Security in Distributed Systems

Chapter 8

Security Aspects

- Security in distributed systems involves two main aspects:
  - Communication among users or processes
    - » *solution*: **secure channels**
  - Authorization of users or processes
    - » *solution*: **access control**
- *Mechanisms*: **cryptographic keys** and **user removing**.

Types of Security Threats

- Interception

  (unauthorized access)

- Interruption

  (denial of service)

- Modification

  (unauthorized data changing)

- Fabrication

  (unauthorized data insertion)

Security Policy

- A secure system needs a

  security policy

  that defines

  the actions that the system entities are allowed to take and the actions that are prohibited.

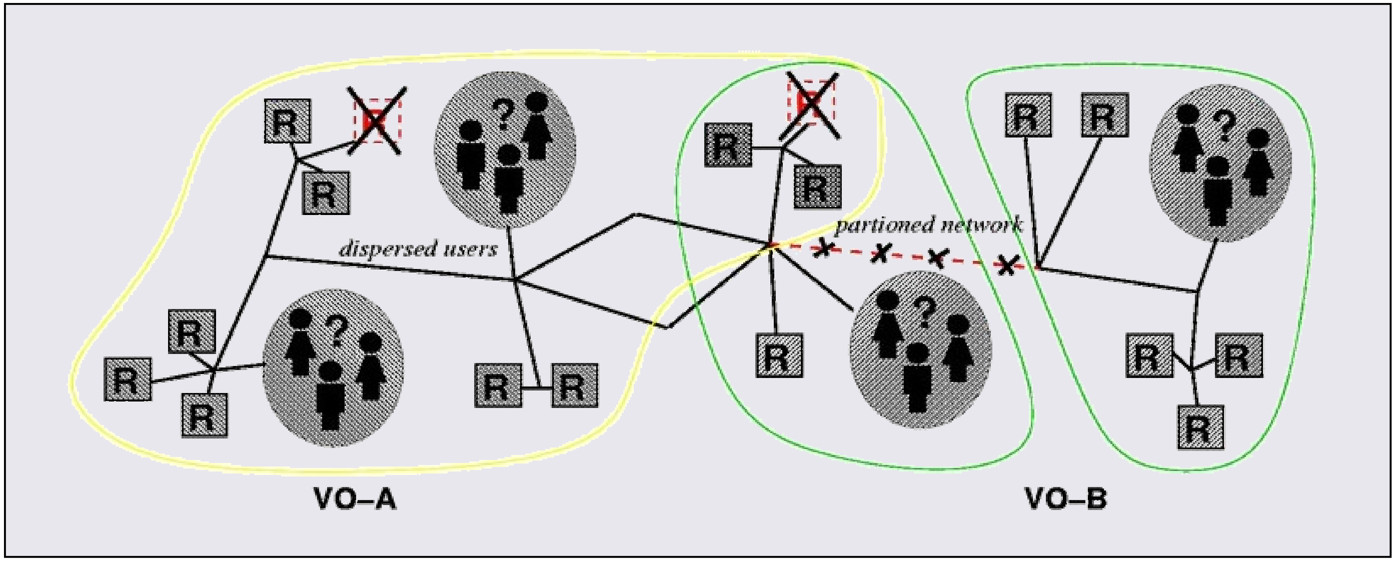
- A policy can be enforced by **security mechanisms**.

Security Mechanisms

- Encryption
- Authentication
- Authorization
- Auditing

Example: The Globus Environment

Globus is a system for configuring and using Grids. Grids are new distributed computing infrastructures. Their main goal is : *Resource sharing & coordinated problem solving in dynamic, multi-institutional virtual organizations*.



Multiple domains in a Grid

Example: Globus Security Policy

Statements composing the security policy of Globus.

1. ***The system consists of several administrative domains.***

2. ***Local operations are subject to a local security policy only.***

3. ***Global operations require the initiator to be known in each domain where the operation is carried out.***

4. ***Operations between entities in different domains require mutual authentication.***

5. ***Global authentication replace local authentication.***

6. ***Controlling access to resources is subject to local security.***

7. ***Users can delegate rights to processes.***

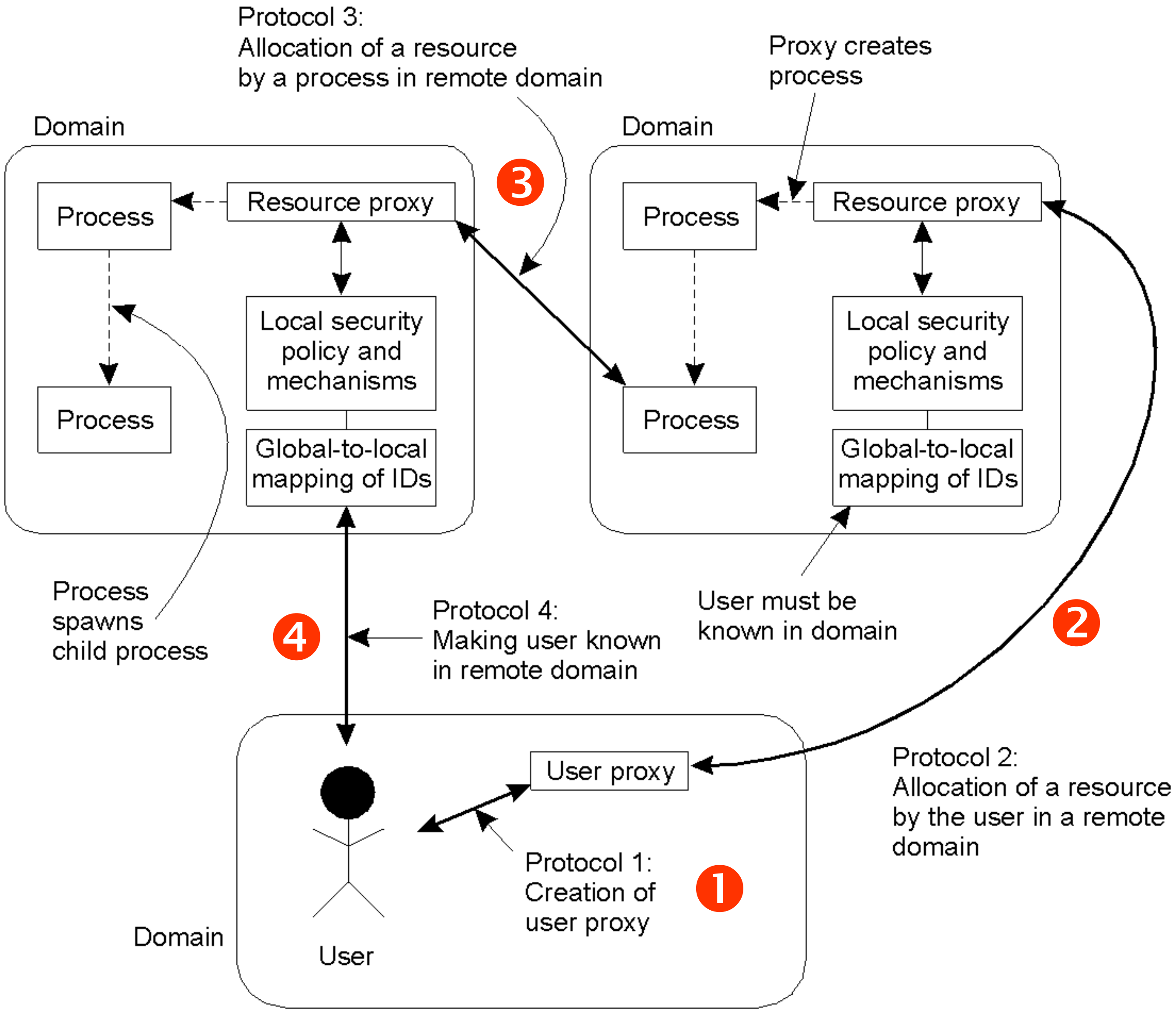
8. ***A group of processes in the same domain can share credentials.***

Example: Globus Security Architecture (1)

- The Globus security architecture consists of entities such as users, processes, user proxies, and resource proxies.
- A **user proxy** is a process that is given permission to act on behalf of a user for a limited period of time.
- A **resource proxy** is a process that in a domain that is used to translate global operations on a resource into local operations that comply with the domain security policy.

Example: Globus Security Architecture (2)

Diagram of Globus security architecture.



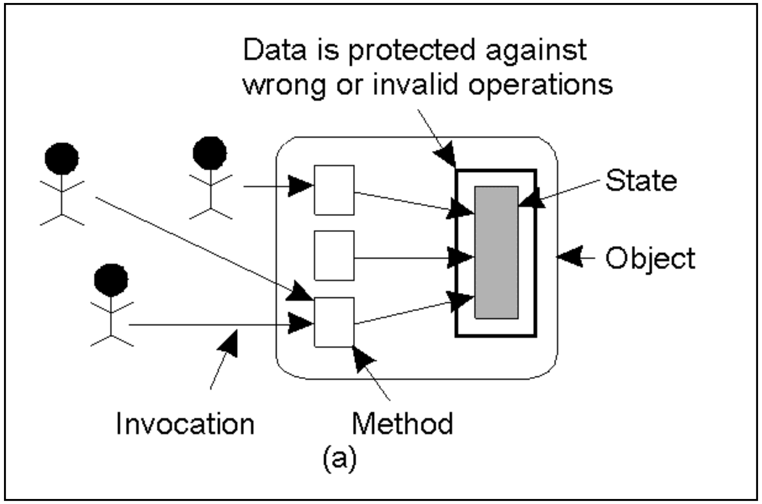
Security Design Issues

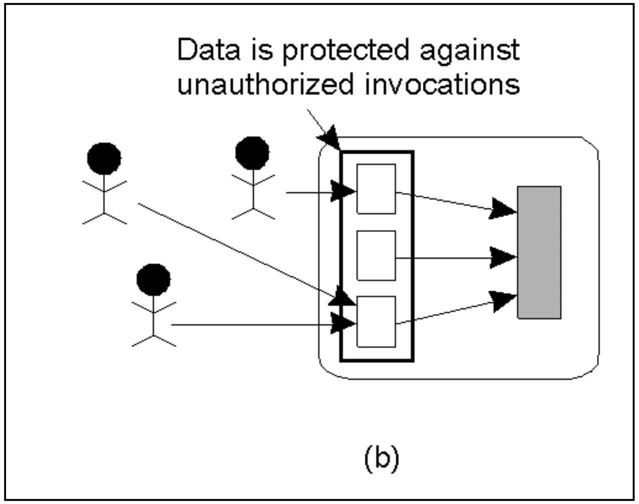
- Security policies can be implementing security services.

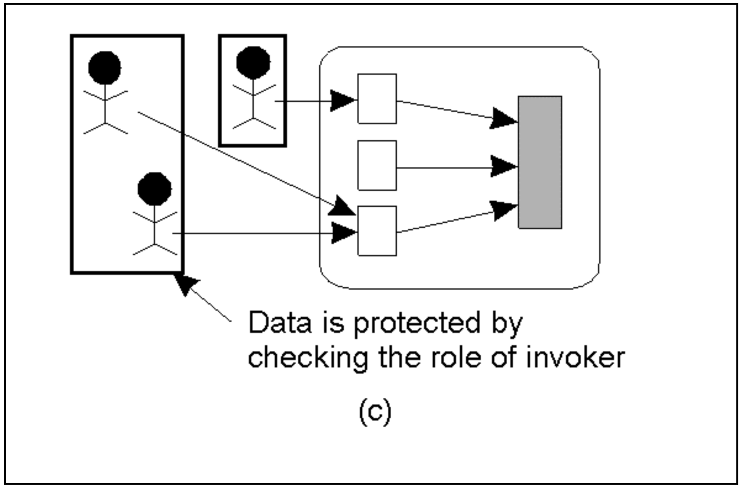
- Issues in designing security services:
  - **focus of control,**
  - **layering mechanisms,**
  - **simplicity.**

Focus of Control

Three approaches for protection against security threats.



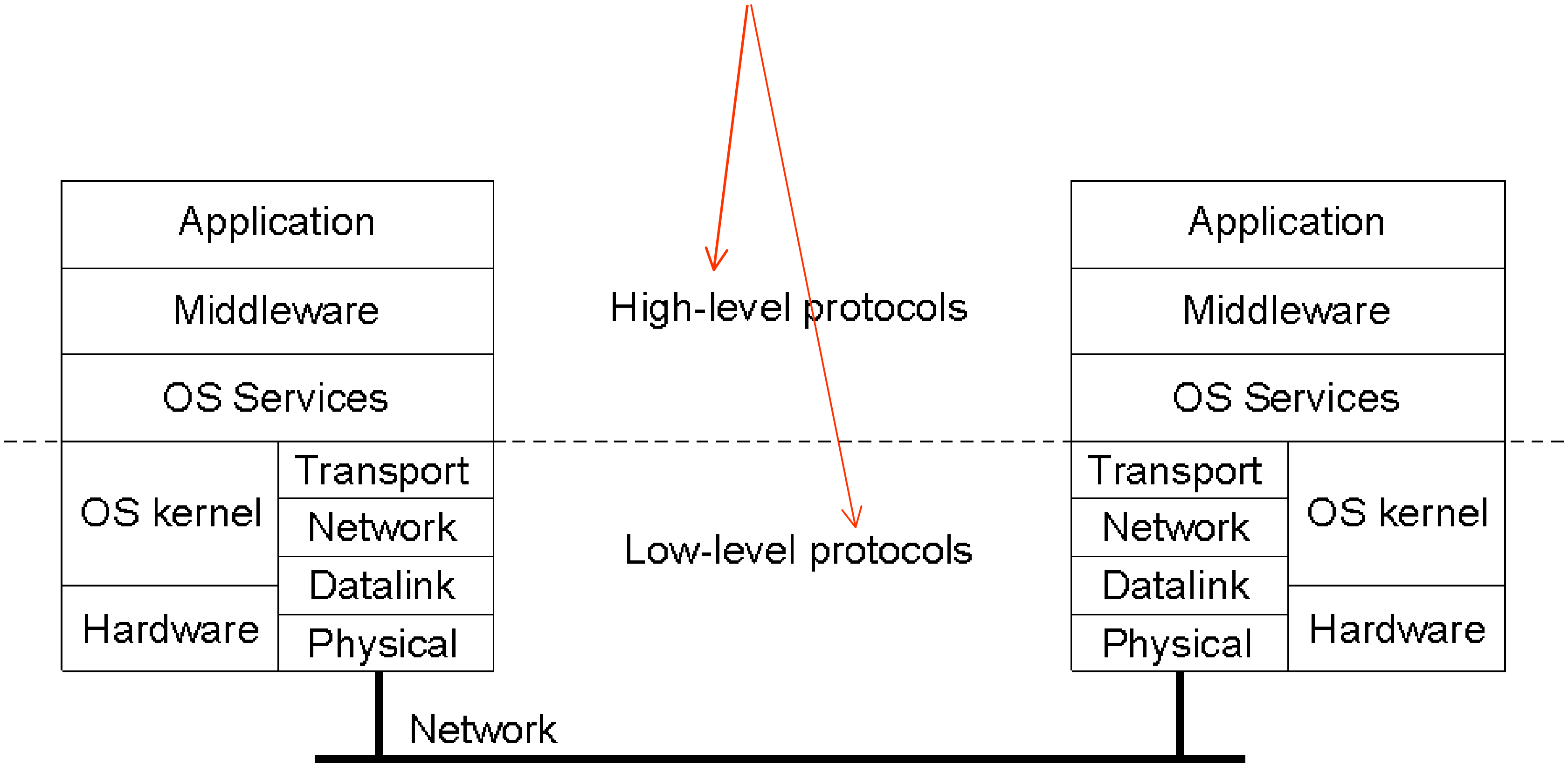




- **(a) Protection against invalid operations**
- **(b) Protection against unauthorized invocations**
- **(c) Protection against unauthorized users**

Layering of Security Mechanisms (1)

At which level security mechanisms should be placed ?



Application

Middleware

OS Services

OS kernel

Transport

Network

Datalink

Physical

Hardware

High-level protocols

Low-level protocols

Application

Middleware

OS Services

Transport

Network

Datalink

Physical

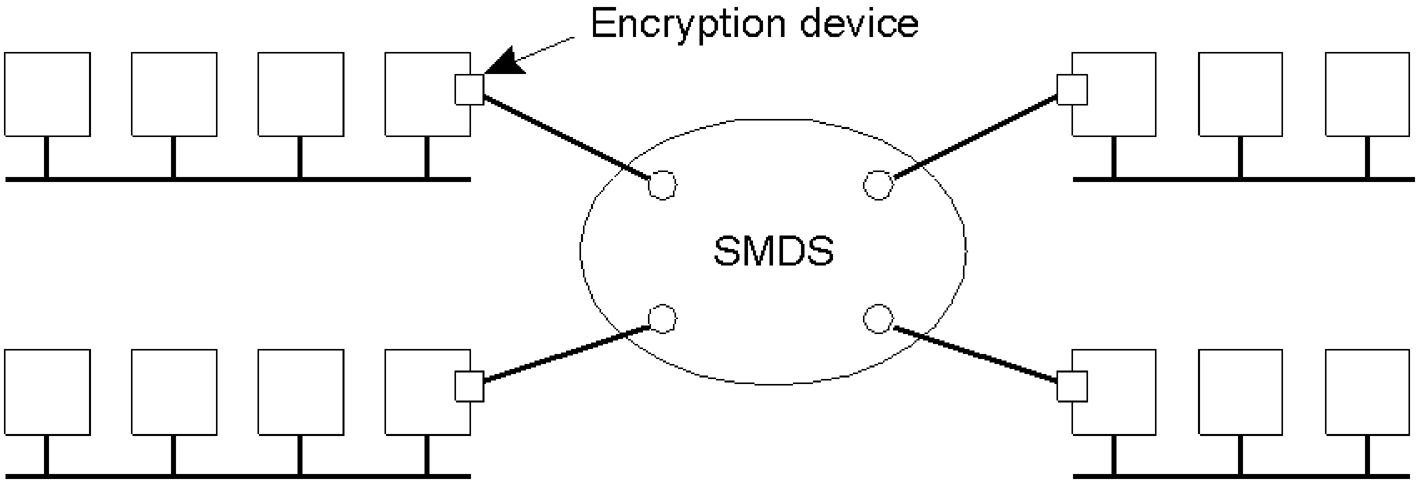
OS kernel

Hardware

Network

The logical organization of a distributed system into several layers.

Layering of Security Mechanisms (2)



Several sites connected through a wide-area backbone service.

Security mechanisms in distributed systems are often placed in the middleware layer.

Distribution of Security Mechanisms (1)

- Dependencies between security services lead to the concept of

  Trusted Computing Base

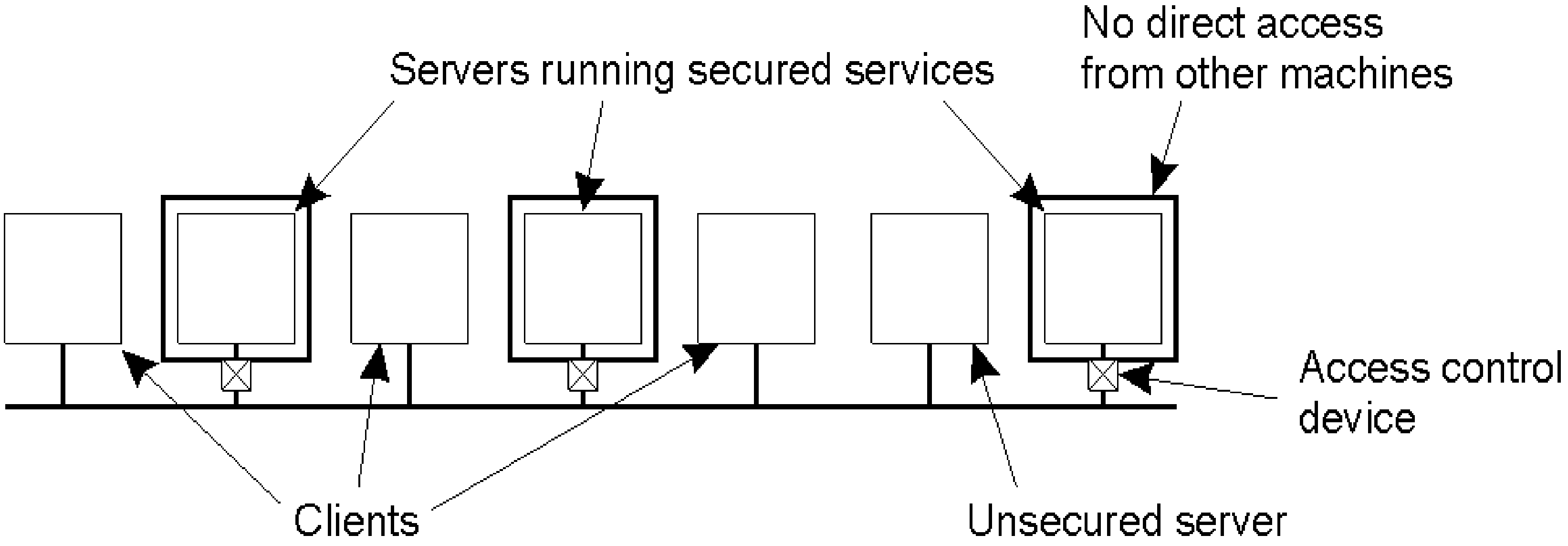
  a set of all security mechanisms in a distributed system that are needed to enforce security.

- TCB in a distributed system may include local operating systems at various hosts.

- **Examples: distributed file system, distributed middleware.**

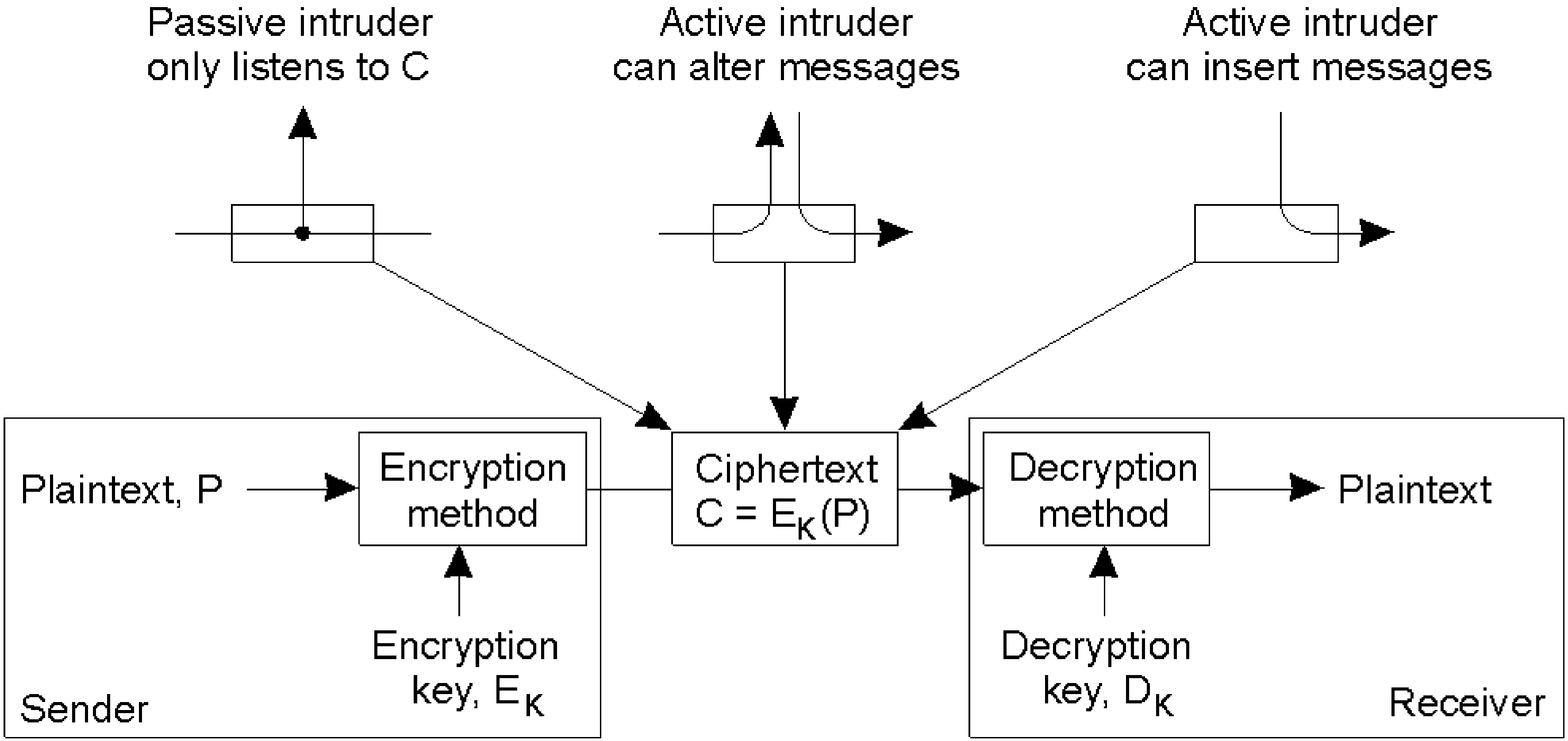
Distribution of Security Mechanisms (2)

- Security services can be isolated from other types of services, reducing the TCB to a small set of machines.
- Reduced Interface for Secure Systems Components


Servers running secured services
No direct access from other machines
Access control device
Clients
Unsecured server

The principle of RISSC as applied to secure distributed systems.

Cryptography (1)



Intruders and eavesdroppers in communication.

Cryptography (2)

- **Symmetric cryptosystem**: $P = D_K(E_K(P))$: same key.

- **Asymmetric cryptosystem**: $P = D_{KD}(E_{KE}(P))$: different keys (one public and one private): public key system.

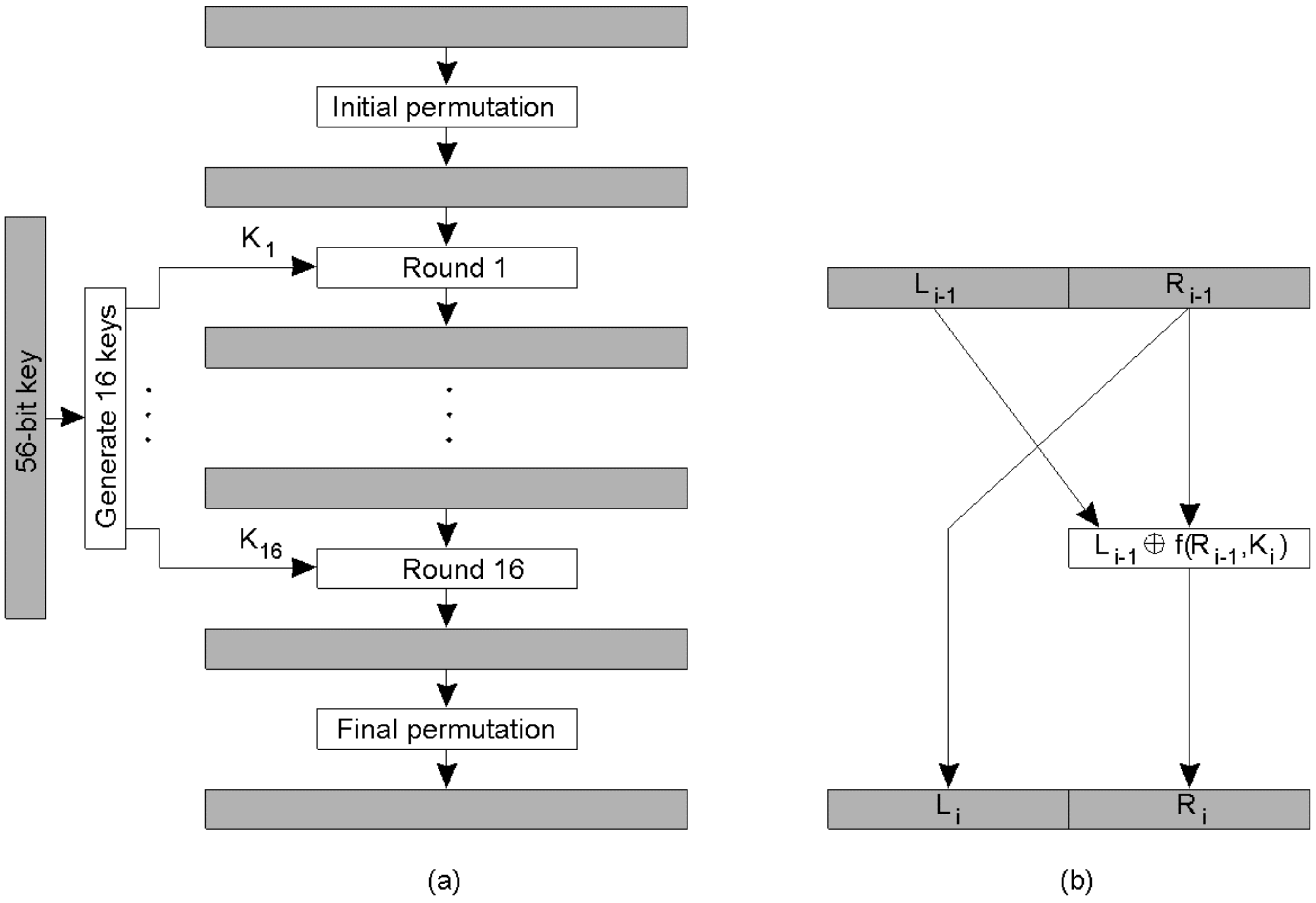
Notation	**Description**
$K_{A, B}$	Secret key shared by A and B
K_A^+	Public key of A
K_A^-	Private key of A

Notation.

Symmetric Cryptosystems: DES (1)

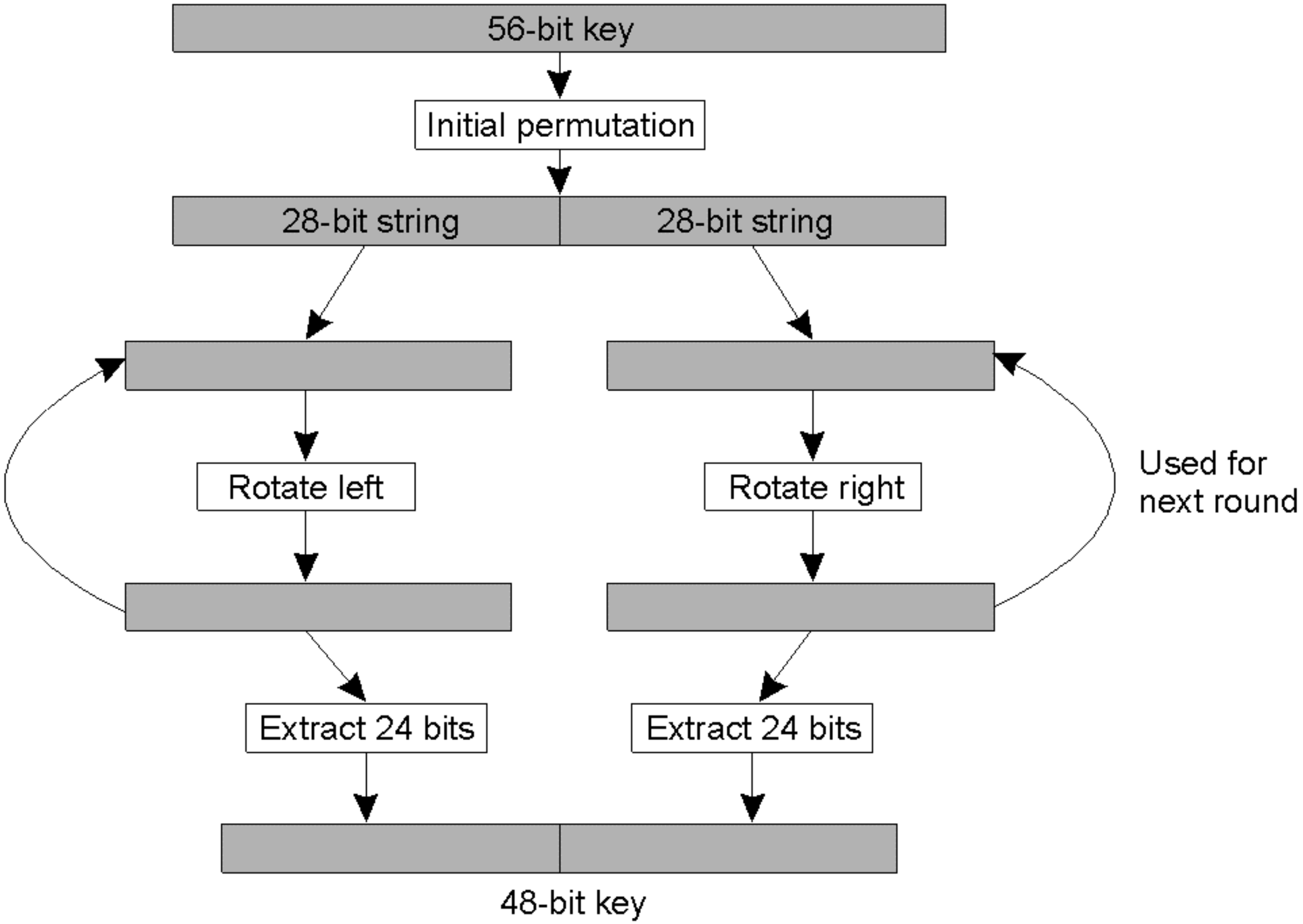
- Data Encryption Standard (DES).
- It operates on 64-bit blocks of data using 16 48-bit keys derived from a 56-bit master key.
- A function mangler ***f*** is used to encrypt a 32-bit block.

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.

Public-Key Cryptosystems: RSA (1)

Asymmetric cryptosystem by Rivest, Shamir and Adleman.

1. Based on the prime numbers of large numbers.

2. Each integer can be written as the product of prime numbers.

3. Public and private keys are constructed from very large prime numbers.

Public-Key Cryptosystems: RSA (2)

Generating the private and public key requires four steps:

- Choose two very large prime numbers, p and q
- Compute $n = p$ x q and $z = (p - 1)$ x $(q - 1)$
- Choose a number d that is relatively prime to z
- Compute the number e such that e x $d = 1 \ mod \ z$

d can be used for decryption and e for encryption.

Security

- Security in distributed systems involves two main aspects:
  - Communication among users or processes
    - *solution*: **secure channels**
  - Authorization of users or processes
    - *solution*: **access control**

Secure Channels (1)

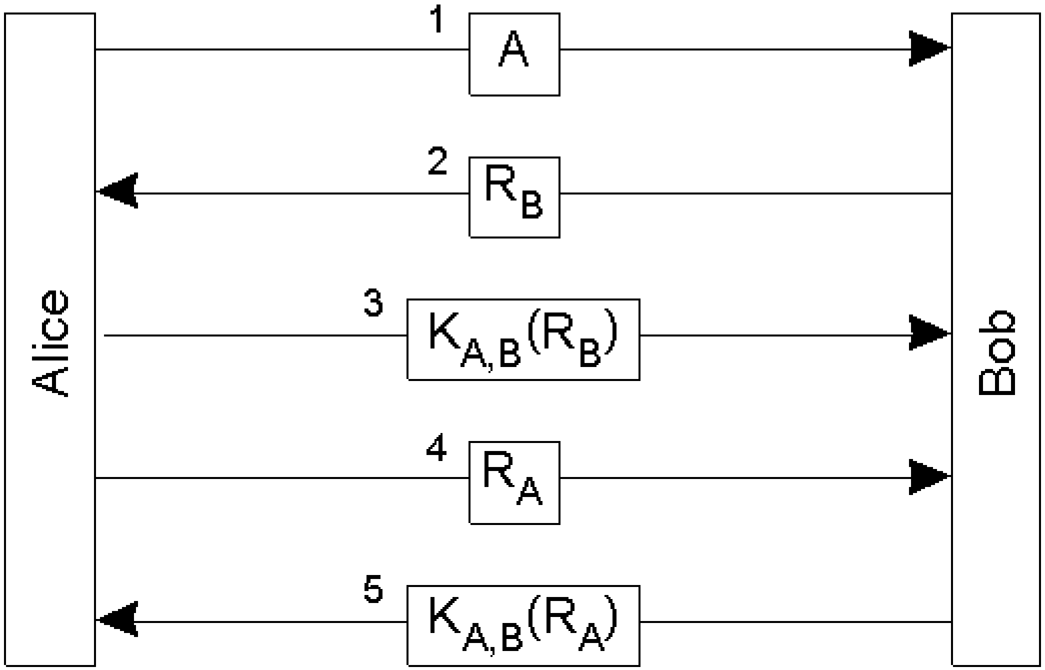
- A Secure Channel protects sender and receiver from
  - Interception *(unauthorized channel access)*,
  - Modification *(unauthorized message changing)*,
  - Fabrication *(unauthorized message insertion)*.

Secure Channels (2)

- Authentication and Message Integrity should be jointly assured.
- A secret key associated to a channel is used : **session key**.
- When a channel is closed the session key is discarded or destroyed.

Authentication (1)

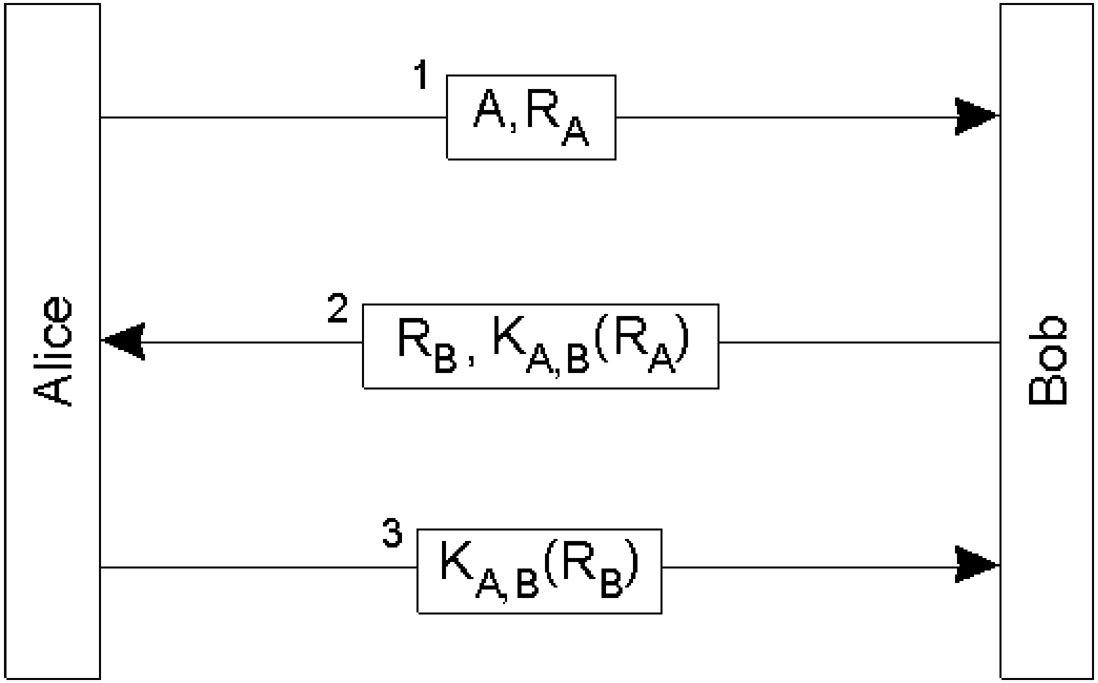
Challenge-Response Protocol



Authentication based on a shared secret key ($K_{A,B}$).

Authentication (2)

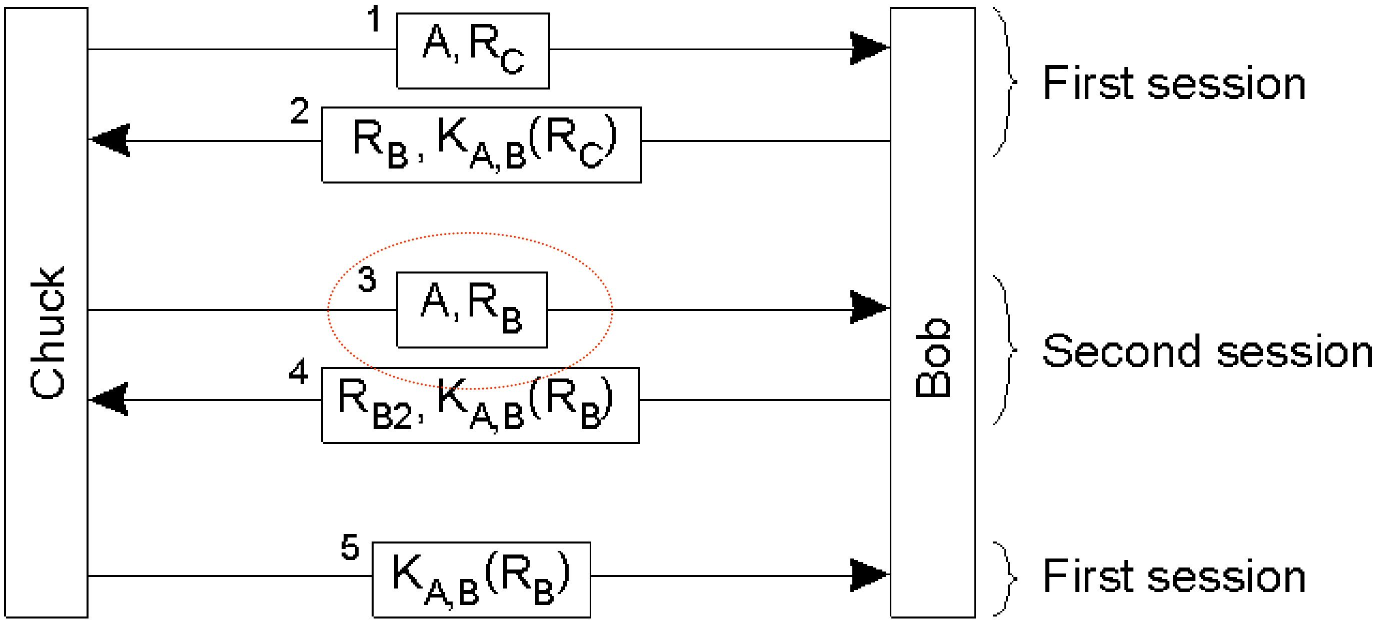
Challenge-Response Protocol optimization ??



Authentication based on a shared secret key, but using three instead of five messages.

Authentication (3)

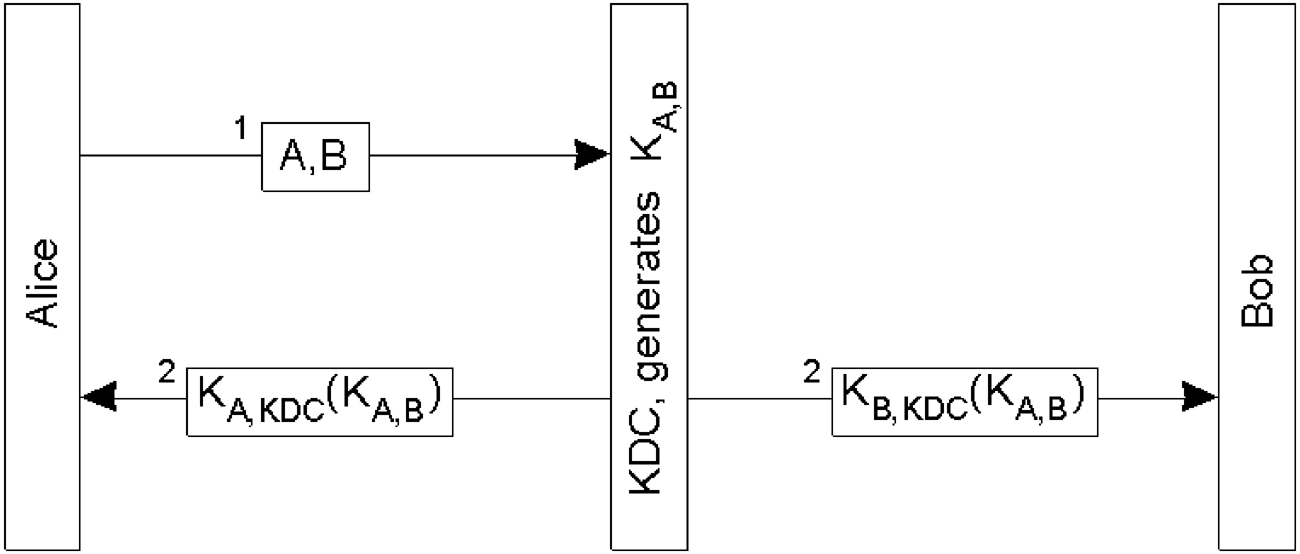
The Reflection Attack defeats the 3-messages protocol.



Different challenges must be used to solve this problem.

Authentication Using a **Key Distribution Center** (1)

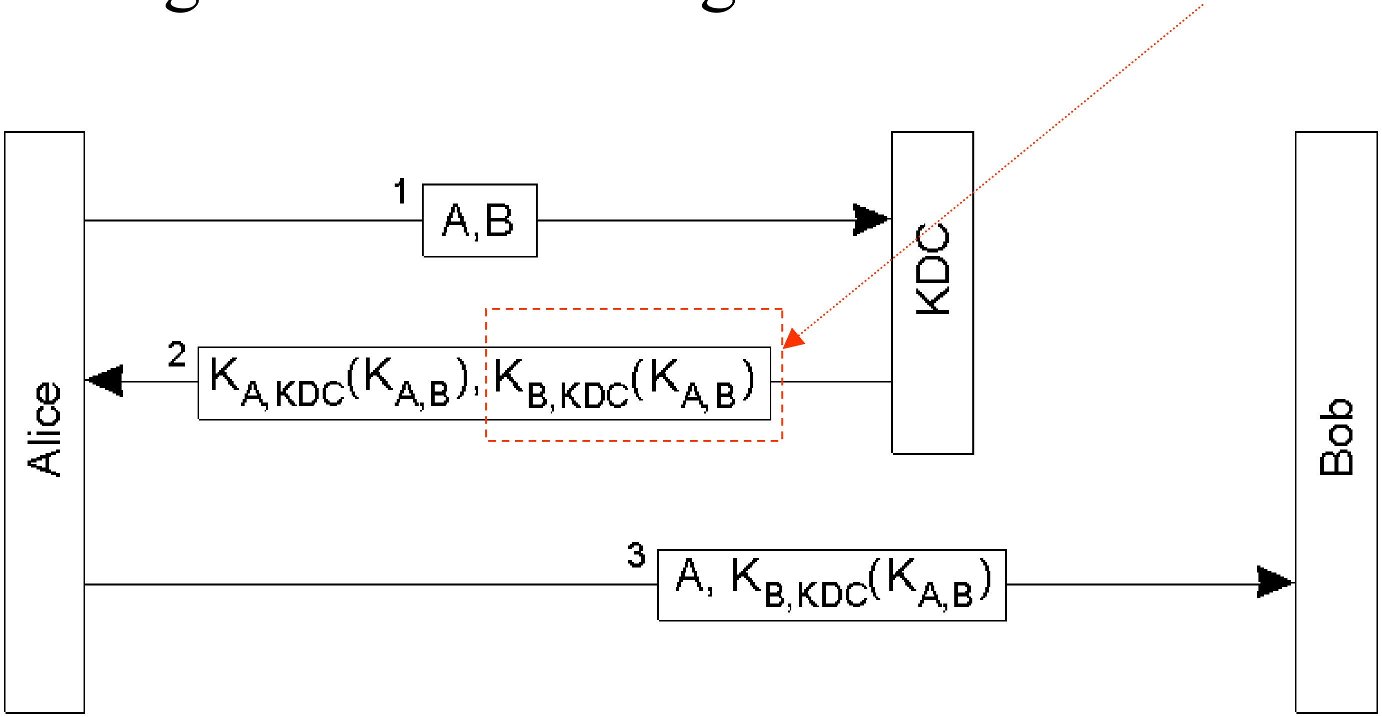
- In a distributed system composed of **N** hosts, each host shares **N-1** keys and globally **N(N-1)/2** secret keys are necessary.
- In this case a centralized approach can be used **--> KDC** handling **N** keys.



The principle of using a KDC.

Authentication Using a **Key Distribution Center** (2)

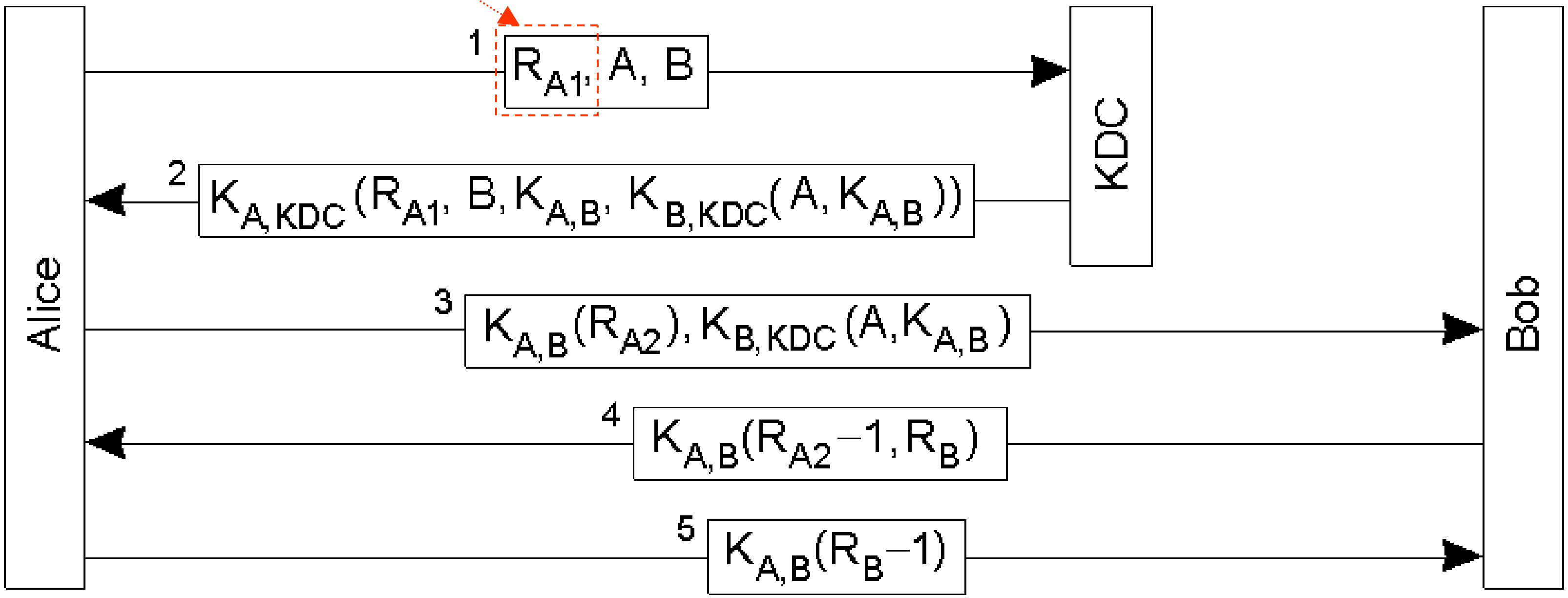
- KDC can pass ***$K_{B,KDC}(K_{A,B})$*** to ***A*** and lets her take care of connecting to ***B***. The message is named a **ticket**.



Using a ticket and letting Alice set up a connection to Bob.

Authentication Using a **Key Distribution Center** (3)

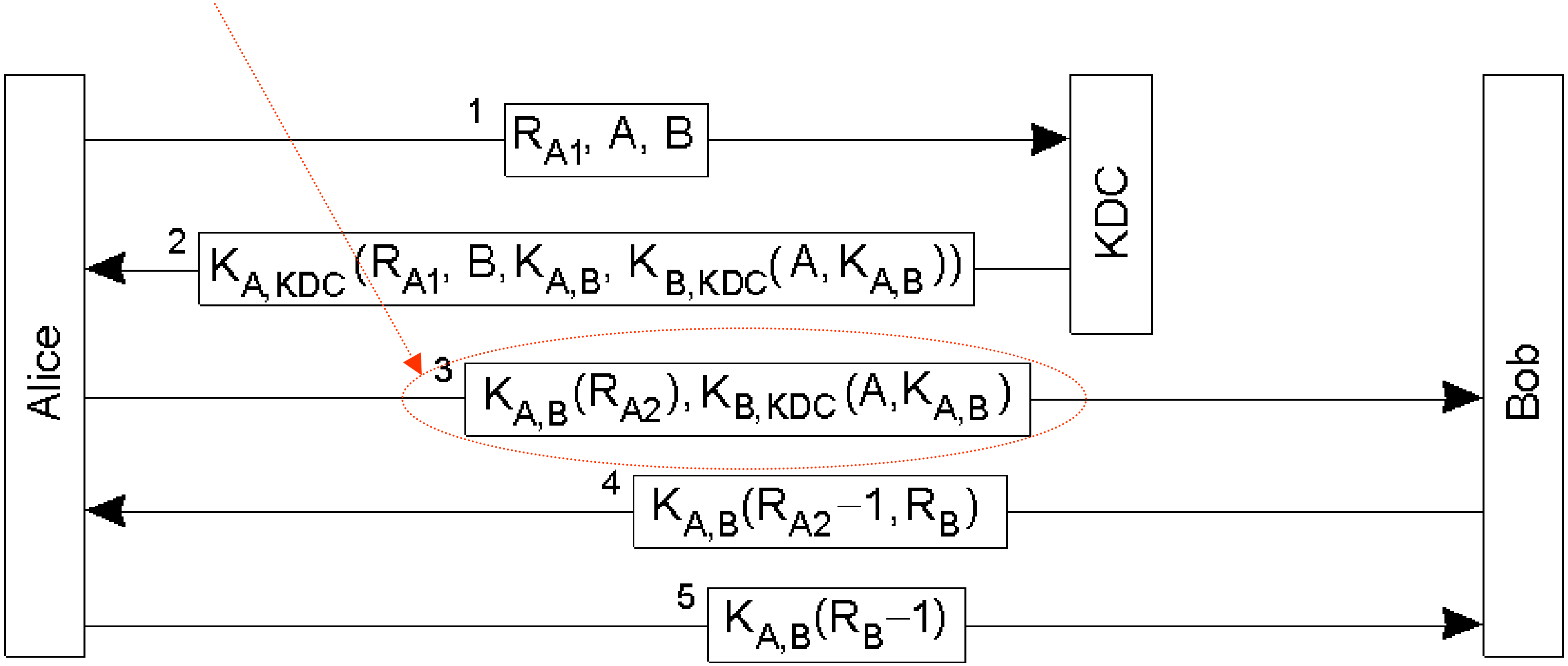
Nonce: random number used only once.


Alice
KDC
Bob
1 R_{A1}, A, B
2 $K_{A,KDC}(R_{A1}, B, K_{A,B}, K_{B,KDC}(A, K_{A,B}))$
3 $K_{A,B}(R_{A2}), K_{B,KDC}(A, K_{A,B})$
4 $K_{A,B}(R_{A2}-1, R_B)$
5 $K_{A,B}(R_B-1)$

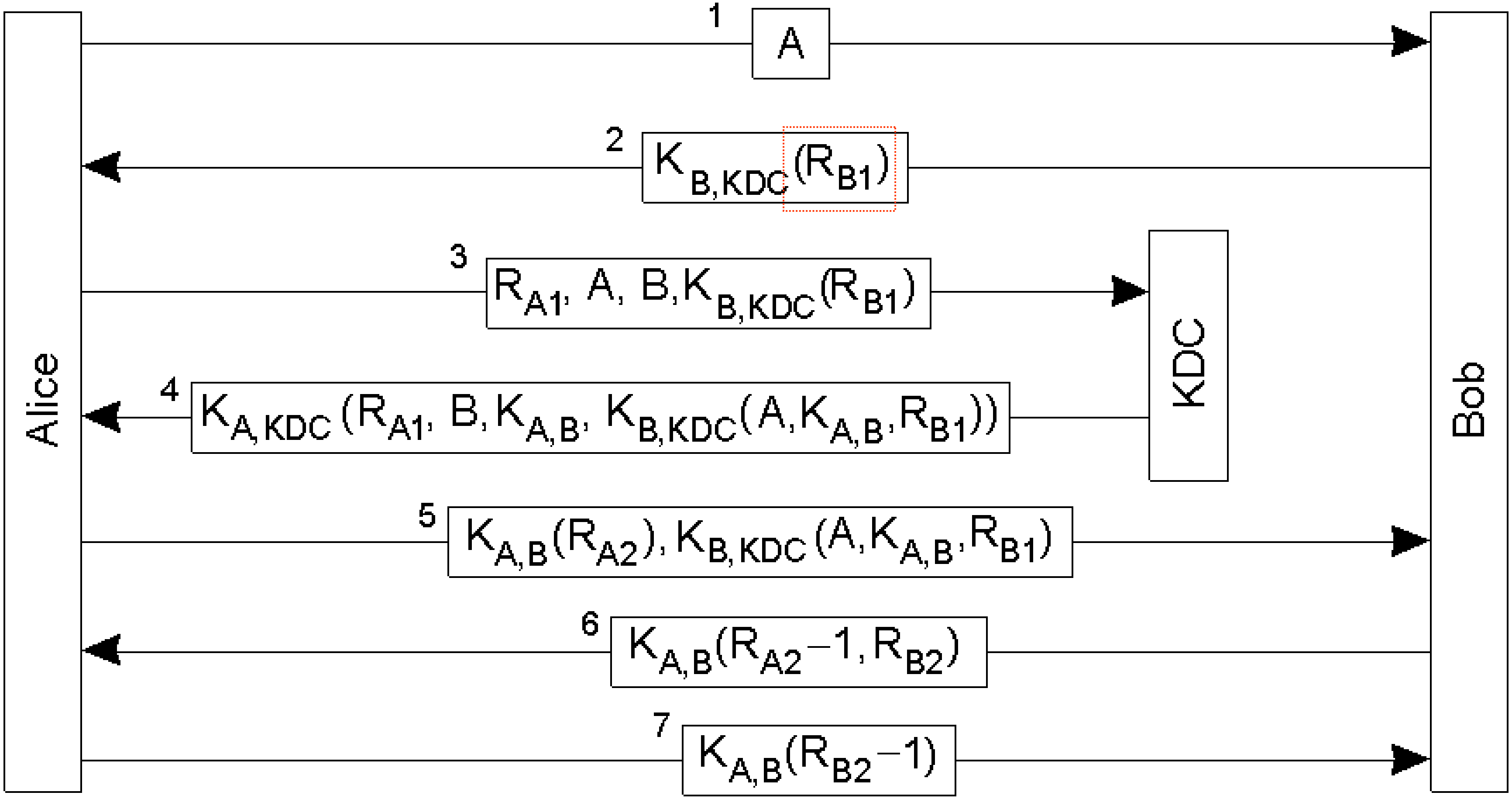
The Needham-Schroeder authentication protocol.

Authentication Using a **Key Distribution Center** (4)

- Message 3 should be related to message 1 to avoid
- instrusion by another entity.
- A nonce can be used in the request by Alice.

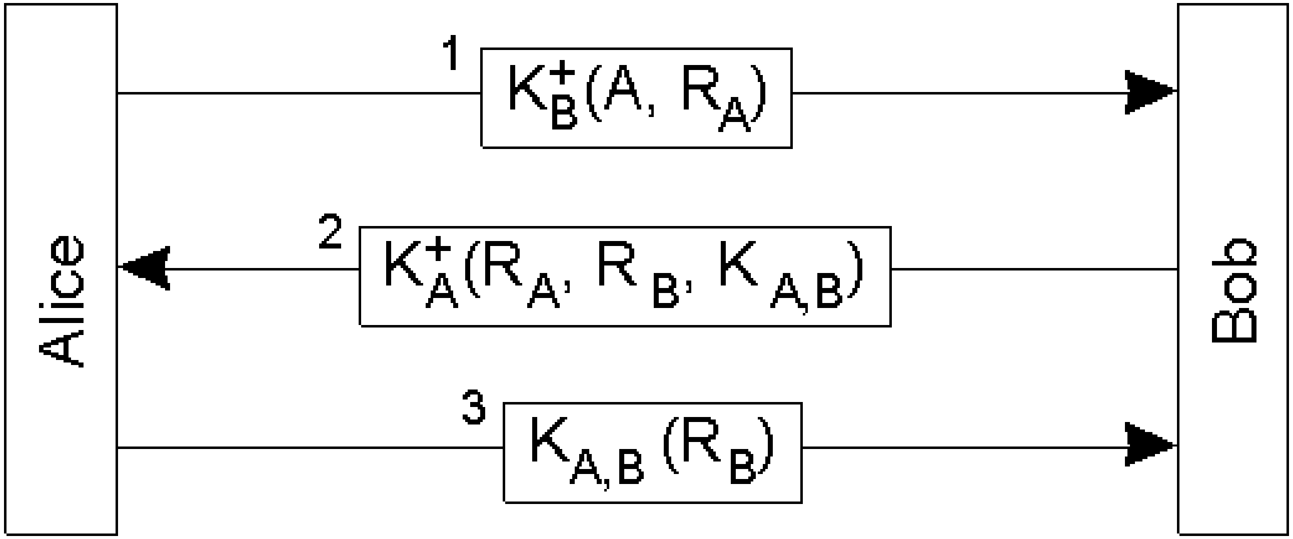

Alice
KDC
Bob
1
R_{A1}, A, B
2
$K_{A,KDC}(R_{A1}, B, K_{A,B}, K_{B,KDC}(A, K_{A,B}))$
3
$K_{A,B}(R_{A2}), K_{B,KDC}(A, K_{A,B})$
4
$K_{A,B}(R_{A2}-1, R_B)$
5
$K_{A,B}(R_B-1)$

Authentication Using a Key Distribution Center (5)


Alice
KDC
Bob
1
A
2
$K_{B,KDC}(R_{B1})$
3
$R_{A1}, A, B, K_{B,KDC}(R_{B1})$
4
$K_{A,KDC}(R_{A1}, B, K_{A,B}, K_{B,KDC}(A, K_{A,B}, R_{B1}))$
5
$K_{A,B}(R_{A2}), K_{B,KDC}(A, K_{A,B}, R_{B1})$
6
$K_{A,B}(R_{A2}-1, R_{B2})$
7
$K_{A,B}(R_{B2}-1)$

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

Authentication Using **Public-Key** Cryptography



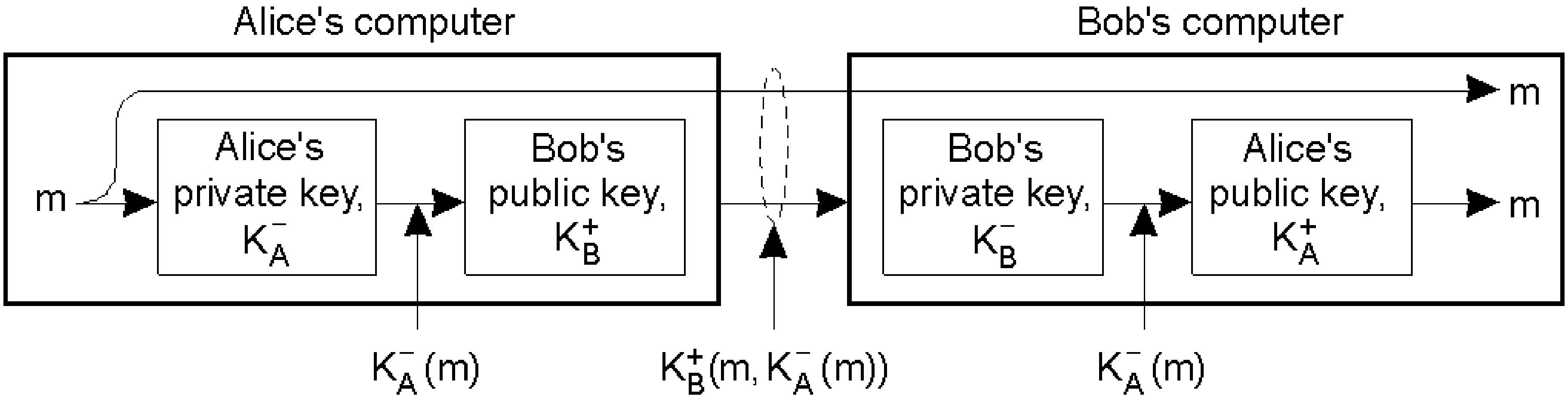
Mutual authentication in a public-key cryptosystem.

Message Integrity and Confidentiality

- Besides Authentication, a secure channel must guarantee:
  - **confidentiality** against interception *by using **encryption***
  - **message integrity** against modifications *by using **digital signatures***

Digital Signatures (1)

- Digital signature can be used to assure that modifications to a message will go unnoticed.


Alice's computer
Bob's computer
m
Alice's private key, K_A^-
Bob's public key, K_B^+
Bob's private key, K_B^-
Alice's public key, K_A^+
m
m
$K_A^-(m)$
$K_B^+(m, K_A^-(m))$
$K_A^-(m)$

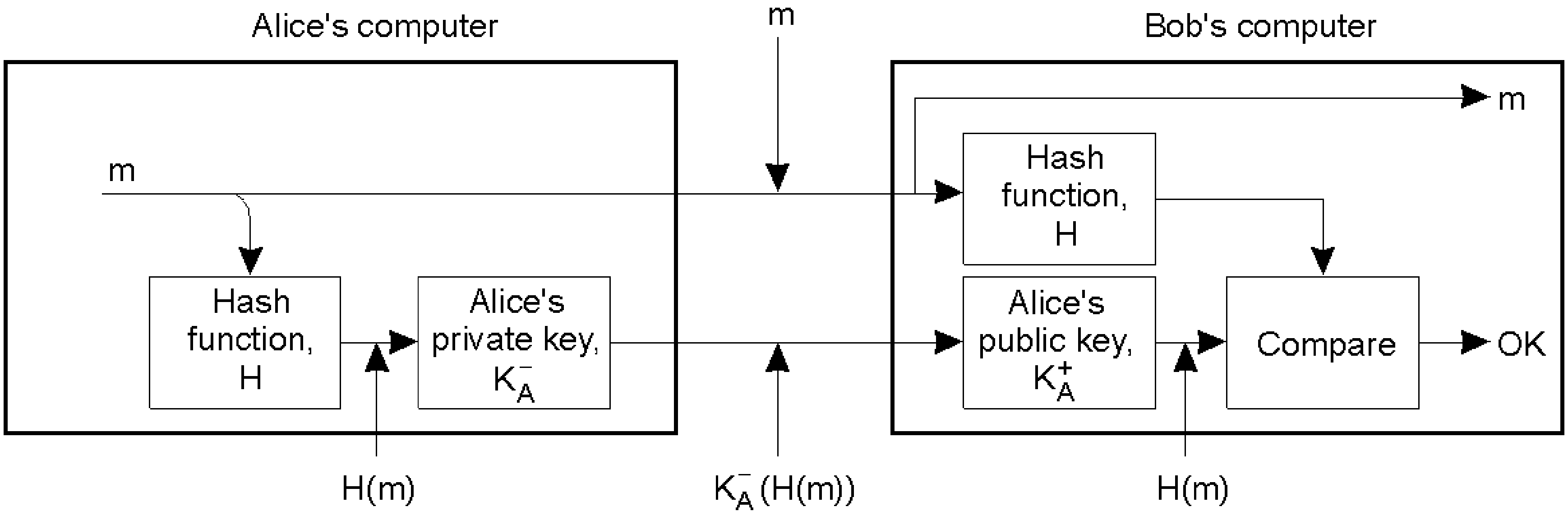
Digital signing a message using public-key cryptography.

Digital Signatures (2)

- Potential problems:
  - The Alice private key must remain a secret
    - *the message integrity is lost if the key was stolen*
  - What happens if Alice changes her key ?
    - *A central authority must keep track of key changing*
  - The message encryption may be costly in terms of processing requirements.
    - *Message digest can be used*

Digital Signatures (3)

- A message digest is a fixed-length bit string ***h*** obtained from a message ***m*** by means of a cryptographic hash function ***H***. If $\boldsymbol{m \neq m'} \Longrightarrow \boldsymbol{H(m') \neq H(m) = h}$



Digitally signing a message using a message digest.

Secure Group Communications

- In distributed systems it is also needed to set up secure communications among more than just two parties.

- For example,
  - **when a server must communication with a set of clients,**
  - **or when several replica servers must communicate using a multicast mechanism**

  by assuring security against modification, interception, and fabrication.

Confidential Group Communications

- If confidentiality must be assured:
  - **a simple scheme can be based on a secret key shared among all the entities in the group. (***vulnerable solution***)**
  - **a more complex scheme can use a separate shared key between each pair of entities. (***many keys***)**
  - **a public-key system can be used, each entity in the group has its own pair (public key, private key). N key pairs are needed. When an entity is not trustworthy is removed from the group.**

Secure Replicated Services (1)

- **Goal**: a client issues a request to a group of replicated servers and it needs to get a secure response from them.
- Reiter proposed a solution for a secure replicated server maintaining replication transparency (clients are unaware of the number of replicas).
- The solution is based on **secret sharing**: a group of entities share a secret, but none of them knows the entire secret.

Secure Replicated Services (2)

- We are seeking a solution by which at most ***c*** out of ***N*** servers can be corrupted by an intruder, but a correct response can be produced for the client.

- Each server ***Si*** produces a response ***ri*** that it returns to the client with a digital signature $\boldsymbol{Sig(S_i, r_i)}$. Let ***md(ri)*** denote the message digest computed by server ***Si***.

$$\boldsymbol{Sig(S_i, r_i)=K_i^-(md(r_i))}$$

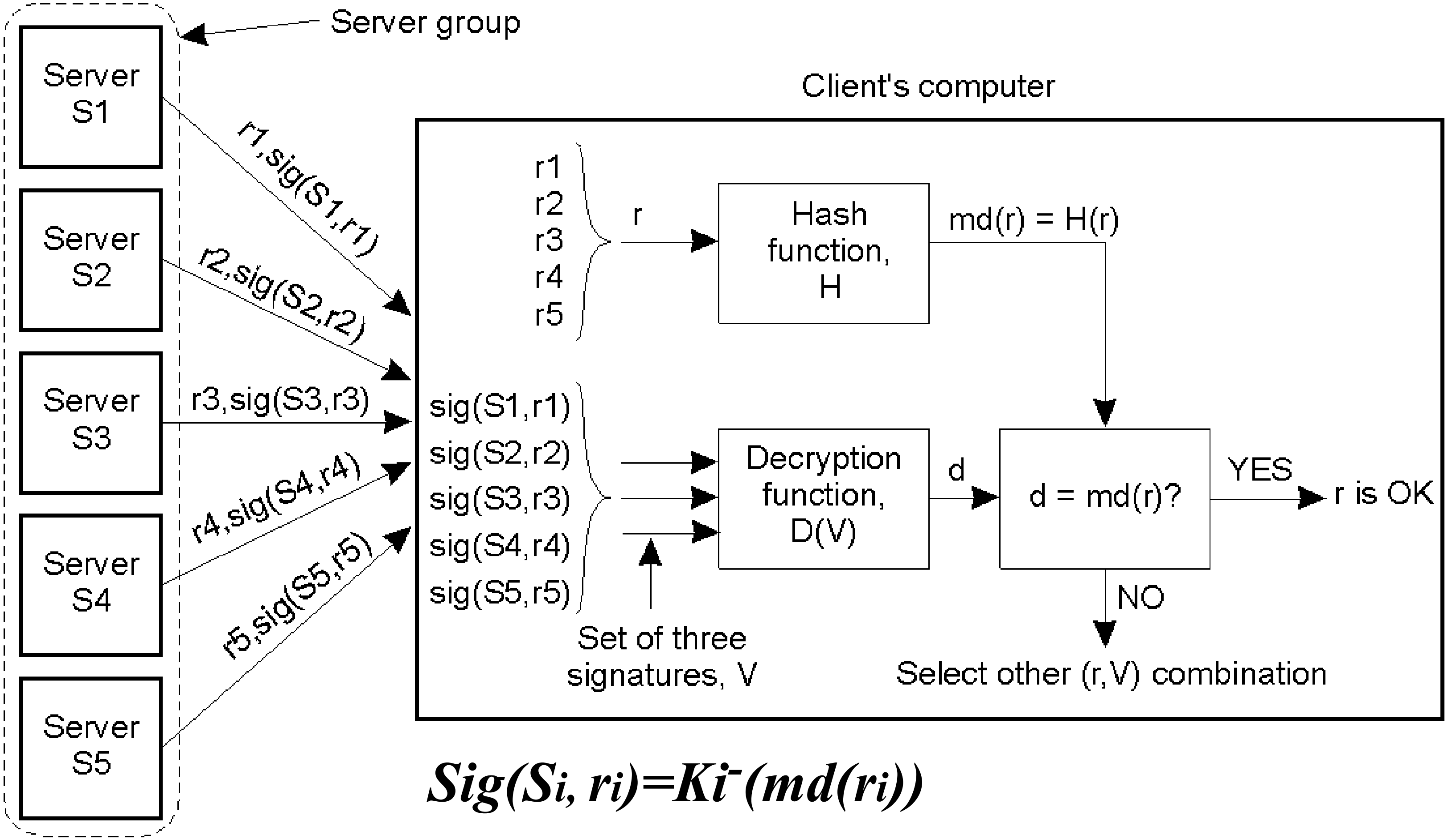
- The solution is based on the

  use of at least ***c+1*** signatures

  to construct a valid signature for the response.

Secure Replicated Services (3)

An example of 5 replicated servers able to tolerate 2 corrupted servers. A ***D(V)*** function of 3 sigs is used.



$Sig(S_i, r_i)=K_i^-(md(r_i))$

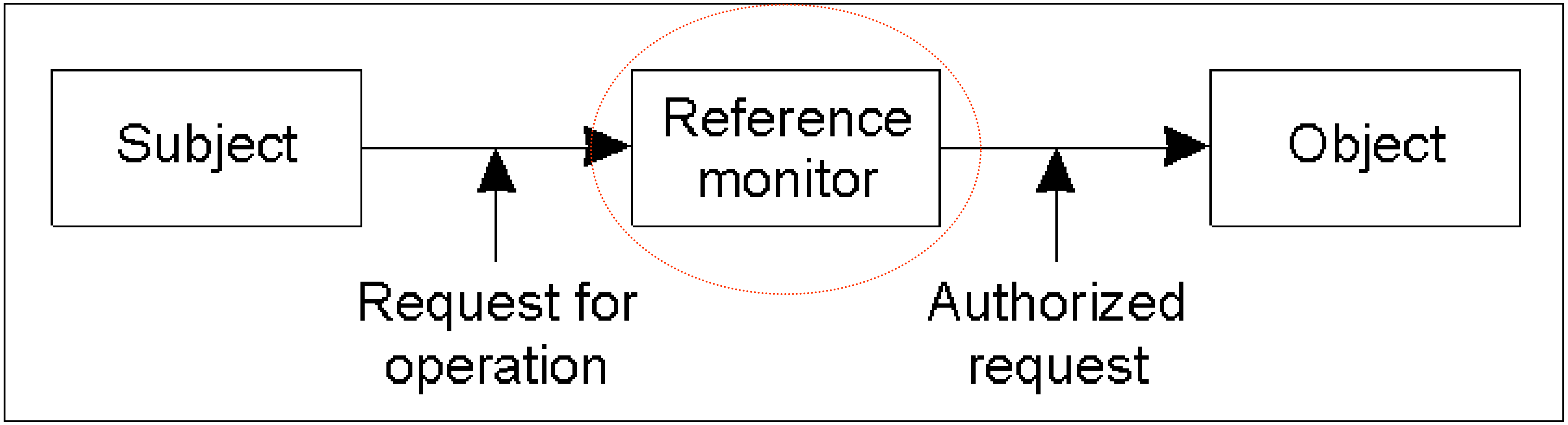
Sharing a secret signature in a group of replicated servers.

Threshold Scheme

- This strategy is called **(c+1,N)-threshold scheme**.
- In this scheme, a message can be divided in N parts, called **shadows**, since any c+1 shadows can be used to reconstruct a message.
- But using c or fewer shadows message reconstruction is not possible.
- Several implementations of the **(c+1,N)-threshold scheme** have been proposed.

General Issues in Access Control

- **Access control**: verifying access rights.
- **Authorization**: granting access rights.



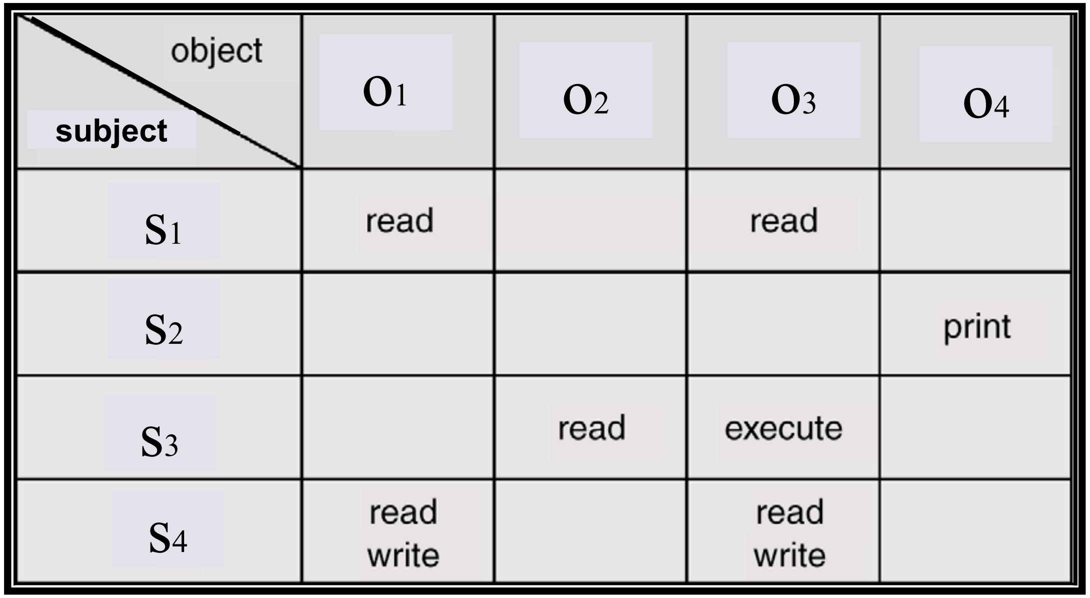
General model of controlling access to objects.

Access Control Matrix (1)

a) Access rights are generally modeled through an access matrix.

b) An object is represented by a column and a subject is represented by a row.

c) An entry M[s,o] specifies which operations the subject s can request on object o.



object subject	O_1	O_2	O_3	O_4
S_1	read		read	
S_2				print
S_3		read	execute	
S_4	read write		read write	

Access Control Matrix (2)

- An access matrix cannot be implemented as true matrix when a large number of objects and subjects must be considered. Too many entries are empty.

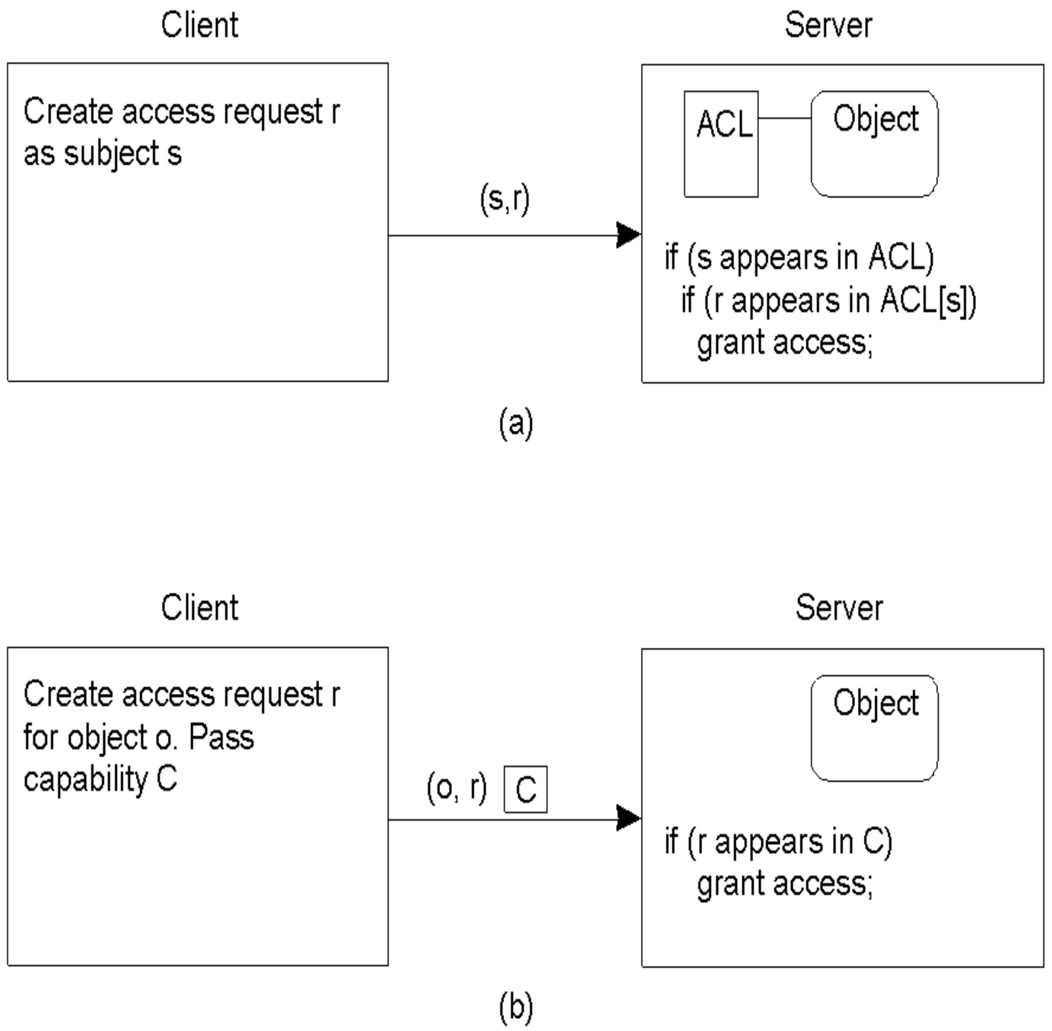
- Two main solutions:
  - **Access Control List (ACL) : each object maintains a list of access rights of subjects.**
  - **Capabilities : each subject has given a list of capabilities it has for objects.**

Access Control Matrix (3)

Comparison between ACLs and capabilities for protecting objects.

(a) Using an **ACL**

(b) Using **capabilities**



Client

Create access request r as subject s

(s,r)

Server

ACL

Object

```
if (s appears in ACL)
  if (r appears in ACL[s])
    grant access;
```

(a)

Client

Create access request r for object o. Pass capability C

(o, r) C

Server

Object

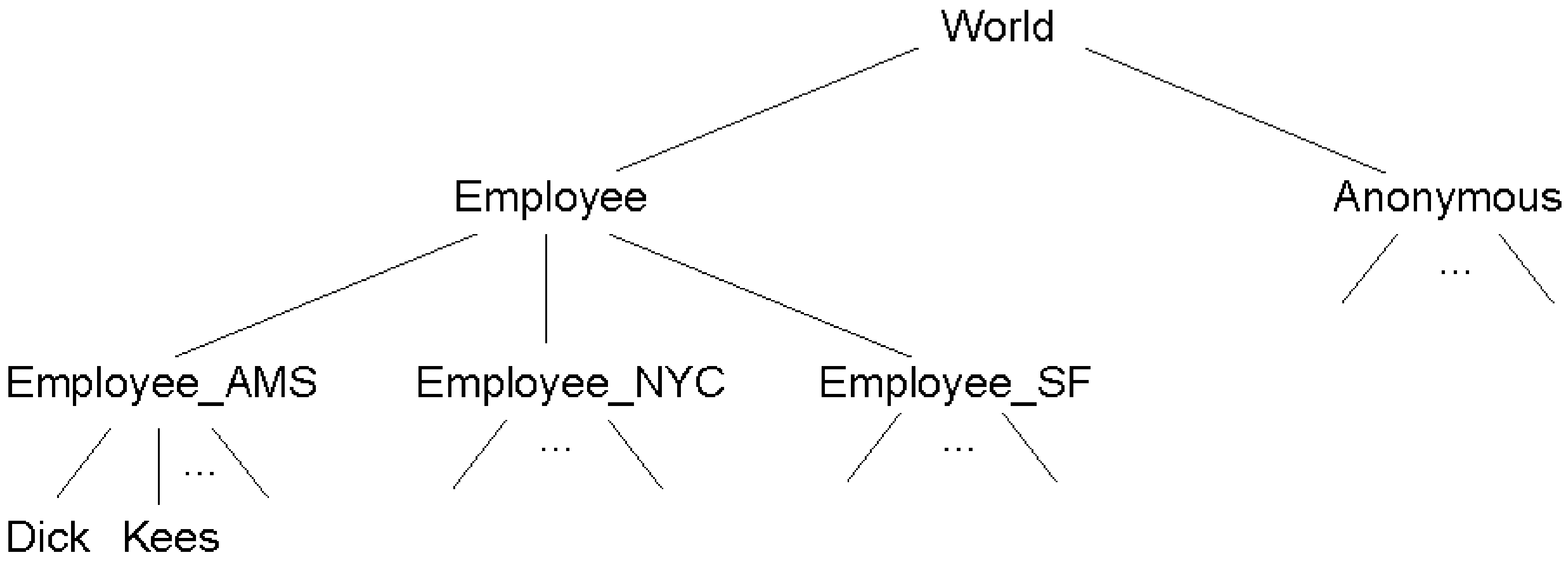
```
if (r appears in C)
  grant access;
```

(b)

Protection Domains (1)

ACLs and capability lists can be limited by using protection domains.

A **protection domain** is a set of *(object, access rights)*.



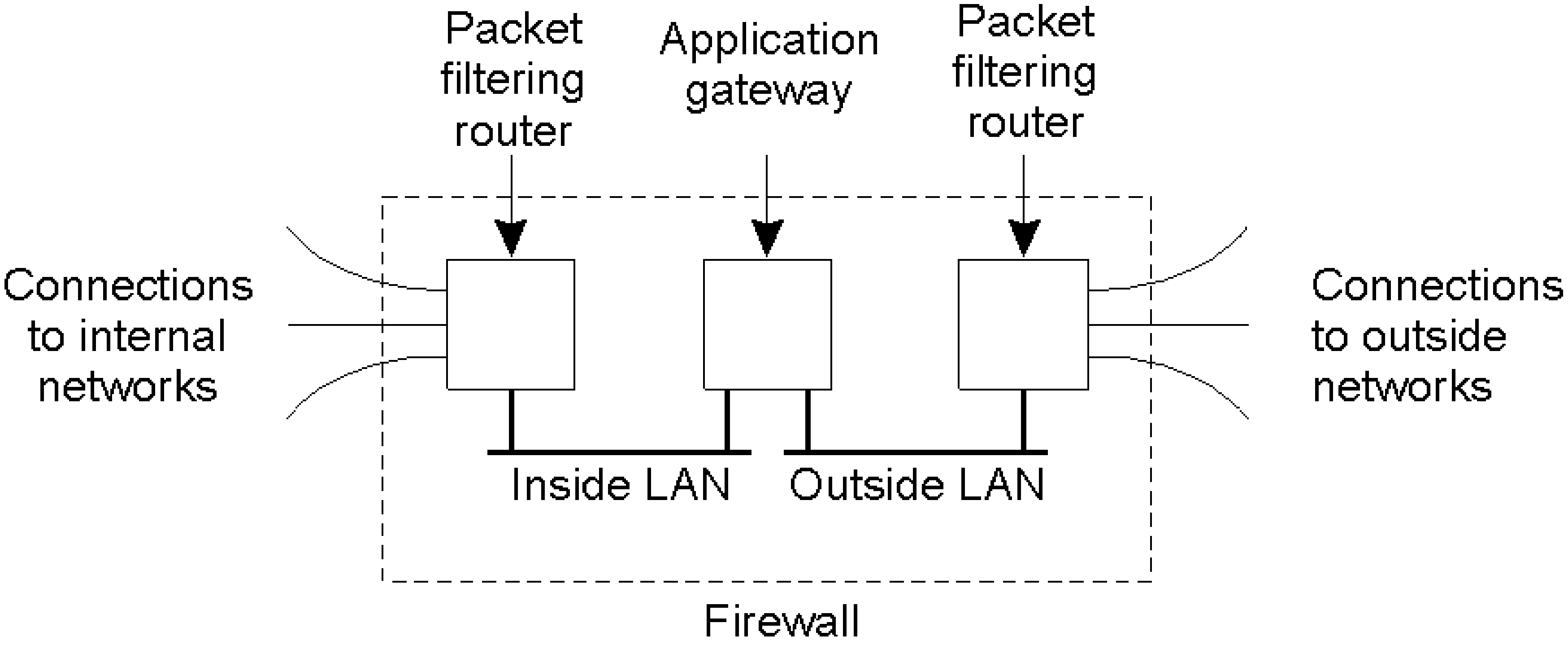
The hierarchical organization of protection domains as groups of users.

Protection Domains (2)

- Other mechanisms for efficient access control are:
  - subject certificates
  - role-based access
  - object grouping
  - combination of domain protection and object grouping.

Firewalls (1)

- When some entities in a distributed system do not respect the same rules other mechanisms are necessary for security.



A common implementation of a firewall.

Firewalls (2)

- A firewall is a critical resource in a distributed system.
- It should never fail!
- A distributed firewall should be more secure than centralized firewalls.

Secure Mobile Code

- When a mobile agent is moving across the Internet:

  1. The agent should be protected against malicious hosts that try to steal or modify its behavior.

  2. Host should be protected against malicious agents.

- Access control schemes are needed.

Protecting Agents (1)

- Whenever an agent moves to a host, that host should not allowed to steal or modify the agent **information** and **code**.
- If it not possible to guarantee no modification, at least agents should be organized in such a way that **modifications should be detected**.
- This approach is used in Ajanta by using
  - ***read-only state,***
  - ***append-only logs***, and
  - ***selective revealing***.

Protecting Agents (2)

- A **read-only state is a collection of data items signed by the agent's owner**.

- An **encrypted message digest is associated** to a read-only state.

- A host may check the correctness of the state checking the signed message digest of the original state.

Protecting Agents (3)

- **Append-only logs** deny data modifications or removal but allow host to add data to the log.
- Data are stored in the log using the public key of the owner and a checksum function.
- When data are back to the owner, it can check data security using its private key.

Protecting Agents (4)

- **Selective revealing** of state is built by using an array of data items, where each item is for a specific server.
- Data items in the array are encrypted by using the public key of the designated server.
- The global array is signed by the array owner to assure integrity.

Protecting Hosts (1)

- Hosts must be protected against malicious code of agents, applets, ect.
- Here what is important is detect malicious code before execution, otherwise it could be too late !
- Used techniques are:
  - ***sandbox***,
  - ***playground***, and
  - ***stack introspection***.

Protecting Hosts (2)

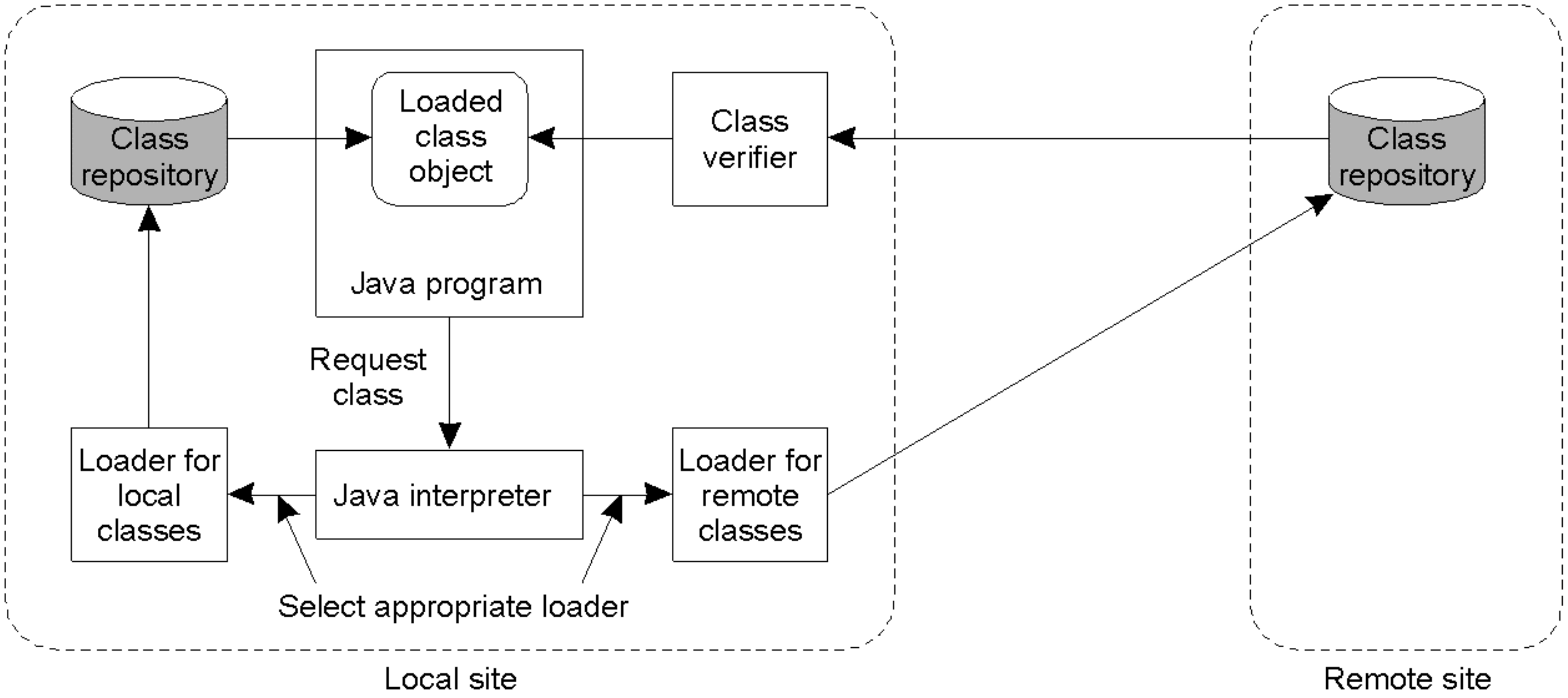
- A **sandbox** is implemented by inserting addition instruction to make run-time checks on agents.
- Java allows sandbox implementation because its class-based approach.

1.Only trusted class loaders are used.

2. A byte-code verifier is used to check instructions that can corrupt the stack or memory.

3. A security manager to perform run-time checks. Java programs can be forced to make use of the security manager that controls file and data access.

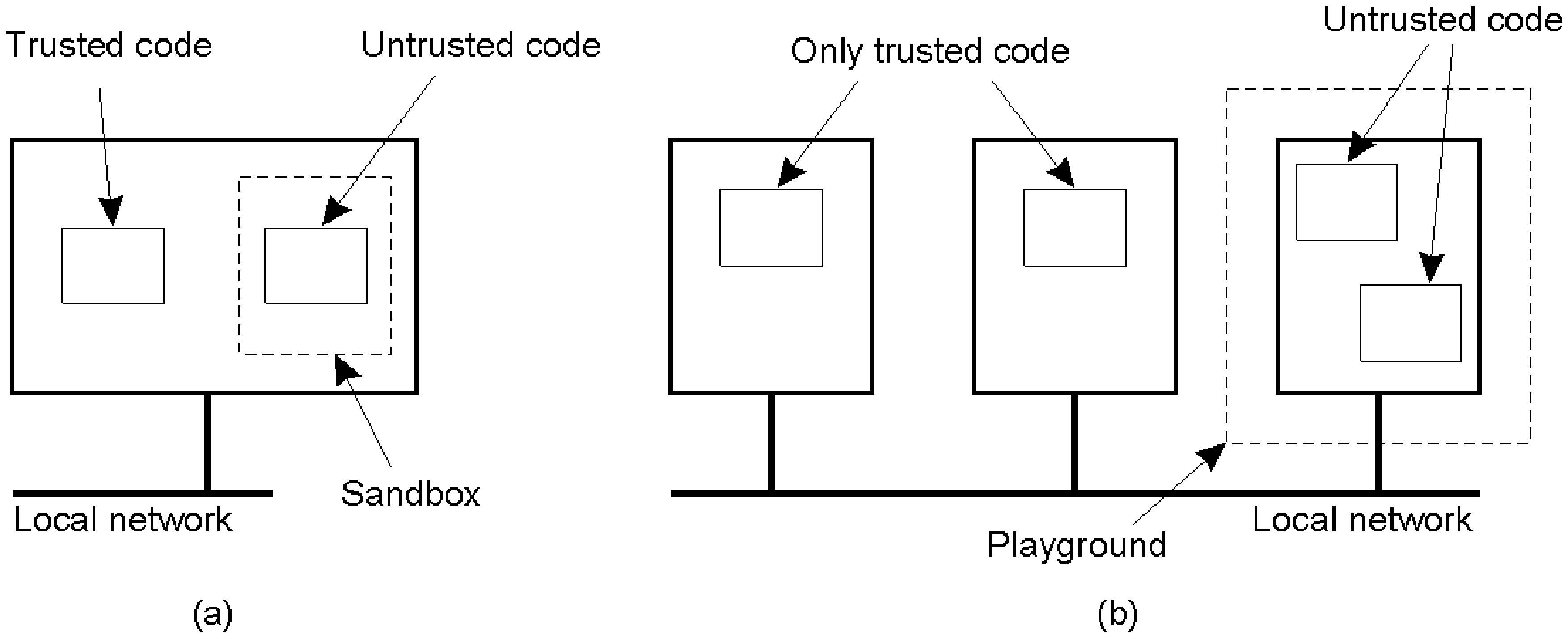
Protecting Hosts (3)



The organization of a Java sandbox.

Protecting Hosts (4)

A **playground** is a separate host exclusively reserved for running mobile code. No mobile code is run on hosts outside of the playground.

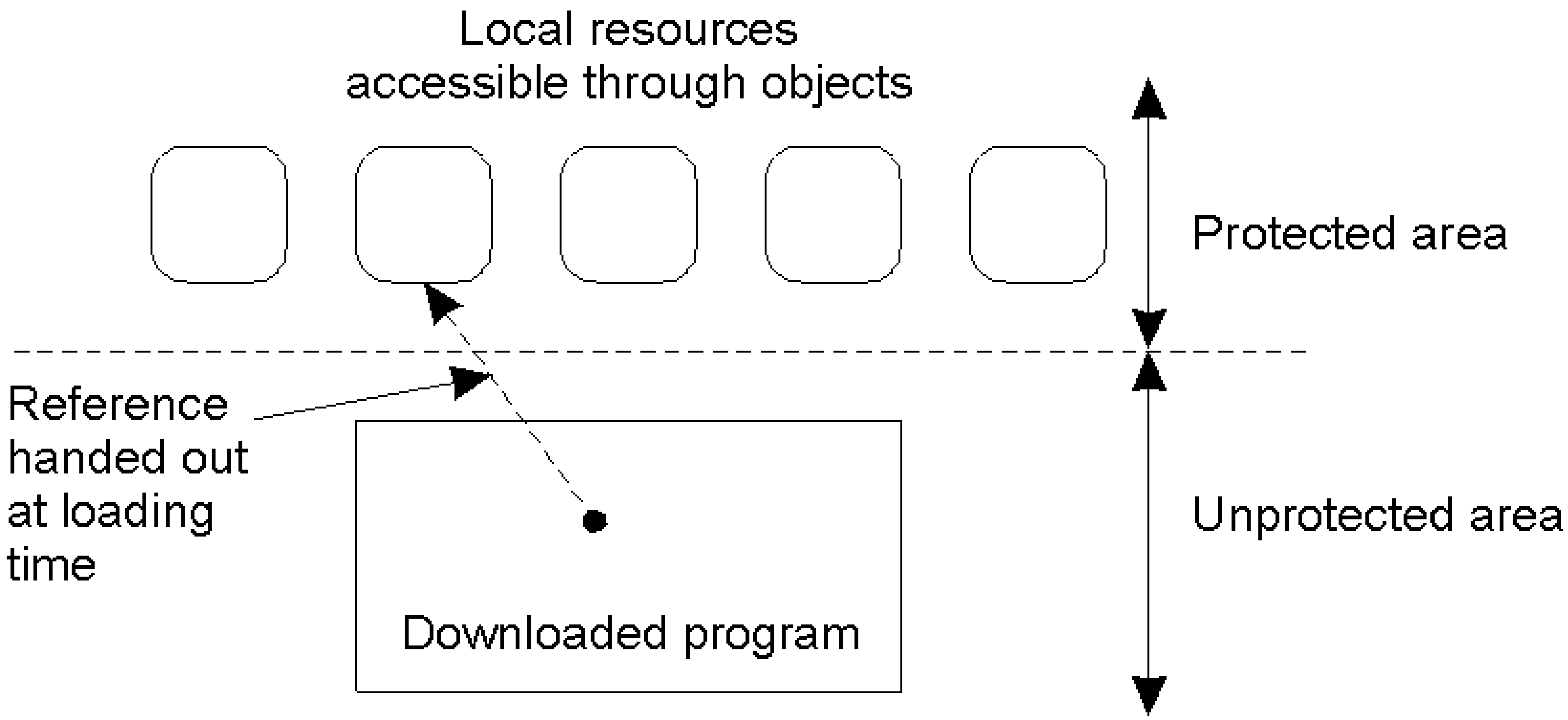


(a) A sandbox **(b) A playground**

Protecting Hosts (5)

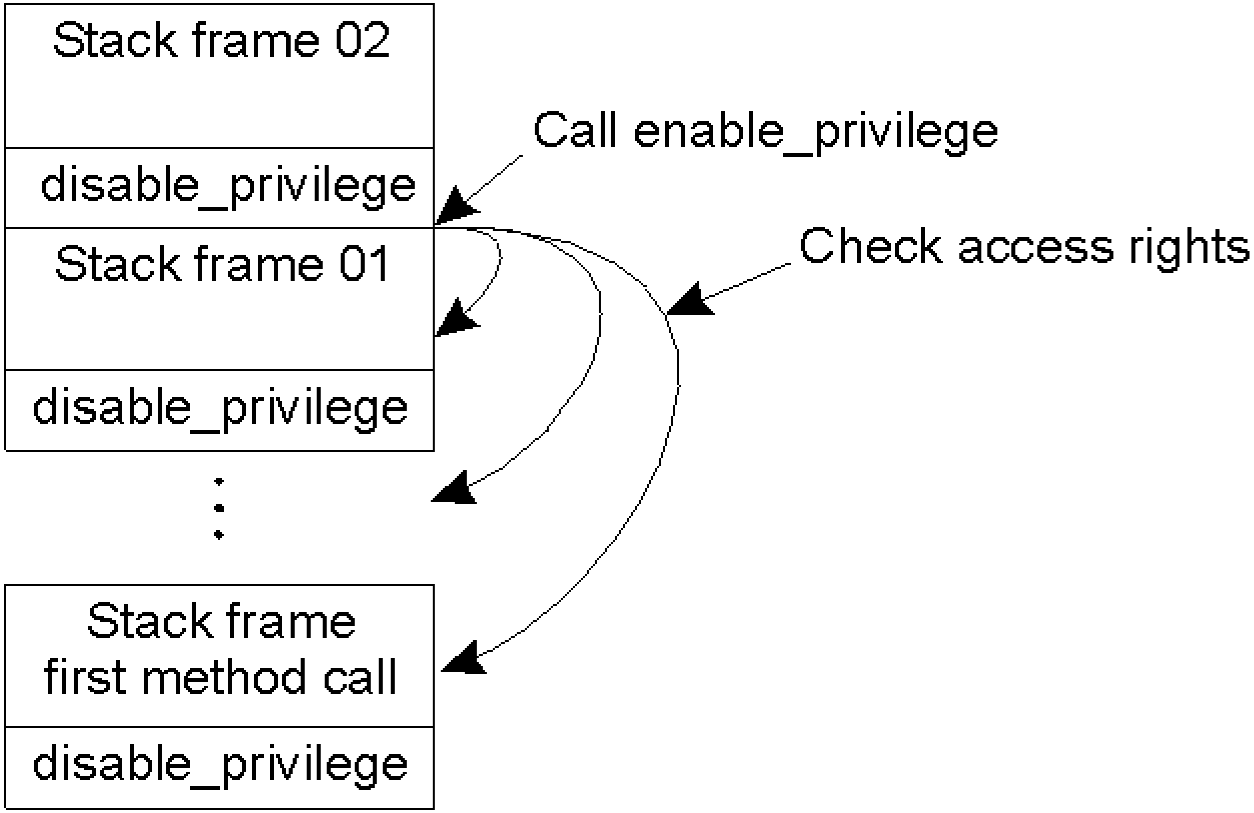
- Other techniques to secure mobile code is based on **code signing** that implements program authentication.
- Some mechanisms are
  - **object references as capabilities** (Java class properties facilitate this approach),
  - **stack introspection** (based on **enable_privilege** and **disable_privilege** for method invocation), and
  - **name space management** (access to file that implement classes accesing the require resources).
- **Secure operating systems** aware of mobile code running.

Protecting Hosts (6)



The principle of using Java object references as capabilities.

Protecting Hosts (7)



The principle of stack introspection.

Security Management

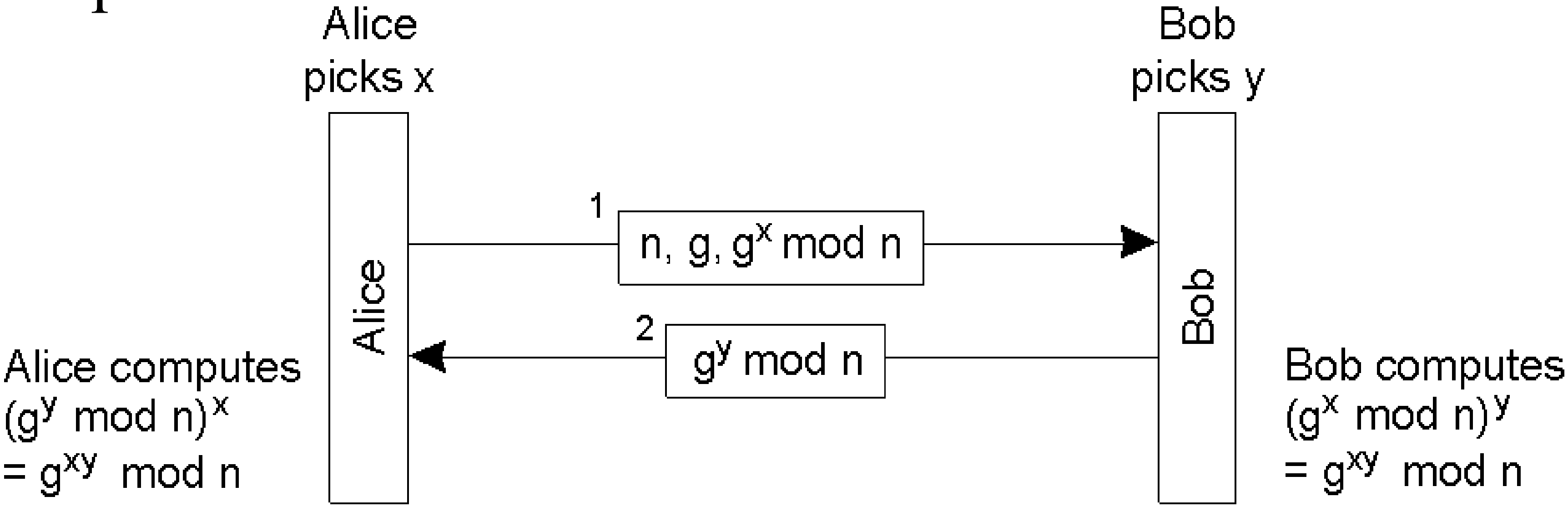
- Three main aspects of management of security issues in a distributed system are
  - cryptographic key distribution and management,
  - secure server addition to a server group, and
  - authorization management and rights delegation.

Key Management

- How cryptographic keys are generated ?
- How cryptographic keys are distributed ?
- How cryptographic keys are revoked ?

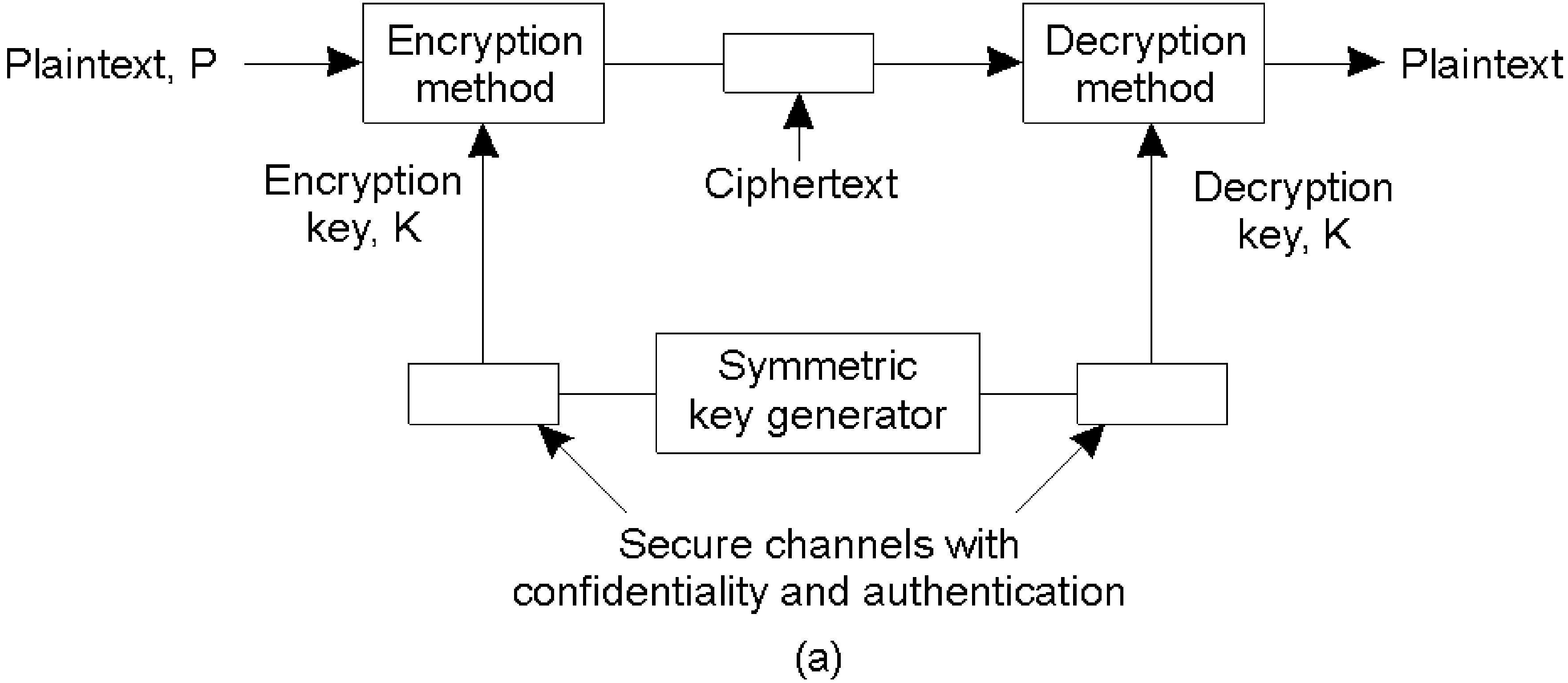
Key Establishment

- Diffie and Hellman proposed a scheme for establishing a shared key across an insecure channel.
- The scheme is based on the use of two large numbers ***n*** and ***g*** made public and two large random number ***x*** and ***y*** kept secret.



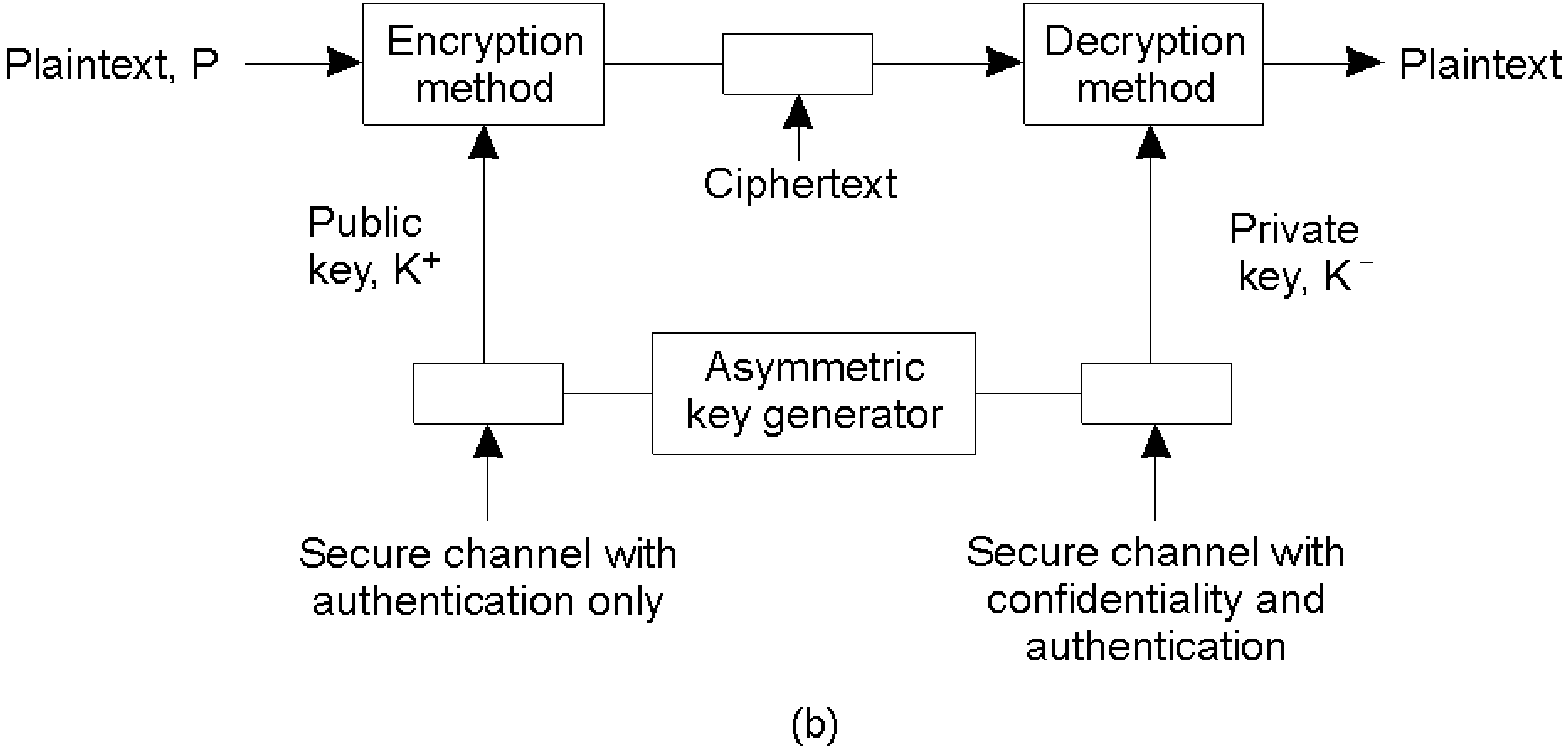
g^x mod n is the Alice Public key, **x** is her private key.

Key Distribution (1)



Shared Secret-key distribution based on secure channels

Key Distribution (2)



Public-key distribution through secure channels

Key Distribution (3)

- Public-key distribution takes place by means of **public-key certificates** that consist of a ***public key*** and an ***entity identifier*** specifying to whom the key is associated (user, process, resource).

- The certificate is signed by the private key $\boldsymbol{K^{-}_{CA}}$ of a **certification authority**.

- A hierarchy of certification authorities is available. (Ex: Privacy Enhanced Mail).

Certificate Lifetime

- Certificate lifetime is important : it must be limited and revoke mechanisms are needed for it.

- Revocation mechanisms used today are:
  - **Certificate Revocation List** (CRL) published regularly by a certification authority.

  - **Certification expiration date**
    - after some time with revocation checking a CRL.
    - Nearly to zero, with continuous online checking.

Secure Group Management (1)

- Certification Authorities, such as **Key Distribution Centers**, must be secure and offer high availability.

- Replication of these services can be a solution for high availability (*see the Reiter scheme based on shared keys*).

- A problem with groups of servers it to ensure **group integrity** when a new server asks to join.

Secure Group Management (2)

A group ***G*** uses

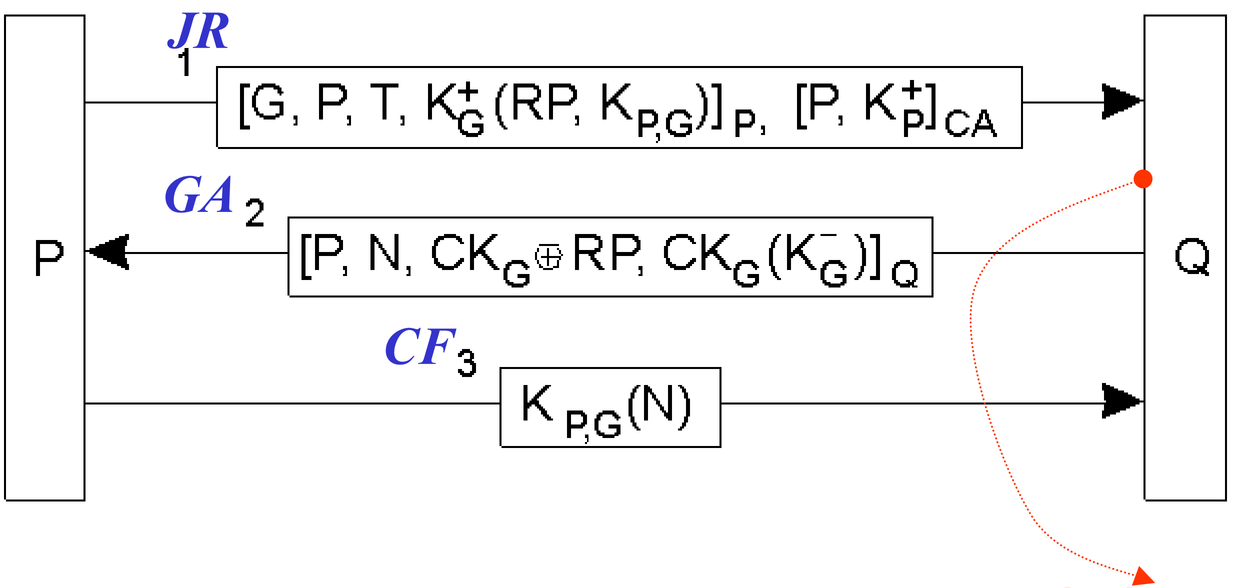
- a secret key $\boldsymbol{CK_G}$ **shared by all the group members** for exchanging messages among them

  and

- a public/private key pair $(\boldsymbol{K^{+}_{G}}, \boldsymbol{K^{-}_{G}})$ for communication with external entities (nongroup members).

Secure Group Management (3)

When a process ***P*** wants to join a group ***G***, it sends a Joint Request (JR) to a group member a with a set of parameters.



P
Q
JR_1
$[G, P, T, K^+_G(RP, K_{P,G})]_{P,}\ [P, K^+_P]_{CA}$
GA_2
$[P, N, CK_G \oplus RP, CK_G(K^-_G)]_Q$
CF_3
$K_{P,G}(N)$

G	**: the group**
P	**: the req. process**
CK_G	**: shared secret key**
RP	**: reply pad**
$K_{P,G}$	**: gener. secret key**
K^+_{G},	**: group public key**
K^-_G	**: group private key**
K^+_P,	**: P public key**
N	**: nonce**

Authent. Checks and Group protocol

Securely admitting a new group member.

Authorization Management

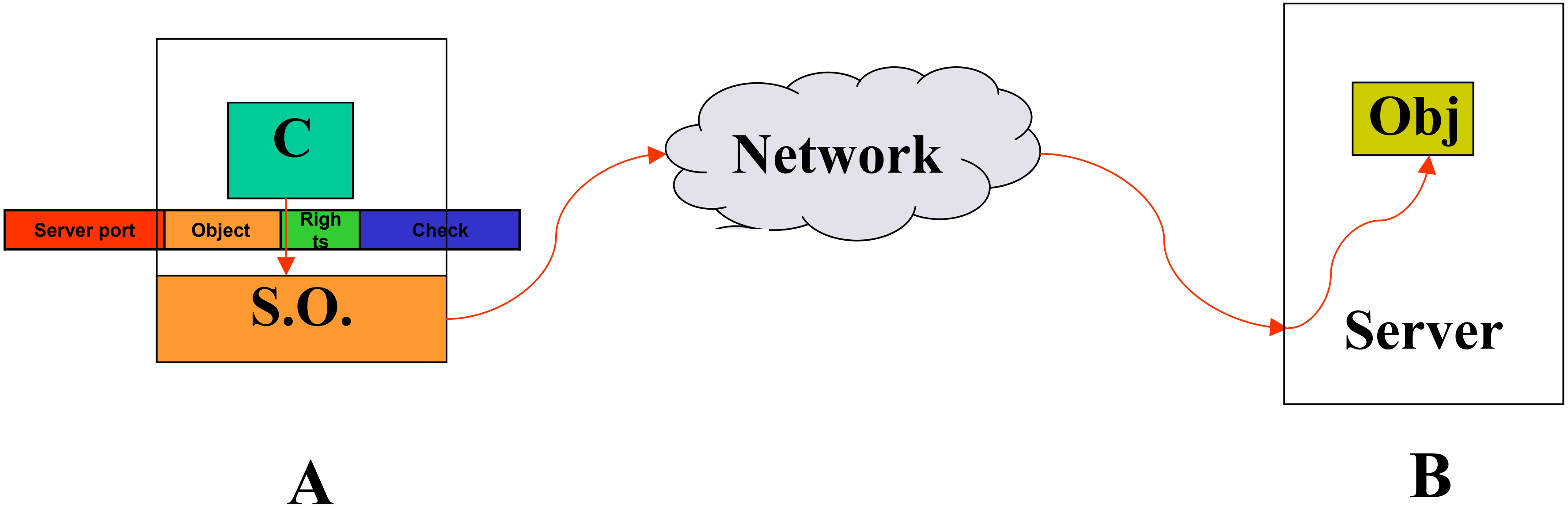
- Two simple schemes to be used in a distributed system managing access rights are
  - creation of an account of each user on each host, (*e.g., network operating systems*)
  - creation of a single account on a central server.

Capabilities and Attribute Certificates (1)

- A more flexible scheme can be based on **capabilities with holders and associated to a resource**.
- Different implementations of capabilities exist.
- **Amoeba** is an object-based distributed system.
- Each object reside on a server and clients can access remote objects by using proxies.

Capabilities and Attribute Certificates (2)

- In Amoeba, to invoke an operation on an object, a client passes a capability to the local O.S. that locates the server for the object and issues an RPC to the server.



Capabilities and Attribute Certificates (2)

48 bits	24 bits	8 bits	48 bits
Server port	**Object**	**Rights**	**Check**

A capability in Amoeba.

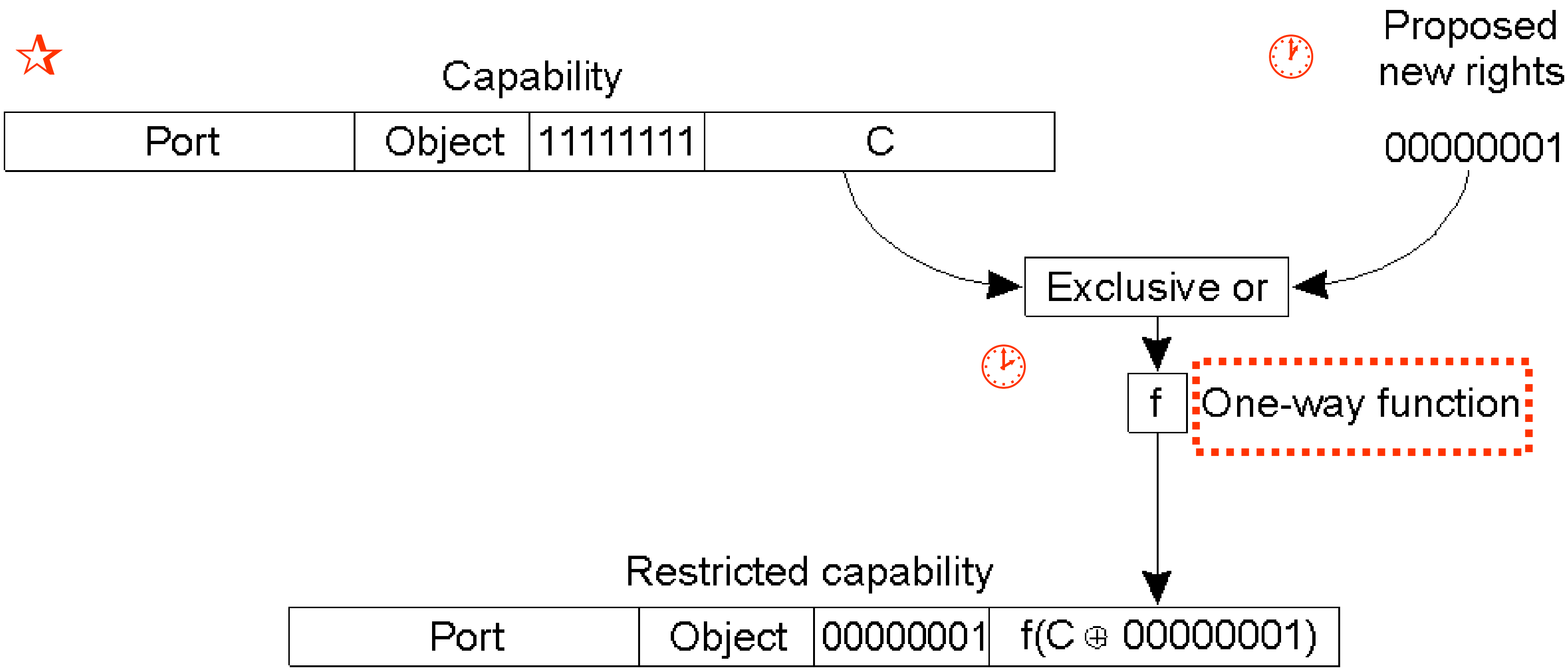
- A capability contains the **server address**, the **object id**, the **access rights** and a check field.

- The **check** field is initialized to a random value.

- When the server receives a request to perform an operation the **check** field is verified.

Capabilities and Attribute Certificates (3)

A **restricted capability** can be created when the owner makes a request and passed to clients.



Generation of a restricted capability from an owner capability.

Capabilities and Attribute Certificates (4)

- When a capability comes back to the server, it can see that is not an owner capability and it verifies the check field.
- A client cannot add rights to a restricted capability.
- Protection is implemented through the one-way function.

Capabilities and Attribute Certificates (5)

- Some distributed systems use **attribute certificates**.
- Attribute certificate lists <*attribute, value*> pairs indicate the access rights that the holder has with respect to the identified resource.
- Attribute certificates are managed by the so called **attribute certificate authorities** that can be different from servers managing the resource and are used to sign the access rights listed in a certificate.

Delegation (1)

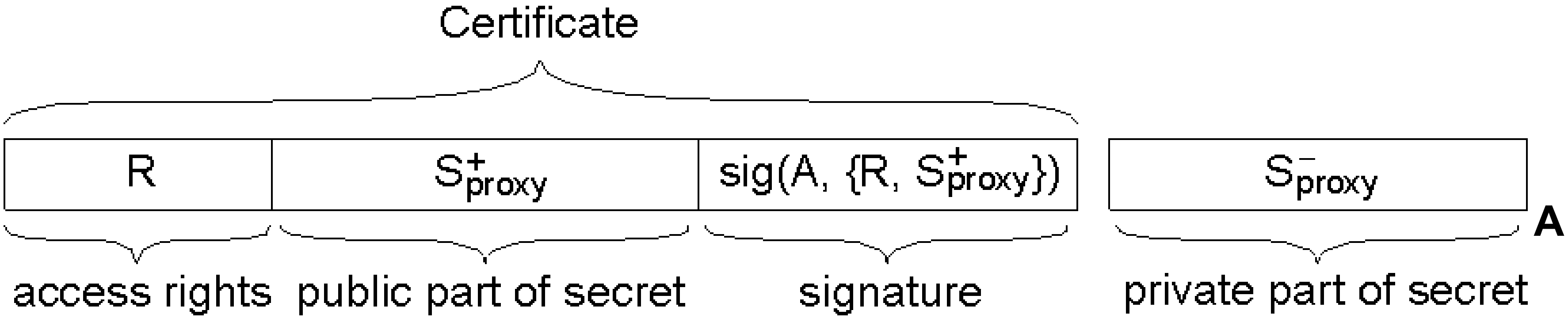
- **Delegation** of access rights is a helpful technique for distributed applications: passing rights from a process to another is important in several cases (e.g., printer, compiler, file handler) to avoid data transfer overhead.

- **Delegation proxies** are tokens used to pass rights from a process to another process that can act on a resource.

- Delegation can be **explicit** ("*A says that B has rights R*") or **implicit** ("*The holder has rights R*"). This second case needs more secure implementation.

Delegation (2)

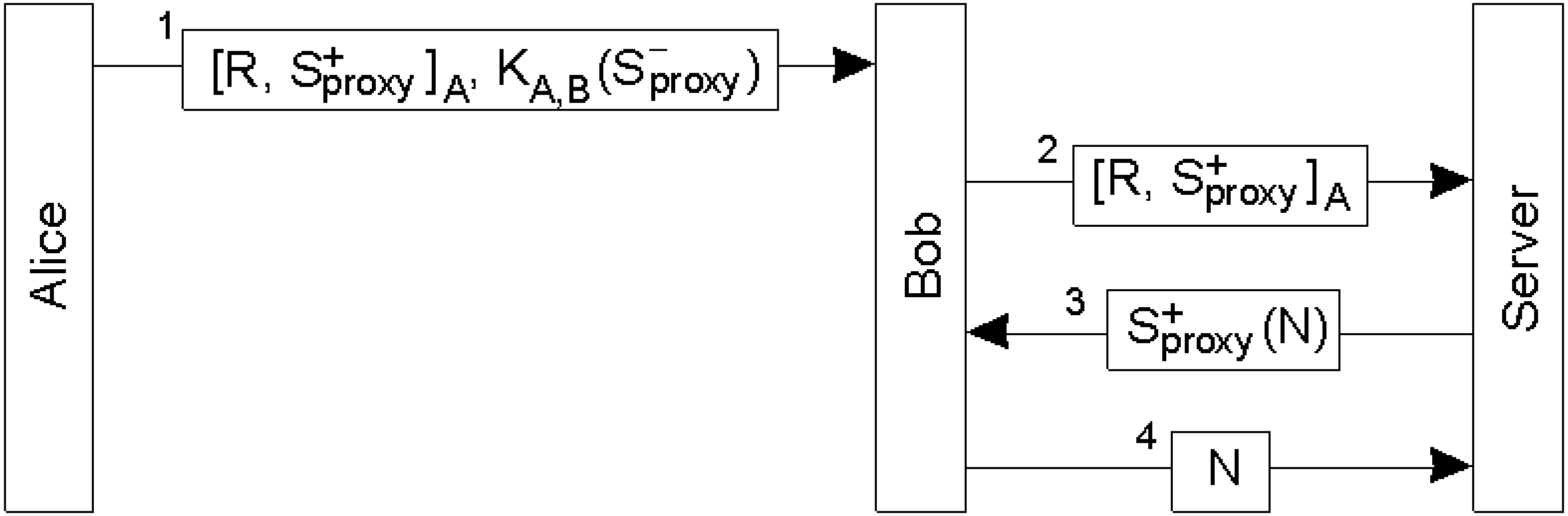
- **A Delegation Proxy** contains two parts
  - **the delegated access rights,**
  - **the public part of the secret to authenticate the holder**
  - **the signature of A to protect against modifications,**
  - **the private part of the secret to authenticate the holder**



The general structure of a delegation proxy.

Delegation (3)

- Bob asks the server to operate on an object whose owner Alice who passed him access rights.
- Bob uses the private part of the secret S^{-}_{proxy} to authenticate itself as the rightful owner of the certificate.



Using a proxy to delegate and prove ownership of access rights.

Example: Kerberos (1)

- **Kerberos** is a system designed at MIT to support security implementation in distributed systems. In particular, it assists clients to set up secure channels with a server.

- Kerberos is based on the **Needham-Schroeder authentication protocol**.

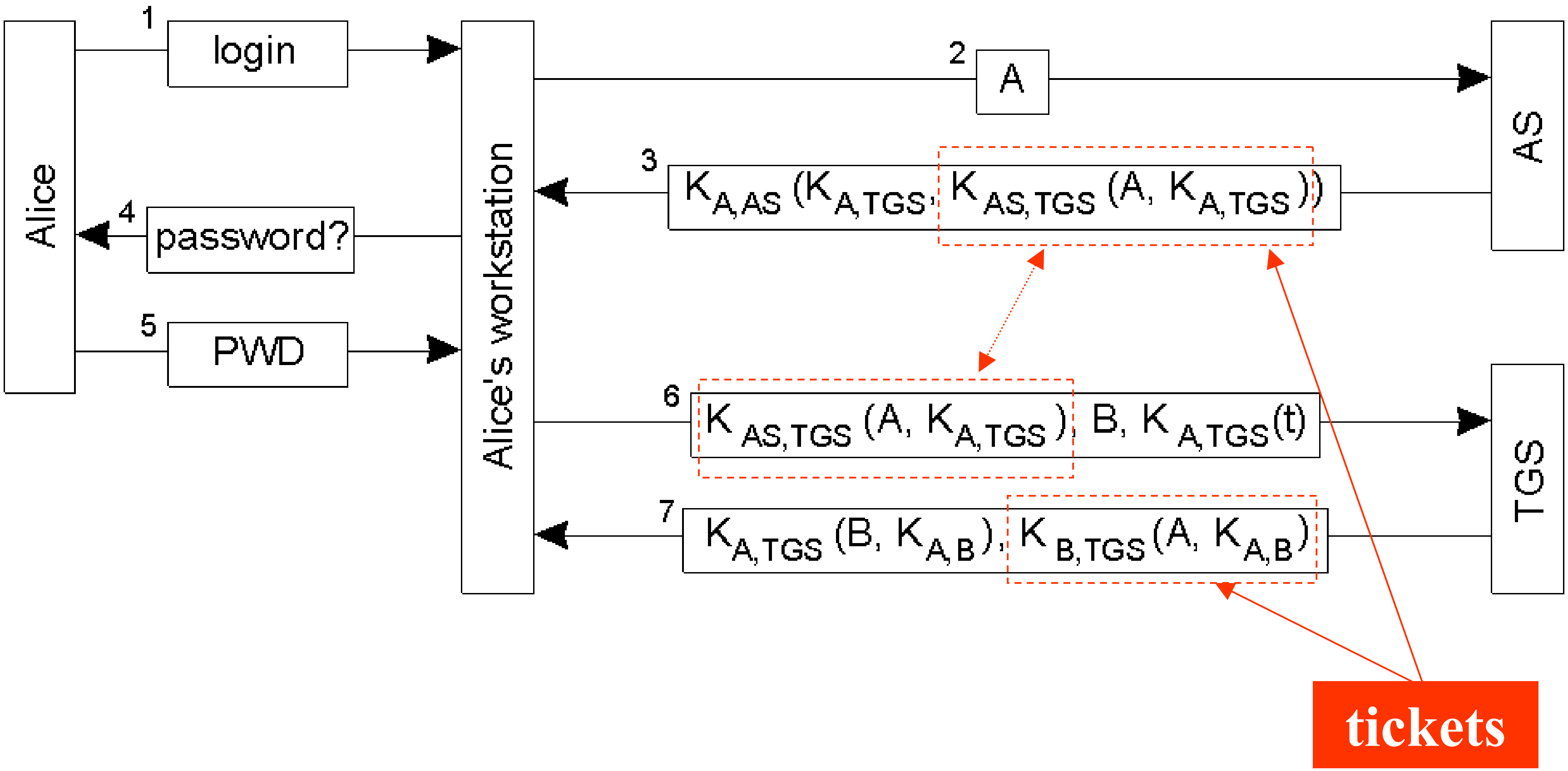
- Kerberos uses two main components:
  - The Authentication Server (AS)
  - The Ticket Granting Service (TGS).

Example: Kerberos (2)

- The **Authentication Server** (AS) authenticates a user and provides a key to be used to set up secure channels with a server.

- The **Ticket Granting Service** (TGS) set up secure channels with a server using tickets (encrypted secret keys).

Example: Kerberos (3)



Alice

Alice's workstation

AS

TGS

1 login

2 A

3 $K_{A,AS}(K_{A,TGS}, K_{AS,TGS}(A, K_{A,TGS}))$

4 password?

5 PWD

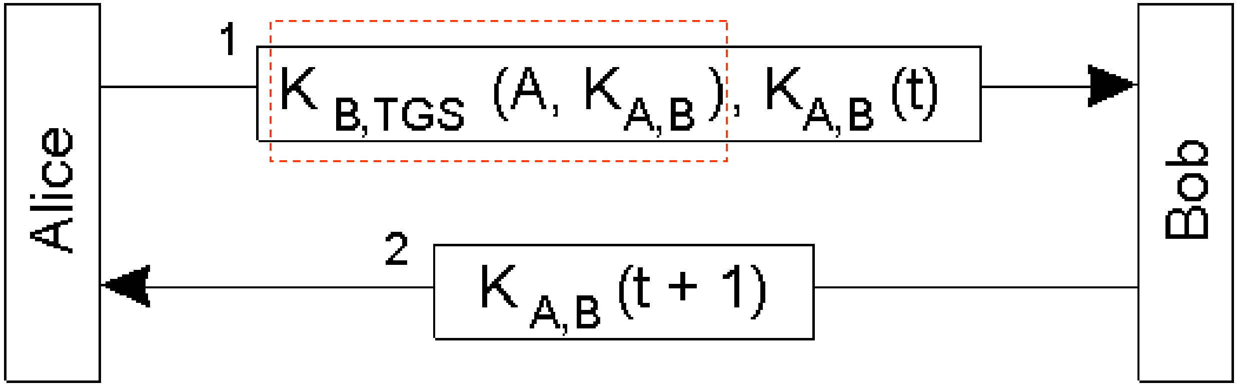
6 $K_{AS,TGS}(A, K_{A,TGS}), B, K_{A,TGS}(t)$

7 $K_{A,TGS}(B, K_{A,B}), K_{B,TGS}(A, K_{A,B})$

tickets

Authentication in Kerberos.

Example: Kerberos (4)



Setting up a secure channel in Kerberos.