
  – which actions the entities in a system are allowed to take
  – Which ones are prohibited
  – Entities
    • Users, services, data, machines, etc
Security Mechanism

- **Encryption**
  - Transforms data into something an attacker cannot understand
  - Provides confidentiality

- **Authentication**
  - Verify the claimed identity
  - (ex) password

- **Authorization**
  - Check permission to actions, data, etc

- **Auditing**
  - Trace which clients accessed what, and which way.
  - (ex) syslog
Example:
Globus Security Architecture

- **Globus**
  - Kinds of virtual organization
  - Resources are located in different administrative domains

- **Security policy for Globus**
  - Multiple organization (administrative domains)
  - Autonomous local operation
  - Global operation
  - Remote authentication
    - Operations between entities in different domains require mutual authentication
    - Global authentication replaces local authentication
  - Controlling access to resources is subject to local security only
  - Users can delegate rights to processes
  - A group of processes in the same domain can share credentials.
Example:
Globus Security Architecture

Globus needs mechanisms for cross-domain authentication!

- Protocol 1: Creation of user proxy
- Protocol 2: Allocation of a resource by the user in a remote domain
- Protocol 3: Allocation of a resource by a process in remote domain
- Protocol 4: Making user known in remote domain

Resource proxy
- Has permission to act on behalf of a user for a limited period times
- Translate global operation into local operation
Design Issues (1)

• **Focus of Control**
  a. Protection of the data
     • Protection against invalid operation
     • Ensure data integrity
  b. Protection against unauthorized invocation
     • Specify exactly which operations may be invoked, and by whom.
  c. Protection against unauthorized users
     • Only specific people can access
Design Issues (2)

Three approaches for protection against security threats

a) Protection against invalid operations

b) Protection against unauthorized invocations

c) Protection against unauthorized users
Design Issues (3)

- Layering of Security Mechanisms (1/3)
  - At which level security mechanisms should be placed

<The logical organization of a distributed system into several layers.>
Design Issues (4)

- **Layering of Security Mechanisms (2/3)**
  - (ex) SMDS
    - Switched Multi-megabit Data Service
    - Link-level backbone connecting various LAN at geographically dispersed site

Security is provided by placing encryption device At each SMDS router

Secure communication is not provided at the same site

< Several sites connected through a wide-area backbone service >
Design Issues (5)

• Layering of Security Mechanisms (3/3)
  – If Alice can’t trust the security of intersite traffic.
  • Alice should decide to take her own measures by using transport-level security services such as SSL.

Security is provided by placing encryption device
At each SMDS router

< Several sites connected through a wide-area backbone service >
Design Issues (6)

• Distribution of Security Mechanisms
  – TCB (Trusted Computing Base)
    • Set of all security mechanisms that are needed to enforce a security policy
  – Separate security services
    • Reduce the TCB to a relatively small number of machines and software components
  – RISSC (Reduced Interfaces for Secure System Components)
    • Prevent clients and their applications direct access to critical services

< The principle of RISSC as applied to secure distributed systems.>
Design Issues (7)

- **Simplicity**
  - Underlying mechanisms need to be relatively simple and easy to understand
  - But, simple mechanisms are often not sufficient
Cryptography (1)

- Cryptographic techniques
  - Fundamental to security in distributed systems
  - Encrypt sending message to protect against security threat

<Intruders and eavesdroppers in communication.>
Cryptography (2)

- **Symmetric cryptosystem**
  - The same key is used to encrypt and decrypt message.
  - \( P=D_K(E_K(P)) \)
  - Shared key must be kept secret

- **Asymmetric cryptosystem**
  - Key for encryption and decryption are different
  - \( P=D_{K_D}(E_{K_E}(P)) \)
  - public/private keys system

- **Notation**
  - \( K_{A,B} \) : Secret key shared by A and B
  - \( K_A^- \) : Private key of A
  - \( K_A^+ \) : Public key of A
Cryptography (3)

• **Hash function (H)**
  – Application of cryptography
  – H takes a messages m of arbitrary length as input
  – Produce a bit string h having a fixed length as output
  – \( h = H(m) \)
  – One-way function
  – Weak collision resistance property
    • Computationally infeasible to find another m’ s.t. \( H(m) = H(m’) \)
  – Strong collision resistance property
    • Computationally infeasible to find any two m and m’, s.t. \( H(m) = H(m’) \)
Symmetric Cryptosystems: DES (1)

- **Data Encryption Standard (DES)**
  - Operate on 64-bit blocks of data
  - Block is transformed into an encrypted block of output in 16 rounds
  - Quite simple, but difficult to break
  - brute-force attack is possible
  - Use 3DES makes system to be more safe
Symmetric Cryptosystems: DES (2)

a) The principle of DES
b) Outline of one encryption round
Symmetric Cryptosystems: DES (3)

Details of per-round key generation in DES.

56-bit key

Initial permutation

28-bit string | 28-bit string

Rotate left | Rotate right

Extract 24 bits | Extract 24 bits

48-bit key

Used for next round

Details of per-round key generation in DES.
Public-Key Cryptosystems: RSA

• Generating the private and public key
  1. Choose two very large prime numbers, \( p \) and \( q \)
  2. Compute \( n = p \times q \) and \( z = (p - 1) \times (q - 1) \)
  3. Choose a number \( d \) that is relatively prime to \( z \)
  4. Compute the number \( e \) such that \( e \times d = 1 \mod z \)
     – \( d \) can be used for decryption
     – \( e \) can be used for encryption

• Characteristics
  – Computationally more complex
  – Encrypting messages is 100~1000 times slower than DES
  – Use RAS to exchange only shared keys
  – Use DES to actually encrypt “normal data”
Fermat’s Numbers

• **Ferma’s theorem**
  – If p is a prime and a is a positive integer not divisible by p, then
    
    • \( a^{p-1} = 1 \mod p \)

• Proof:
  – Listing the first p-1 positive multiples of a: \( \{a, 2a, 3a, \ldots \} \)
  – Multiply all these congruences together and we find
  – \( a \times 2a \times 3a \times \ldots \times (p-1)a = 1 \times 2 \times 3 \times \ldots \times (p-1) \mod p \)
  – Therefore, \( a^{p-1}(p-1)! = (p-1)! \mod p \)
  – Divide both side by \( (p-1)! \) to complete the proof. Q.E.D.
Fermat’s Numbers

- **Corollary to Fermat’s theorem**
  - If \( p \) is a prime and \( a \) is a positive integer not divisible by \( p \), then
    - \( a^p = a \mod p \)
- **Example:**
  - \( a = 3, p = 5 \) then \( 3^5 = 3 \mod 5 \)
  - \( A = 10, p = 5 \) then \( 10^5 = 10 \mod 5 \)
Euler’s Numbers

• **Euler’s Totient Function** $\phi(n)$
  - $\phi(n) = \text{the number of positive integers less than } n \text{ and relatively prime to } n$.
  - $\phi(n) = 1$.
  - $\phi(p) = p-1$, if $p$ is a prime.
  - $\phi(pq) = (p-1)(q-1)$, if $p, q$ are prime and not same.

• Proof:
  - The residues not relatively prime to $n$ are $0, \{p, 2p, .. (q-1)p\}, \{q, 2q, .. (p-1)q\}$.
  - Hence, $\phi(pq) = pq - [1+(q-1)+(p-1)] = (p-1)(q-1) = \phi(p)\phi(q)$. 
Euler’s Theorem

• Generalization of Fermat’s little theorem
  – For every a and n that are relatively prime,
  – \( a^{\phi(n)} = 1 \mod n \)

• Proof:
  – Case 1: If n is a prime, then Fermat’s theorem holds, \( \phi(p) = p-1 \)
  – Case 2: For any integer n,
    • Consider the set of \( \phi(n) \), \( R = \{x_1, x_2, \ldots, x_{\phi(n)}\} \)
    • Multiply each element by a modulo n, then result in a permutation of R
    • \( S = \{ax_1 \mod n, ax_2 \mod n, \ldots, ax_{\phi(n)} \mod n\} \)
    • Therefore, two products are congruent:
      – \( (ax_1)(ax_2) \ldots (ax_{\phi(n)}) \mod n = x_1x_2 \ldots x_{\phi(n)} \)
    • Dividing by the left hand side proves the theorem. Q.E.D.
Public-Key Cryptosystems: RSA

- Rivest/Shamir/Adleman (RSA) in 1977 MIT
- Algorithm
  - Public key: \( n = p \times q, e \) where \((e, \phi(n)) = 1\).
  - Private key: \( d \) such that \( ed = a \mod \phi(n) \)
    where \( \phi(n) = (p-1)(q-1) \).
  - Encryption: \( c = m^e \mod n \).
  - Encryption: \( m = c^d \mod n \).
  \[ c^d = m^{ed} = m^{k\phi(n)+1} \mod n \]
Workings of RSA

- **Receiver**:
  - Picks two large primes $p$ and $q$ and computes $n=pq$.
  - Picks a number $e$ that has no common factors with $p-1$ and $q-1$.
  - Finds a number $d$ such that $de = 1 \mod (p-1)(q-1)$.
  - Makes $(n,e)$ public.

- **Example**:
  - Let $p = 61$, $q = 53$, $n = pq = 3233$
  - Let $e = 17$
  - Find $d = 2753$
Workings of RSA

• **How to find \(d\) ?**
  • \(17 \times d = 1 \mod (61-1)(53-1) = 1 \mod 3120.\)
  • \(3120 = 183 \times 17 + 9\)
    
    \[
    17 = 1 \times 9 + 8
    \]
    
    \[
    9 = 1 \times 8 + 1
    \]
  • \(1 = 9 - 8\)
    
    \[
    = 9 - (17 - 9)
    \]
    
    \[
    = 2 \times 9 - 17
    \]
    
    \[
    = 2(3120 - 17 \times 183) - 17
    \]
    
    \[
    = 2 \times 3120 - 367 \times 17
    \]
  • \(d = (– 367) \mod 3120 = 2753.\)
Workings of RSA

• Sender:
  – Converts the message to a string of digits using the usual letter number correspondence (A-00, B-01, C-02, Z-25)
  – Breaks up the message into blocks of digits P1, P2, … Pr
  – Calculates and sends \( C_i = P_i^e \) (mod n)

• Example:
  – Plain text 123
  – Encrypt(123) = 123^{17} \mod 3233 = 855
Workings of RSA

• **Receiver :**
  
  • For each $C_i$, compute $C_i^d \pmod{n}$
  
  • $C_i^d = P_i \pmod{n}$

• **Example :**
  
  – Plain text 123
  – $\text{Encrypt}(123) = 123^{17} \pmod{3233} = 855$
  – $\text{Decrypt}(855) = 855^{2753} \pmod{3233} = 123$
Workings of RSA

• How to calculate $855^{2753} \mod 3233$?
  
  1. $2753 = 1 + 64 + 128 + 512 + 2048$
  2. $2753 = (101011000001)_{\text{base 2}}$
  3. Therefore,

$$855^{2753} = 855^{(1 + 64 + 128 + 512 + 2048)} = 855 \times 916 \times 1709 \times 1160 \times 2197 = 123 \mod 3233.$$ 

• $855^1 = 855 \quad 855^2 = 367 \pmod{3233}$
• $855^4 = 367^2 = 2136 \pmod{3233}$
• $855^8 = 2136^2 = 733 \pmod{3233}, \; 611, \; 1526, \; 916, \ldots$
Hash Functions: MD5 (1)

- **MD5**
  - Hash function for computing a 128-bit, fixed length message digest from an arbitrary length binary input string

The structure of MD5
The 16 iterations during the first round in a phase in MD5.

<table>
<thead>
<tr>
<th>Iterations 1-8</th>
<th>Iterations 9-16</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p \leftarrow (p + F(q,r,s) + b_0 + C_1) \equiv 7$</td>
<td>$p \leftarrow (p + F(q,r,s) + b_8 + C_9) \equiv 7$</td>
</tr>
<tr>
<td>$s \leftarrow (s + F(p,q,r) + b_1 + C_2) \equiv 12$</td>
<td>$s \leftarrow (s + F(p,q,r) + b_9 + C_{10}) \equiv 12$</td>
</tr>
<tr>
<td>$r \leftarrow (r + F(s,p,q) + b_2 + C_3) \equiv 17$</td>
<td>$r \leftarrow (r + F(s,p,q) + b_{10} + C_{11}) \equiv 17$</td>
</tr>
<tr>
<td>$q \leftarrow (q + F(r,s,p) + b_3 + C_4) \equiv 22$</td>
<td>$q \leftarrow (q + F(r,s,p) + b_{11} + C_{12}) \equiv 22$</td>
</tr>
<tr>
<td>$p \leftarrow (p + F(q,r,s) + b_4 + C_5) \equiv 7$</td>
<td>$p \leftarrow (p + F(q,r,s) + b_{12} + C_{13}) \equiv 7$</td>
</tr>
<tr>
<td>$s \leftarrow (s + F(p,q,r) + b_5 + C_6) \equiv 12$</td>
<td>$s \leftarrow (s + F(p,q,r) + b_{13} + C_{14}) \equiv 12$</td>
</tr>
<tr>
<td>$r \leftarrow (r + F(s,p,q) + b_6 + C_7) \equiv 17$</td>
<td>$r \leftarrow (r + F(s,p,q) + b_{14} + C_{15}) \equiv 17$</td>
</tr>
<tr>
<td>$q \leftarrow (q + F(r,s,p) + b_7 + C_8) \equiv 22$</td>
<td>$q \leftarrow (q + F(r,s,p) + b_{15} + C_{16}) \equiv 22$</td>
</tr>
</tbody>
</table>
Secure Channels

- **Two major issues**
  - How to make the communication between clients and servers secure?
    - Require authentication
    - Ensuring message integrity and confidentiality
    - Protect the communication within a group of servers
  - Authorization
    - Control access to resources

- **Secure channel**
  - Protects senders and receivers against interception, modification, and fabrication of messages
Authentication (1)

• **Authentication and message integrity**
  – Alice knows for sure that she is talking to Bob once the channel has been set up.
  – Data messages are exchanged after authentication has taken place.
  – Both of them should go together!

• **Session Key**
  – Used to encrypt messages for integrity
  – Used only for as long as the channel exist
Authentication (2)

- **Authentication based on a shared secret key**
  - Assume that key \((K_{A,B})\) is already shared between A and B
  - Challenge-response protocol
    - One party challenges the other to a response that can be correct only if the other knows the shared secret key

<Authentication based on a shared secret key.>
Authentication (3)

• Optimization of authentication protocol
  – Number of messages has been reduced from 5 to 3
  – Idea: A eventually want to challenge B, A send a challenge along with her identity
  – Reflection attack is possible!

<Optimized authentication based on a shared secret key>
Authentication (4)

- **Reflection attack**
  - Chuck’s goal is to set up a channel with Bob so that Bob believes he is talking to Alice

Chuck need to prove he knows the secret key by returning $K_{A,B}(R_B)$

But Chuck does not have $K_{A,B}$

3. Chuck attempt to set up a Second channel with $R_B$

Now Chuck got $K_{A,B}(R_B)$

OK! Let’s finish setting up the first session!!
Authentication (5)

- Authentication using a Key Distribution Center (1)
  - To solve the scalability problem of shared secret key
  - Only KDC shares a secret key with each of the host

1. If Alice wants to set up a secure channel with Bob, she requests help to KDC.

2. KDC returns an encrypted message containing a secret key $K_{A,B}$ and sends it also to Bob.

If Alice wants to start setting up a secure channel with Bob before Bob has received the shared key from KDC, what happens?
Authentication (6)

- Authentication using a Key Distribution Center (2)
  - Using a ticket and letting Alice set up a connection to Bob.
  - Bob is the only one besides the DKC who knows $K_{B,KDC}$

If Chuck had stolen one of Bob’s old key, he can do Replay Attack!!
Authentication (7)

- Needham-Schroeder authentication protocol
  - Multiway challenge-response protocol

Alice sends a request to KDC with a **challenge (or nonce) $R_{A1}$**

1. $R_{A1}, A, B$


3. $K_{A,B}(R_{A2}), K_{B,KDC}(A, K_{A,B})$

4. $K_{A,B}(R_{A2}^{-1}, R_{B})$

5. $K_{A,B}(R_{B}^{-1})$

Alice encrypts new challenge $RA2$ with secret shared key $KA,B$

Bob can decrypt ticket with $KB,KDC$

Bob becomes to know $KA,B$

Bob can decrypt $RA2$

If Chuck ever got a hold of an old key $KA,B$, he could replay message 3.
Authentication (8)

- **Needham-Schroeder authentication protocol**
  - Protection against malicious reuse of a previously generated session key in the Needham-Schroeder protocol.

1. Alice first request Bob’s challenge.
2. Bob send encrypting challenge to A
3. KDC decrypt Bob’s challenge with $K_{B,KDC}$.
4. KDC encrypt $K_{A,B}$ containing Bob’s challenge in order Bob to know that the session key is tied to the original request from Alice to talk to Bob.
Authentication (9)

- **Authentication Using Public-Key Cryptography**
  - Does not require KDC
  - It is important that Alice is guaranteed to be using Bob’s public key, and not the public key of someone else.

<Mutual authentication in a public-key cryptosystem.>

Only Bob can decrypt this message
Only Alice can decrypt this message
Message Integrity and Confidentiality (1)

- **Issues to guarantee integrity**
  - No maliciously changing of message
  - Sender cannot deny ever having sent the message
- **Digital Signatures (1)**
  - Sender digitally signs the messages
  - Prevent modification of messages

<Digital signing a message using public-key cryptography.>
Message Integrity and Confidentiality (2)

• Digital Signatures (2)
  – Problems of using public-key cryptography
    • Private key should remain a secret
    • When keys are changed, key should be notified to the opposite
    • Encrypting entire messages is costly and actually unnecessary
  – Digitally signing only message digest

<Digitally signing a message using a message digest.>
Message Integrity and Confidentiality (3)

- **Session Keys**
  - To establish a secure channel
  - Used for confidentiality
  - Discard when the channel is no longer used
    - The key which is used often becomes easier to reveal.
    - Much safer to use the session keys as little as possible
    - Ensure protection against replay attacks
Secure Group Communication (1)

- **Group communication**
  - Secure communication between more than just two parties

- **Confidential Group Communication**
  - All group members share the same secret key
  - Separate shared secret key betw each pair of group members
  - Use public-key cryptosystem
    - Each pair has its own (public key, private key) pair
Secure Group Communication (2)

• Secure Replicated Servers (1)
  – Client issues a request to a group of replicated server
  – Client expects Byzantine failure tolerant response

• Secret sharing
  – Multiple user share a secret, non of them knows the entire secret
  – Secret can be revealed only if they all get together

• (n, t)-secret sharing
  – Secret can be revealed if t number of users get together
Secure Group Communication (3)

- Secure Replicated Servers (2)
  - When servers are actively replicated, each server accompanies its response with a digital signature

  \[ r_i : \text{response from server } S_i \]
  \[ \text{md}(r_i) : \text{message digest} \]

  - (m,n)-threshold scheme
  - m = c+1, n = N
  - c : possible number of corrupted server

<Sharing a secret signature in a group of replicated servers.>
Access Control (1)

• Access right, Access control and Authorization
  – when request from a client invoke a method of a specific object, access right for that invocation is needed
  – Verifying access rights is referred to as access control
  – Authorization is about granting access rights
Access Control (2)

- **General Issues in Access Control**
  - Subject issue a request to access an object
  - Reference monitor
    - Protection object against invocation
    - Record which subject may do what
    - Decide whether a subject is allowed to have a specific operation carried out

General model of controlling access to objects.
Access Control (3)

- Access Control Matrix (1)
  - Modeling the access rights of subjects
  - Matrix $M[s,o]$
    - subject $s$ can request to be carried out on object $o$
  - Many entries in the matrix will be empty
- Access Control List (ACL)
  - Each object maintain a list of the access rights of subjects
  - Matrix is distributed column-wise across all objects
- Capabilities
  - Holds a given certain rights for objects
Access Control (3)

- **Access Control Matrix (2)**

  Comparison between ACLs and capabilities

<table>
<thead>
<tr>
<th>Client</th>
<th>Server</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create access request $r$ as subject $s$</td>
<td>( (s, r) ) ( r ) appears in ACL(s) grant access;</td>
</tr>
</tbody>
</table>

**Using ACL**

<table>
<thead>
<tr>
<th>Client</th>
<th>Server</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create access request $r$ for object $o$. Pass capability $C$</td>
<td>( (o, r) ) ( r ) appears in $C$ grant access;</td>
</tr>
</tbody>
</table>

**Using capabilities**

When a client sends a request to a server, the server’s reference monitor will check whether that client is allowed to have the requested operation carried out.

A client simply sends its request to the server. The server is not interested in whether it knows the client. The capability says enough.
Access Control (4)

- Protection Domains (1)
  - One general way of reducing ACLs
  - Set of (object, access rights) pairs
  - Construct groups of users
  - Hierarchical grouping is more flexible

<The hierarchical organization of protection domains as groups of users.>
Access Control (5)

• **Protection Domains (2)**
  – Hierarchical group is easy to manage group membership
  – Large group can be constructed efficiently
  – Looking up member can be costly if the membership DB is distributed

• **Alternatives**
  – Each subject carry a *certificate* listing the groups it belongs to
  – Implement protection domains as *roles*
    • User always logs into the system with a specific role
Firewalls (1)

• **Packet-filtering gateway**
  – Make decisions as to whether or not to pass a network packet based on the source and destination address
  • (ex) drop all incoming packets addressed to specific Web server
  – Inspects only the header of network packet

• **Application-level gateway**
  – Inspects the content of an incoming or outgoing message
  • (ex) mail gateway filtering spam e-mail
  – **Proxy gateway**
  • Works as a front end to a specific kind of application
  • Ensure that only those messages are passed that meet certain criteria
Firewalls (2)

- Common implementation of a firewall
Secure Mobile Code (1)

- Mobile Agents
  - Protecting an Agent
    - Should be protected against malicious hosts
    - Against stealing or modifying information
  - Protecting the Target
    - Host need to be protected against malicious agents
    - The program should not be allowed unauthorized access to the host’s resource
Secure Mobile Code (2)

• Protecting an Agent
  – Read-only state
    • Data item is signed by the agent’s owner
    • Owner make message digest
    • When agent arrived at a host, that host can easily verify the read-only messages
  – Append-only logs
    • Data can only be appended to the log (no removal or modification)
    • Use checksum and private and public key
  – Selective revealing
    • Provide array of data item, where each entry is intended for a designated server
    • Entry is encrypted by server’s public key for confidentiality
Secure Mobile Code (3)

- Protecting an Target (1)
  - Sandbox in JVM
    - Class loader is responsible for fetching a specified class from a server and install it in the client’s address space
    - Security manager performs various checks at runtime
    - Byte code verifier

Byte code verifier checks whether a downloaded Class obeys the security rules of the sandbox
Secure Mobile Code (4)

- **Protecting an Target (2)**
  - Sandbox
    - downloaded program is executed in such a way that each of its instructions can be fully controlled.
  - Playground
    - Separate, designated machine exclusively reserved for running mobile code
Secure Mobile Code (5)

- Protecting an Target (3)
  - Using Java object reference as capabilities
    - To access local resource, program must have a reference to a specific object that handles file operation
    - All interfaces to objects are initially hidden from the program
    - After type checking, it is impossible to construct a reference to one of these interfaces at runtime
Secure Mobile Code (6)

• Protecting an Target (4)
  – (extended) stack introspection
    • Before calling to any method, `enable_privilege` procedure is preceded
    • If the invocation is authorized, gives temporal privilege to the caller
    • After finishing the invocation, `disable_privilege` is invoked to disable the privilege
Secure Mobile Code (7)

- Protecting an Target (5)
  - Name space management
    - To give programs access to local resources
      - Need to attain access by including the appropriate files that contain the classes implementing those resources
      - Inclusion requires that a name is given to the interpreter
      - Resolve it to a class
    - Name resolution is handled by class loader
Security Management

• How keys are obtained?
• Secure Group Management
• Authorization Management
Key Management (1)

- **Key Establishment**
  - How session keys can be established?
  - Diffie-Hellman key exchange

Alice and Bob, and only those two, will have the shared secret key \( g^{xy} \mod n \).

Neither of them needed to make their private number (x and y, respectively) known to the other.
Key Management (2)

• **Key Distribution (1)**
  - In a symmetric cryptosystem,
    - the initial shared secret key must be communicated along a secure channel that provides authentication and confidentiality

<Secret Key Distribution>
Key Management (3)

Key Distribution (2)

- In a public-key cryptosystem,
  - The receivers can be sure that the key is indeed paired to a claimed private key

<Public-Key Distribution (See also Menezes et al., 1996)>
Key Management (4)

- **Key Distribution (3)**
  - Authenticated distribution of public key
    - Public key and identifier have together been signed by a CA
  - Certification Authority (CA)
    - Sign with private key of CA
    - Public key of CA are well known
    - If client want to ascertain that the public key which he knows indeed belongs to the identified entity
      - Use public key of CA
Key Management (5)

• **Lifetime of Certificates**
  – Certificate Revocation List (CRL)
    • Published regularly by CA
    • Whenever client checks a certificate, it must check the CRL
    • Client should contact the CA each time a new CRL is published
  – Restrict the lifetime of certificate
    • Validity of a certificate automatically expires after some time
    • Client should check the latest CRL whenever verifying a certificate
  – Reduce lifetime of a certificate to nearly zero
    • Client always have to contact the CA to check the validity of a public key
Secure Group Management

- **Goal**

  When a process $P$ asks to join a group $G$, $Q$ should send secret key of group safely.

  1. Joint request with identifying $G$ and $P$, $P$’s local time $T$, reply pad $RP$ generated secret key $K_{P,G}$.

  $T$ is used to make sure that the certificate was still valid at the time it was sent.

$P$ knows $RP$, so $P$ can know $CK_G$, then eventually $P$ can know private key of group

$<$Securely admitting a new group member.$>$
Authorization Management (2)

- **Capabilities and Attribute Certificates (1)**
  - Amoeba operating system
    - Object resides at a server
    - Clients are offered transparent access to that object
    - Client passes a capability to its local OS to invoke an operation on an object
  - Capability in Amoeba

<table>
<thead>
<tr>
<th>48 bits</th>
<th>24 bits</th>
<th>8 bits</th>
<th>48 bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>Server port</td>
<td>Object</td>
<td>Rights</td>
<td>Check</td>
</tr>
</tbody>
</table>

Object identifier

Access rights of the holder

random

When an object is created, its server pick a random check filed and stores it both in the capability
Authorization Management (2)

• Capabilities and Attribute Certificates (2)
  - Create a restricted capability

  ![Diagram showing the process of creating a restricted capability from an owner capability]

  - Modification of capability check
  • With original random number, right field and one-way hash function

  <Generation of a restricted capability from an owner capability.>
Authorization Management (3)

- Delegation (1)
  - Passing certain access rights from one process to another
  - Easier to distribute work
  - (ex) Proxy
    - Token that allows its owner to operate with the same or restricted rights
  - Two simple approach
    - Delegation is relatively simple if delegator knows everyone
    - Simply construct a secure certificate

<The general structure of a proxy as used for delegation.>
**Authorization Management (4)**

- **Delegation (2)**
  - Another way of looking at the proxy

  ![Diagram](image)

  - When delegating some of rights, Alice gives the signed list of rights
  - Bob sends the signed rights to server, server will ask him nasty question
  - If Bob knows the answer to it, server will know for sure that Alice had indeed delegated the listed rights to Bob.
Authorization Management (5)

- Delegation (3)
  - Properties
    - Alice need not be consulted
    - Bob can decide to pass on the rights to someone else.
  - Basic ideas is “show you know a secret”
Example: Kerberos (1)

- **Kerberos (1)**
  - Based on the Needham-Schroeder authentication protocol
  - assists clients in setting up a secure channel with a server

- **Authentication Server (AS)**
  - Handle a login request from a user
  - Authenticates a user
  - Provides a key that can be used to set up secure channel with server

- **Ticket Granting Service (TGS)**
  - Hands out tickets that are used to convince a server that the client is really who he claims to be
Example: Kerberos (2)

- **Kerberos (2)**
  - Login procedure
    - For Alice to log onto the system, AS returns a session key and ticket that will be needed to hand over to the TGS
    - After login, Alice can contact other user or servers.
Example: Kerberos (3)

- Kerberos (3)
  - Setting up a secure channel
Example: SESAME (1)

- **Sesame**
  - Secure European System for Application in a Multi-vendor Environment
  - Use public-key cryptography combined with shared secret keys
- **Sesame Components (1)**
  - General security components
  - Client-side security components
  - Server-side security components
Example: SESAME (2)
• Sesame Components (2)

<Overview of components in SESAME.>
Example: SESAME (3)

- **Sesame Components (3)**
  - General Security Components
    - Handle authentication, authorization, and key distribution
    - Authentication Server (AS)
      - Responsible for authenticating users and applications
    - Security Management Information Base (SMIB)
      - Collection of local files (secret keys)
    - Privilege Attribute Server (PAS)
      - Grant attribute certificates listing the access rights
    - Key Distribution Server (KDS)
      - Generate session keys for setting up secure communication with server
Example: SESAME (4)

- **Sesame Components (4)**
  - **Client-side Component**
    - **User Sponsor**
      - Allows a user to login, logout, and to change roles
      - Attains certain default privileges depending on the role
    - **Authentication and Privilege Access Client (APA Client)**
      - Library containing simple routines that give the user interface and client applications access to the domain security server
    - **Secure Association Context Management (SACM)**
      - Responsible for initializing and maintaining the information that is needed for a client to communicate with a server application
Example: SESAME (5)

- **Sesame Components (5)**
  - Server-side Component
    - SACM
    - Maintain security information for communication with client and other application
  - PAC validation Facility (PVF)
    - Extract and validates all the necessary information from incoming request
    - (ex1) decrypt a message to extract a session key
    - (ex2) extract the access rights from a PAC after verifying the PAC’s signature
Example: SESAME (6)

- Privilege Attribute Certificates (PACs)

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Issuer domain</td>
<td>Name the security domain of the issuer</td>
</tr>
<tr>
<td>Issuer identity</td>
<td>Name the PAS in the issuer's domain</td>
</tr>
<tr>
<td>Serial number</td>
<td>A unique number for this PAC, generated by the PAS</td>
</tr>
<tr>
<td>Creation time</td>
<td>UTC time when this PAC was created</td>
</tr>
<tr>
<td>Validity</td>
<td>Time interval when this PAC is valid</td>
</tr>
<tr>
<td>Time periods</td>
<td>Additional time periods outside which the PAC is invalid</td>
</tr>
<tr>
<td>Algorithm ID</td>
<td>Identifier of the algorithm used to sign this PAC</td>
</tr>
<tr>
<td>Signature value</td>
<td>The signature placed on the PAC</td>
</tr>
<tr>
<td>Privileges</td>
<td>A list of (attribute, value)-pairs describing privileges</td>
</tr>
<tr>
<td>Certificate information</td>
<td>Additional information to be used by the PVF</td>
</tr>
<tr>
<td>Miscellaneous</td>
<td>Currently used for auditing purposes only</td>
</tr>
<tr>
<td>Protection methods</td>
<td>Fields to control how the PAC is used</td>
</tr>
</tbody>
</table>
Example: Electronic Payment Systems (1)

- **Electronic Payment Systems (1)**
  - Cash-based system
    - Withdraw money from their bank
    - Hand over to the merchant
  - Check-based system
    - Customer obtain a signed statement from their bank (check)
    - Banks are involved in a settlement
  - Use credit card
    - Customer instructs the bank to transfer money to the merchant’s account
    - Customer hands the merchant a credit card and signing the credit card slip
    - Actual transfer of money occurs when the merchant shows the slip to bank
Example: Electronic Payment Systems (2)

Payment systems based on direct payment between customer and merchant.

a) Paying in cash.
b) Using a check.
c) Using a credit card.
Example: Electronic Payment Systems (3)

- Electronic Payment Systems (2)
  - Payment by money order
    - transfer money explicitly from the customer’s bank to the merchant’s bank
    - Inform the merchant when money has been deposited into his bank
  - Payment through debit order
    - Customer authorize the merchant to instruct the merchant’s bank to transfer money from the customer’s account
    - (ex) regular payments like membership dues or monthly bills
Example: Electronic Payment Systems (4)

Payment systems based on money transfer between banks.

a) Payment by money order.

b) Payment through debit order.
Security in Electronic Payment Systems (1)

- **General Requirements**
  - Use ATM
    - Personal Identification Number (PIN) is needed
  - use Smart Card
    - Capable of storing digital money
    - Using digital money requires protection its use against fraud
      - (ex) use digital money more than once
    - Integrity mechanism is needed
    - Support for nonrepudiation can be provided by digital signature
Security in Electronic Payment Systems (2)

• Privacy (1)
  – To achieve anonymity
  – Hide the identity of a customer
  – In standard cash system, anonymous payment is simple
  – In electronic payment system?
    • Protecting a system against privacy violations concentrates on making the customer anonymous
  – Conditional anonymity
    • Sometimes, it may be necessary to reveal identity
  – Pseudonym
    • Alias that is used during a transaction or series of transaction
Security in Electronic Payment Systems (3)

- **Privacy (2)**
  - What extent data are hidden in payment system?
  - In a traditional cash payment
    - Bank actually knows nothing

<table>
<thead>
<tr>
<th></th>
<th>Merchant</th>
<th>Customer</th>
<th>Date</th>
<th>Amount</th>
<th>Item</th>
</tr>
</thead>
<tbody>
<tr>
<td>Merchant</td>
<td>Full</td>
<td>Partial</td>
<td>Full</td>
<td>Full</td>
<td>Full</td>
</tr>
<tr>
<td>Customer</td>
<td>Full</td>
<td>Full</td>
<td>Full</td>
<td>Full</td>
<td>Full</td>
</tr>
<tr>
<td>Bank</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>Observer</td>
<td>Full</td>
<td>Partial</td>
<td>Full</td>
<td>Full</td>
<td>Full</td>
</tr>
</tbody>
</table>

<Information hiding in a traditional cash payment.>
Security in Electronic Payment Systems (4)

- Privacy (3)
  - Traditional credit-card transaction
    - Bank will know everything except what has actually been bought
    - Do not provide much privacy

<table>
<thead>
<tr>
<th></th>
<th>Merchant</th>
<th>Customer</th>
<th>Date</th>
<th>Amount</th>
<th>Item</th>
</tr>
</thead>
<tbody>
<tr>
<td>Merchant</td>
<td>Full</td>
<td>Full</td>
<td>Full</td>
<td>Full</td>
<td>Full</td>
</tr>
<tr>
<td>Customer</td>
<td>Full</td>
<td>Full</td>
<td>Full</td>
<td>Full</td>
<td></td>
</tr>
<tr>
<td>Bank</td>
<td>Full</td>
<td>Full</td>
<td>Full</td>
<td>Full</td>
<td>None</td>
</tr>
<tr>
<td>Observer</td>
<td>Full</td>
<td>Partial</td>
<td>Full</td>
<td>Full</td>
<td>Full</td>
</tr>
</tbody>
</table>

<Information hiding in a traditional credit-card system (see also [camp.lj96a])>
Example Protocols (1)

• E-cash (1)
  – Main concern is achieving anonymity
  – Blind signature
  • Prevent the bank from being able to keep track of which digital notes it hands out to whom

<The principle of anonymous electronic cash using blind signatures.>
Example Protocols (3)

- **Secure Electronic Transactions (SET)**
  - Multiple protocol for electronic credit-card payments
  - **Dual signature**
    - (ex) possible situation
      - Alice want to purchase some goods from Bob using a credit card.
      - Alice don’t want Bob to know every detail about the payment information.
      - Alice don’t want bank to learn about what Alice has ordered.
    - Construct a **messages digests** for order and payment information
    - Construct a **third digest** over the concatenation of both digests
Example Protocols (4)

- **The Different steps in SET**

1. $[\text{order | pay\_info}]_{A}$, $K_{A,\text{bank}}(\text{pay\_info}), K_{\text{bank}}^{+}(K_{A,\text{bank}})$

2. $K_{B1,\text{bank}}(\text{auth}), K_{\text{bank}}^{+}(K_{B1,\text{bank}}),$ $K_{A,\text{bank}}(\text{pay\_info}), K_{\text{bank}}^{+}(K_{A,\text{bank}})$

3. $K_{B2,\text{bank}}([\text{auth\_OK}]_{\text{bank}}), K_{B}^{+}(K_{B2,\text{bank}}),$ $K_{B3,\text{bank}}([\text{cap}]_{\text{bank}}), K_{B}^{+}(K_{B3,\text{bank}})$

4. $[\text{pay\_OK}]_{B}$

5. $K_{B4,\text{bank}}([\text{pay\_me}]_{B}), K_{\text{bank}}^{+}(K_{B4,\text{bank}}),$ $K_{B3,\text{bank}}([\text{cap}]_{\text{bank}}), K_{\text{bank}}^{+}(K_{B3,\text{bank}})$

6. $K_{B5,\text{bank}}([\text{cap\_OK}]_{\text{bank}}), K_{B}^{+}(K_{B5,\text{bank}})$

**Dually signed messages**

**Capture messages** which can be used later on Bob to actually get the payment from the bank

**Bob can’t decrypt payment information**