Security

Strategies for securing Distributed Systems

Generally very similar to techniques used in a non-distributed system, only much more difficult to implement …

Difficult to get right, impossible to get perfect!

Security Topics

1. Providing a *secure communications channel* – authentication, confidentiality and integrity.

2. Handling *authorization* – who is entitled to use what in the system?

3. Providing effective *Security Management*.

4. Example systems: *SESAME* and *e-payment systems*.

Types of Threats

- **Interception** – unauthorized access to data.
- **Interruption** – a service becomes unavailable.
- **Modification** – unauthorized changes to, and tampering of, data.
- **Fabrication** – non-normal, additional activity.

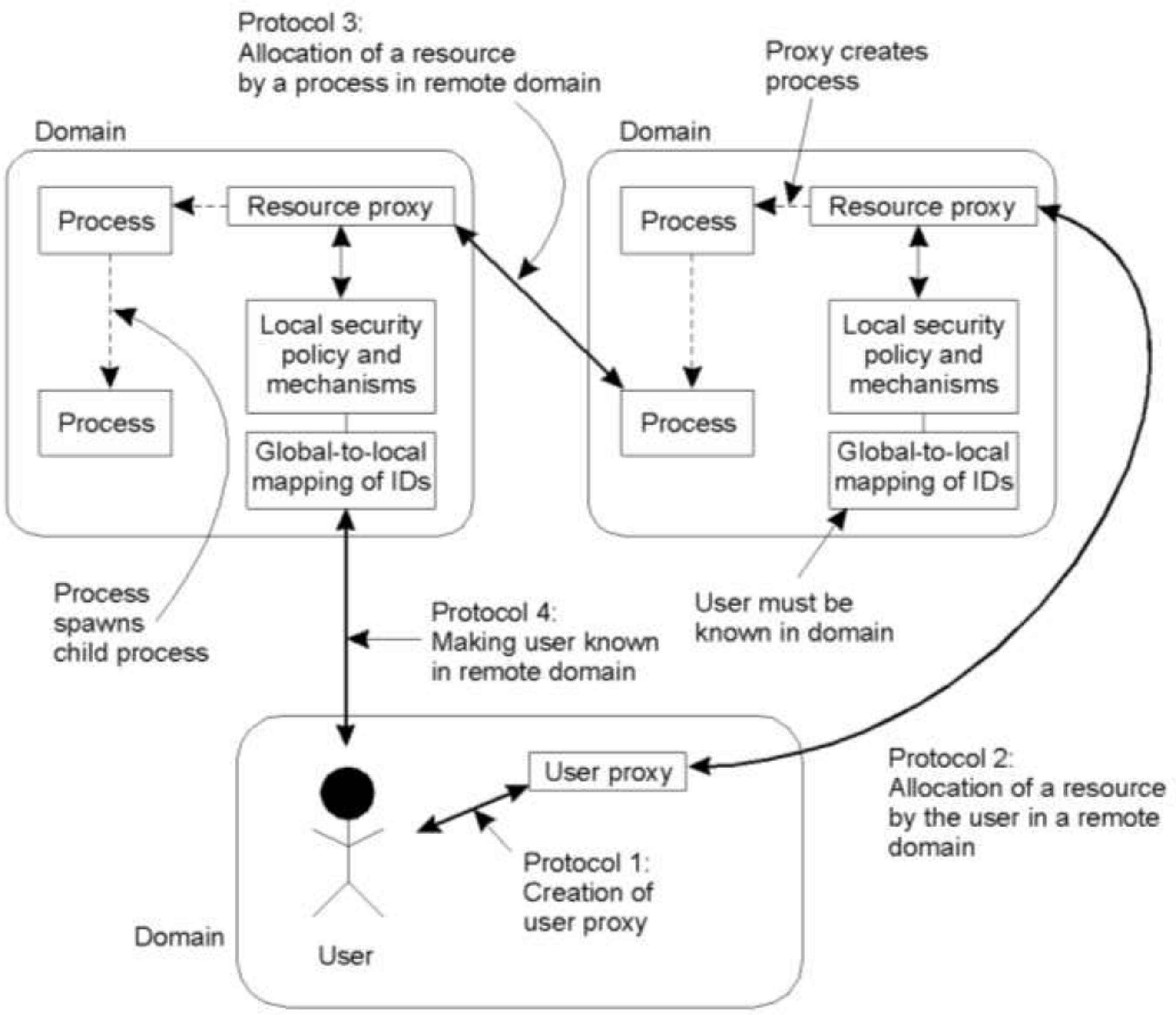
Security Mechanisms

- **Encryption** – fundamental technique: used to implement confidentiality and integrity.
- **Authentication** – verifying identities.
- **Authorization** – verifying allowable operations.
- **Auditing** – who did what to what and when/how did they do it?

Key Point

Matching *security mechanisms* to *threats* is only possible when a *Policy* on security and security issues exists.

Example: The Globus Security Architecture


Protocol 3: Allocation of a resource by a process in remote domain
Proxy creates process
Domain
Domain
Process
Resource proxy
Process
Resource proxy
Local security policy and mechanisms
Local security policy and mechanisms
Process
Process
Global-to-local mapping of IDs
Global-to-local mapping of IDs
Process spawns child process
Protocol 4: Making user known in remote domain
User must be known in domain
User proxy
Protocol 2: Allocation of a resource by the user in a remote domain
Protocol 1: Creation of user proxy
Domain
User

Refer to the textbook for additional details on this DS security architecture.

Design Issues

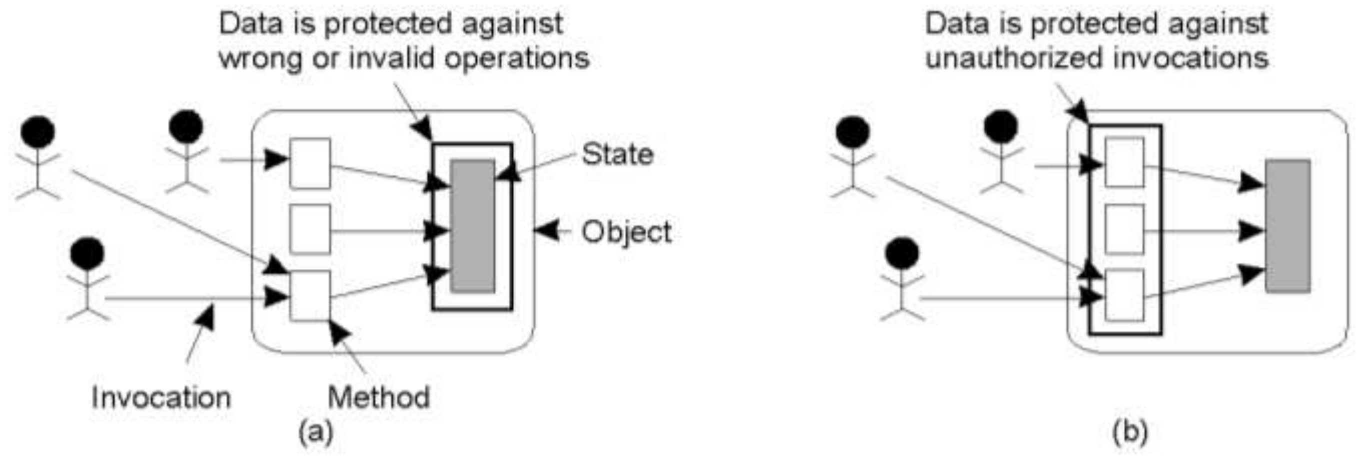
There are three main design issues to consider when considering security:

1. Focus of Control.

2. Layering of Security Mechanisms.

3. Simplicity.

Design Issue: Focus of Control

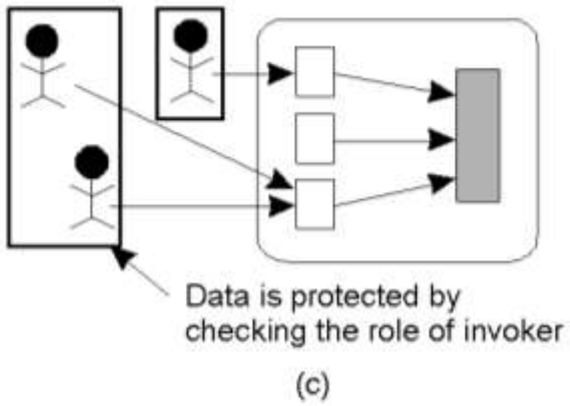

Data is protected against wrong or invalid operations
State
Object
Invocation
Method
(a)
Data is protected against unauthorized invocations
(b)

Three approaches for protection against DS security threats:

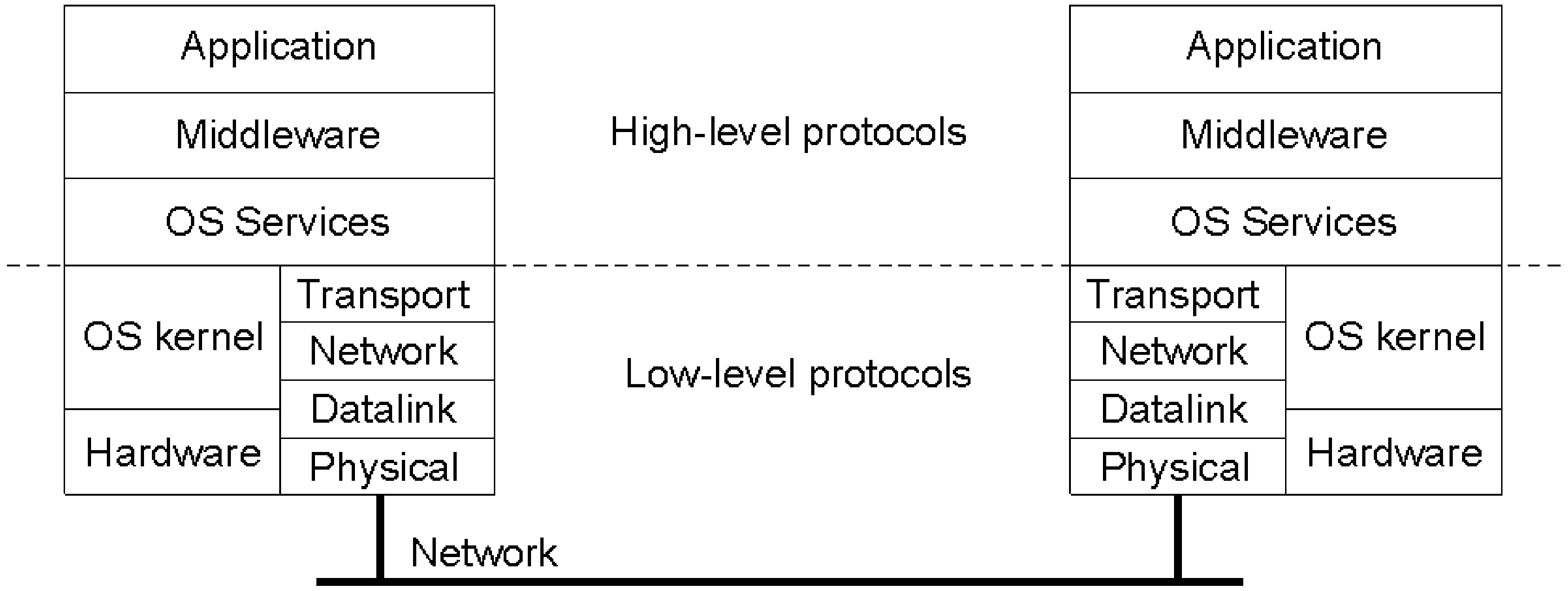
a) Protection against *invalid operations* - integrity.

b) Protection against *unauthorized invocations* – access control mechanisms.

c) Protection against *unauthorized users* – defining security roles.



Design Issue: Layering of Security Mechanisms



The logical organization of a distributed system into several layers.

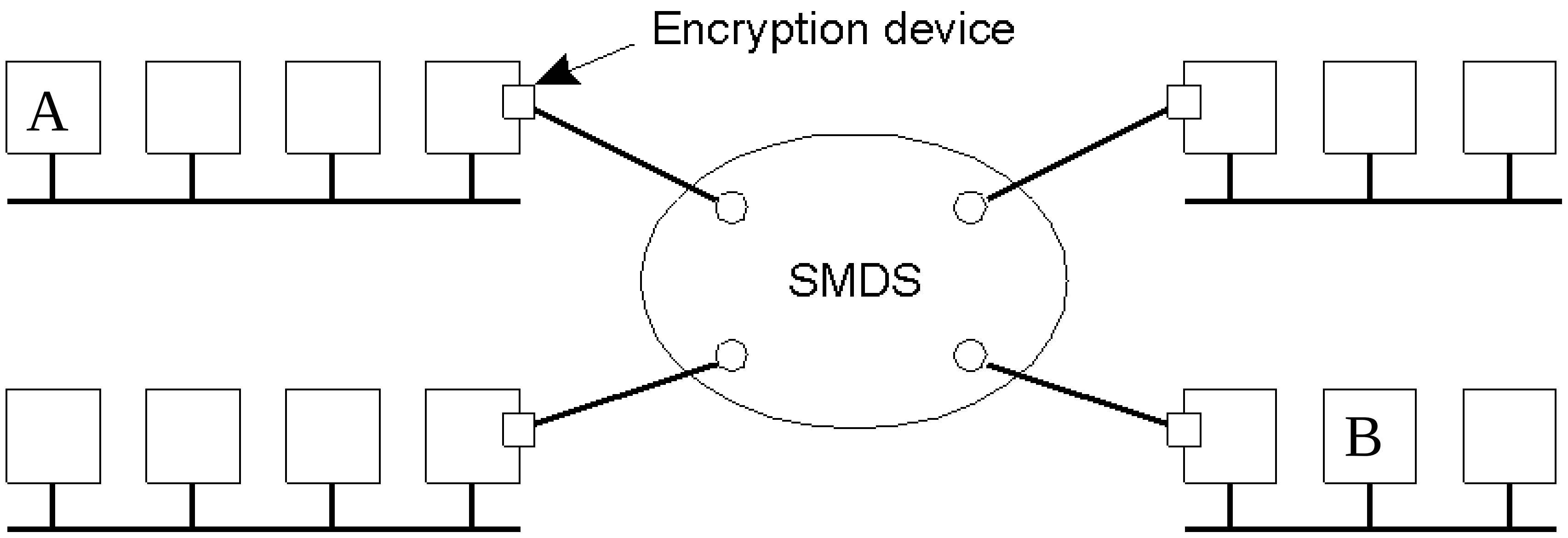
A decision is required as to where in the model security the mechanism is to be placed.

Trust and Security

- A system is either secure or it is not.
- Whether a client considers a systems to be secure is a matter of trust.

In which layer security mechanisms are placed depends on the *trust* a client has in how secure the services are in any particular layer.

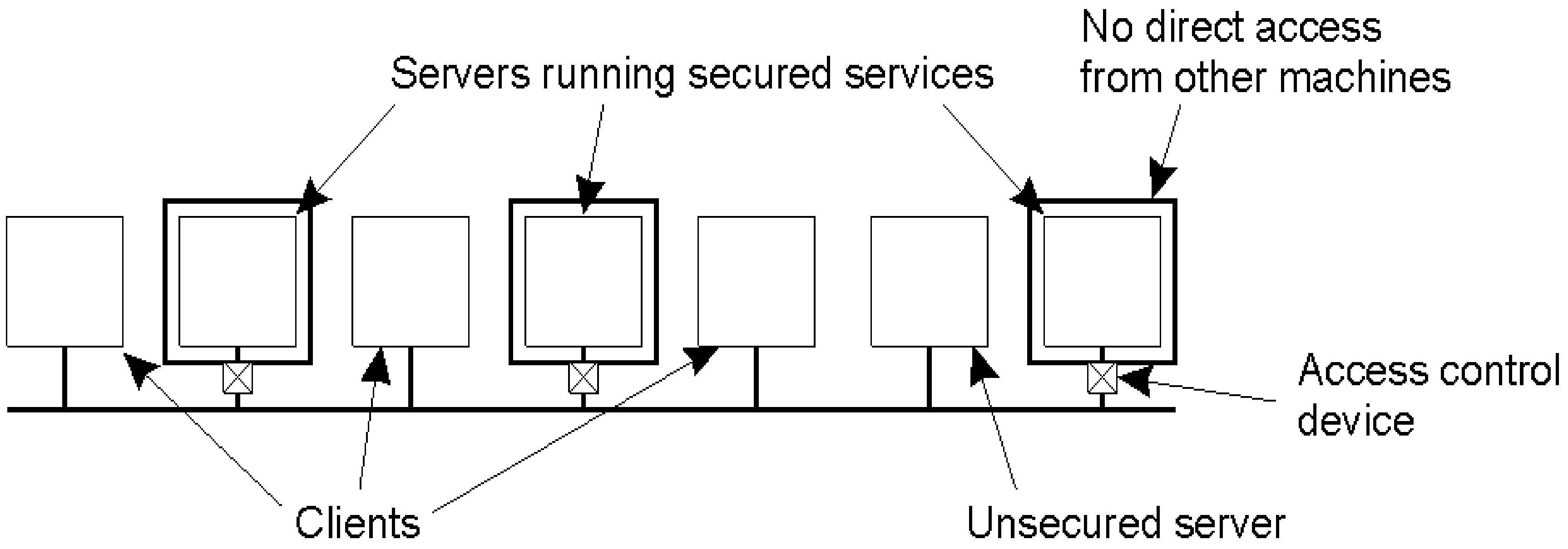
More on Layering of Security Mechanisms



Several sites connected through a wide-area backbone service.

If the encryption mechanisms cannot be trusted, additional mechanisms can be employed by clients to provide the level of trust required (e.g., using SSL at the Application Layer).

Distribution of Security Mechanisms


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

The *Trusted Computing Base (TCB)* is the set of services/mechanisms within a distributed system required to support a security policy.

Design Issue: Simplicity

Designing a secure computer system is considered a difficult task, regardless of whether it is also a DS.

The use of a few simple mechanisms that are easily understood and trusted to work would be the ideal situation.

Unfortunately, the real world is not this clear cut, as introducing security mechanisms to an already complex system can often matters worse.

However, this is still a design goal to aim for!

Security Mechanisms

Fundamental technique
within any distributed systems
security environment:

Cryptography.

Types of Cryptosystems

Symmetric: often referred to as *conventional cryptography*, defined as:

$$P = D_k(E_k(P))$$

Asymmetric: often referred to as *public-key cryptography*, defined as:

$$P = D_{k_d}(E_{k_e}(P))$$

Hash (One-Way) Functions

$h = H(m)$

Given *H* and *m*, *h* is easy to compute.

However, given *H* and *h*, it is computationally infeasible to compute *m*. That is, the function only works **One-Way**.

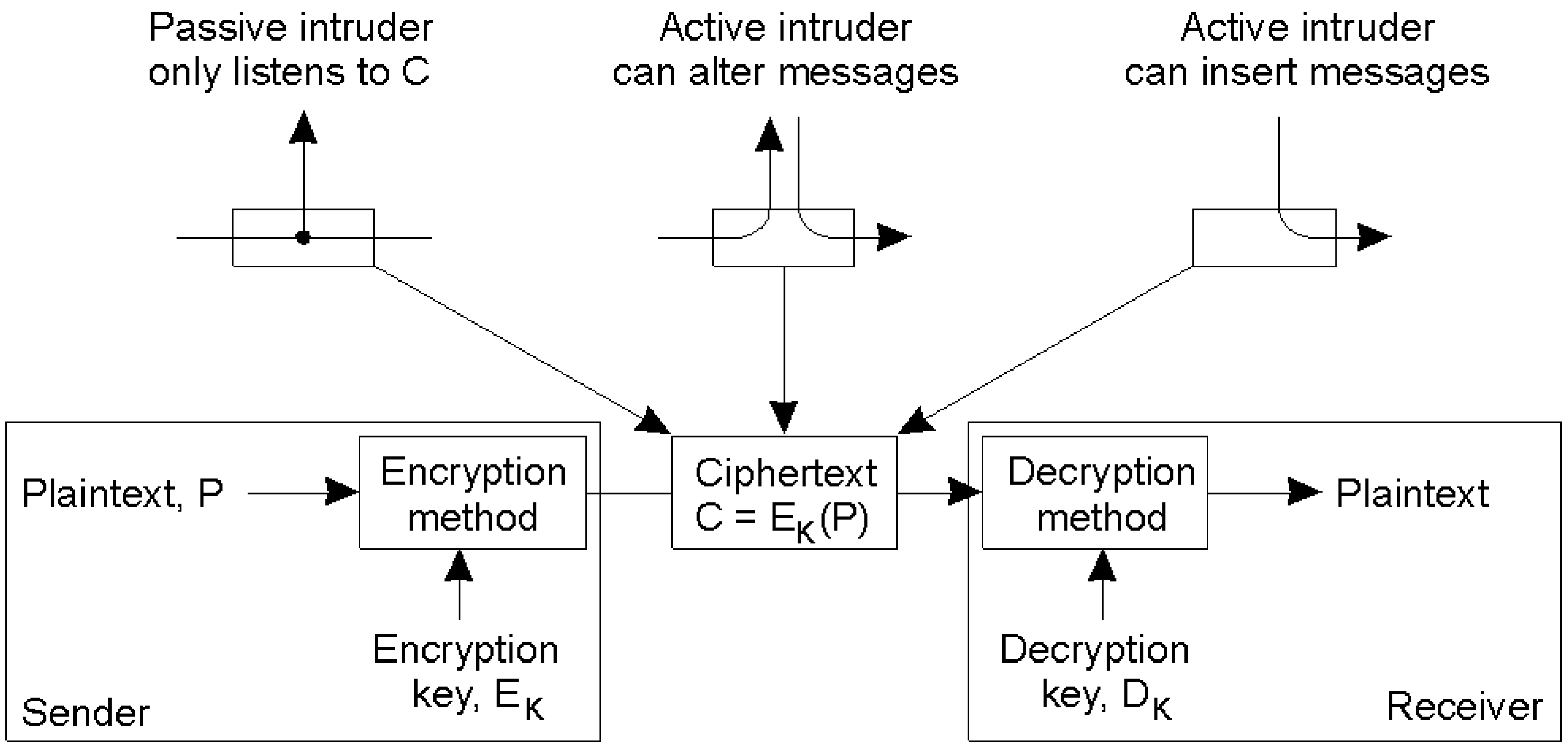
Weak Collision Resistence: given *m* and $h = H(m)$, it is hard to find another *m*, for instance, *m2*, such that *m* and *m2* produce the same *h*.

Strong Collision Resistence: Given *H*, it is infeasible to find an *m* and an *m2* such that $H(m)$ and $H(m2)$ produce the same value.

Notation for Cryptography

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

Participants/Components



Intruders and eavesdroppers in communication.

Introducing Alice, Bob & Co.

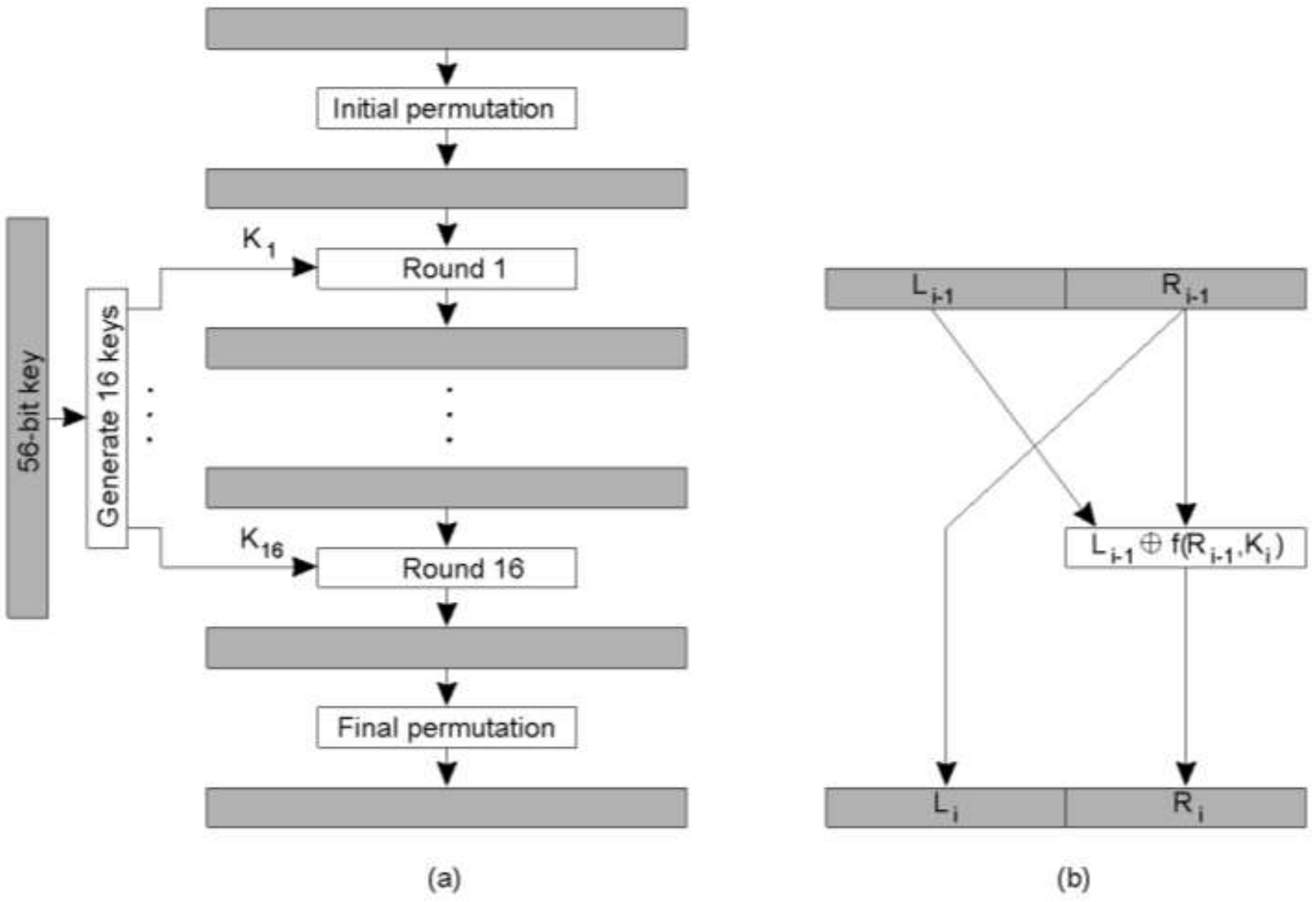
Alice and Bob are the *good guys*.

Chuck and Eve are usually the *bad guys*.

For Alice to send a confidential message to Bob, she uses Bob's *public key*. Only Bob can decrypt this message (with his *private key*).

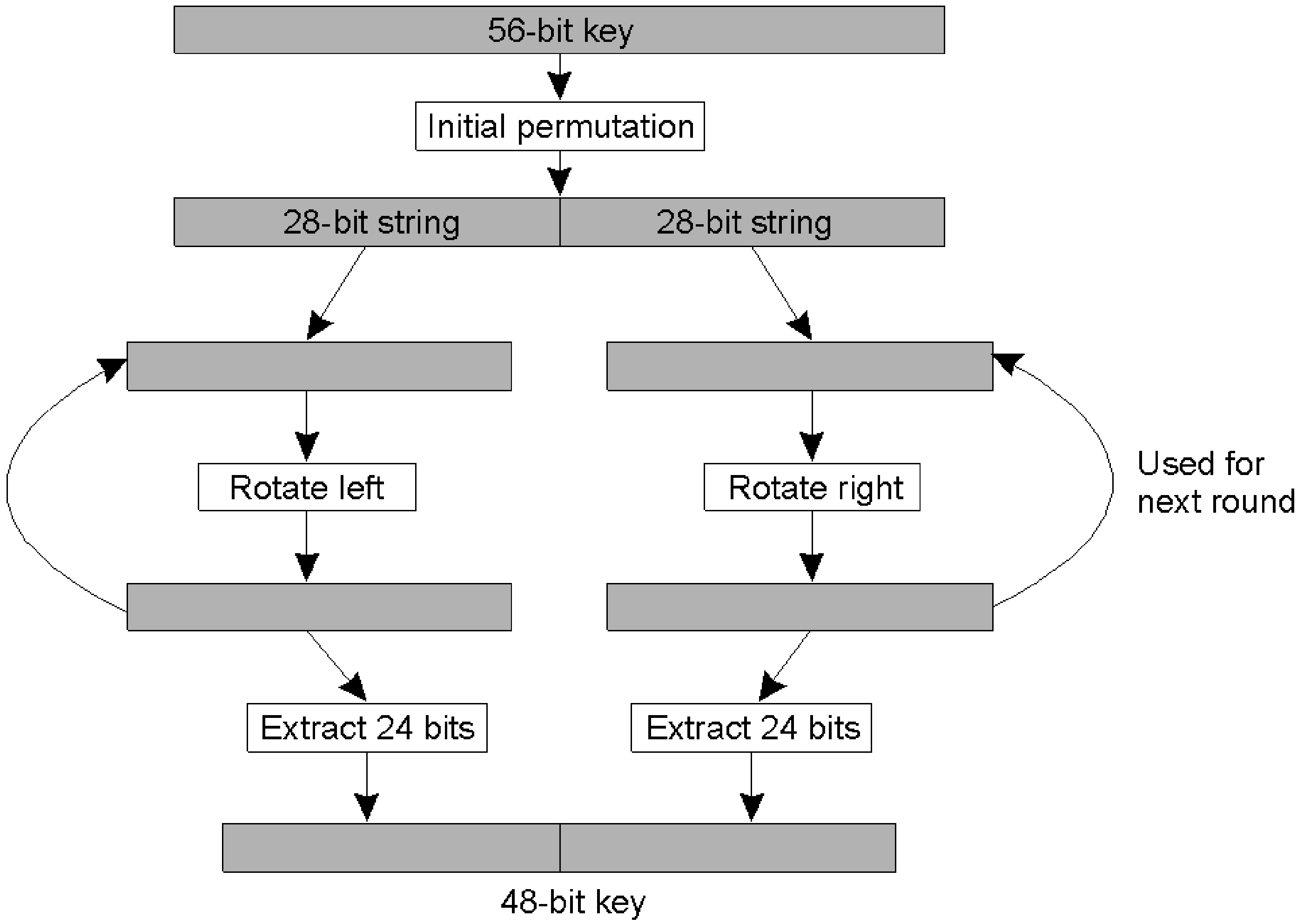
However, Bob also wants to be sure the message is actually from Alice, so Alice uses her *private key* to **sign** the message, and Bob uses Alice's *public key* to decrypt it. If a correctly formatted message appears, Bob *knows* Alice sent it.

Symmetric Cryptosystem: DES – 1 of 2



a) The principle of DES.
b) Outline of one encryption round.

Symmetric Cryptosystem: DES – 2 of 2



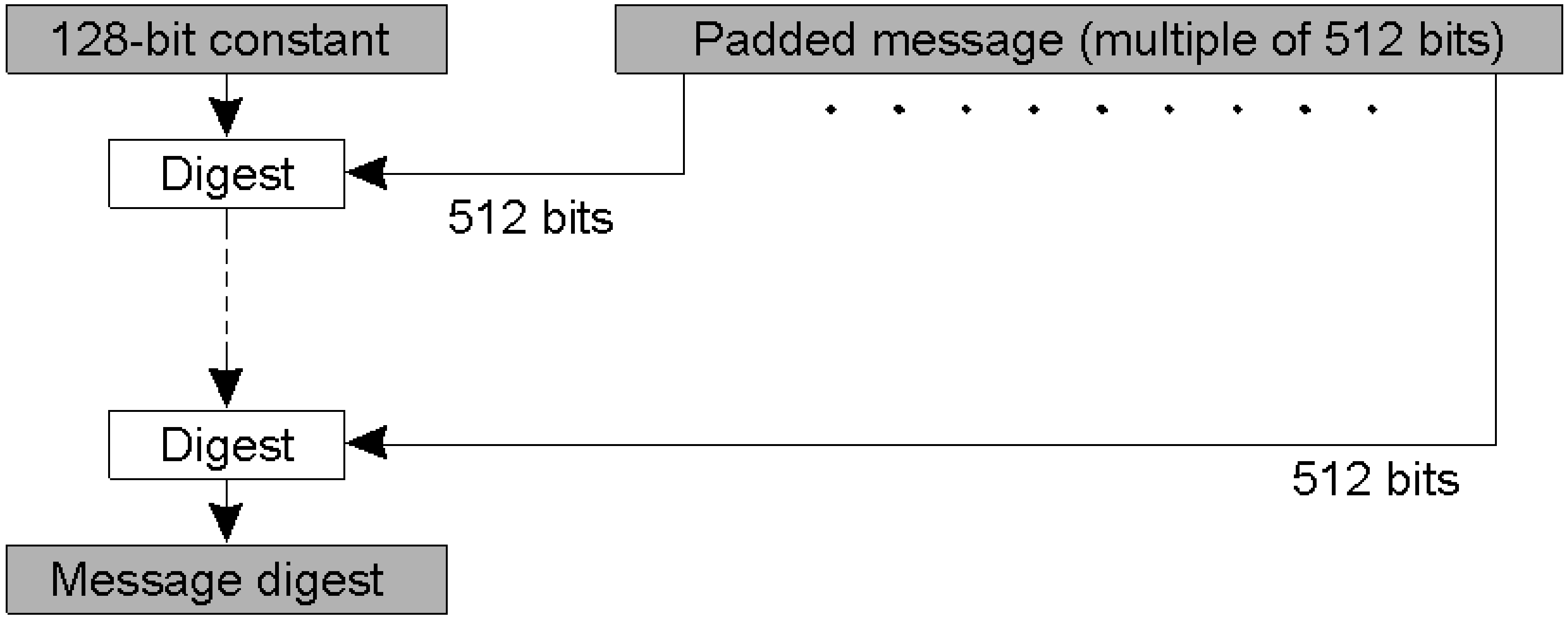
Details of per-round key generation in DES.

Public-Key Cryptosystem: RSA

Generating the private and public key requires four steps:

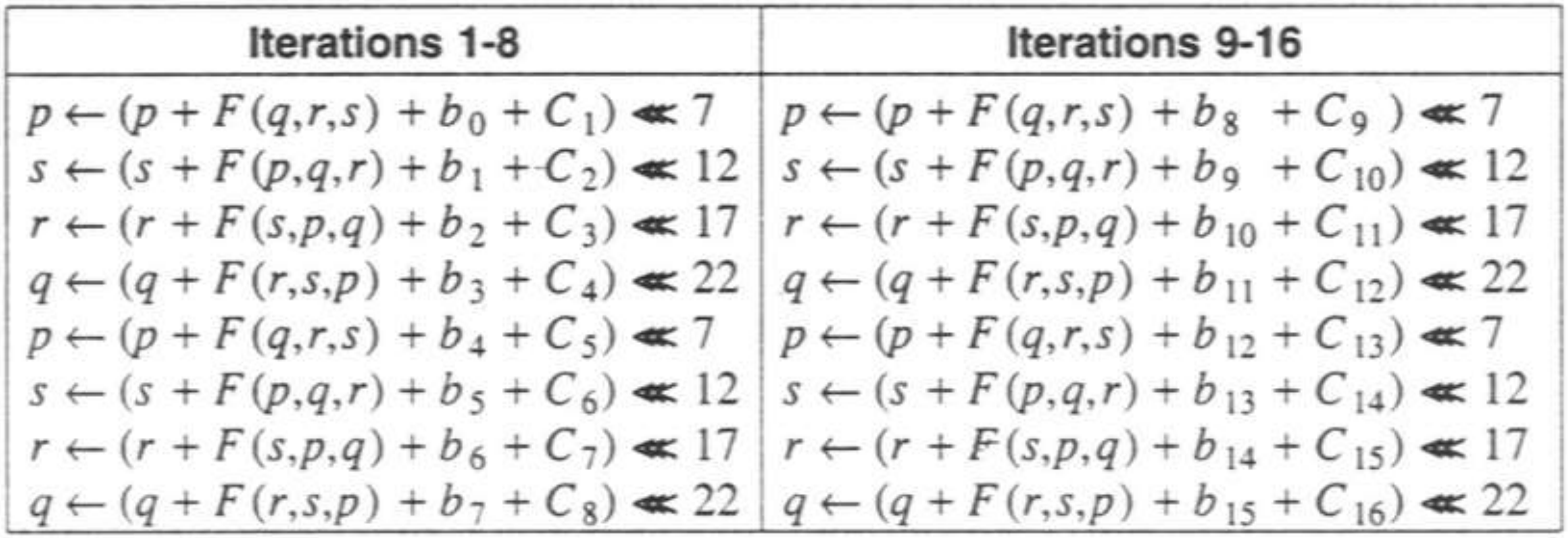
1. Choose two very large prime numbers, *p* and *q*.
2. Compute $n = p \times q$ and $z = (p - 1) \times (q - 1)$.
3. Choose a number *d* that is relatively prime to *z*.
4. Compute the number *e* such that $e \times d = 1 \bmod z$.

Hash Function: MD5 (1 of 2)



The structure of MD5 (a related technique is that employed by one-way functions).

Hash Function: MD5 (2 of 2)



Iterations 1-8	Iterations 9-16
$p \leftarrow (p + F(q,r,s) + b_0 + C_1) \lll 7$	$p \leftarrow (p + F(q,r,s) + b_8 + C_9) \lll 7$
$s \leftarrow (s + F(p,q,r) + b_1 + C_2) \lll 12$	$s \leftarrow (s + F(p,q,r) + b_9 + C_{10}) \lll 12$
$r \leftarrow (r + F(s,p,q) + b_2 + C_3) \lll 17$	$r \leftarrow (r + F(s,p,q) + b_{10} + C_{11}) \lll 17$
$q \leftarrow (q + F(r,s,p) + b_3 + C_4) \lll 22$	$q \leftarrow (q + F(r,s,p) + b_{11} + C_{12}) \lll 22$
$p \leftarrow (p + F(q,r,s) + b_4 + C_5) \lll 7$	$p \leftarrow (p + F(q,r,s) + b_{12} + C_{13}) \lll 7$
$s \leftarrow (s + F(p,q,r) + b_5 + C_6) \lll 12$	$s \leftarrow (s + F(p,q,r) + b_{13} + C_{14}) \lll 12$
$r \leftarrow (r + F(s,p,q) + b_6 + C_7) \lll 17$	$r \leftarrow (r + F(s,p,q) + b_{14} + C_{15}) \lll 17$
$q \leftarrow (q + F(r,s,p) + b_7 + C_8) \lll 22$	$q \leftarrow (q + F(r,s,p) + b_{15} + C_{16}) \lll 22$

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

DS Security: Two Major Issues

1. Secure communications between parties.
2. Authorisation.

Secure channels protect against (protected by):

Interception (confidentiality).

Modification (auth. and integrity).

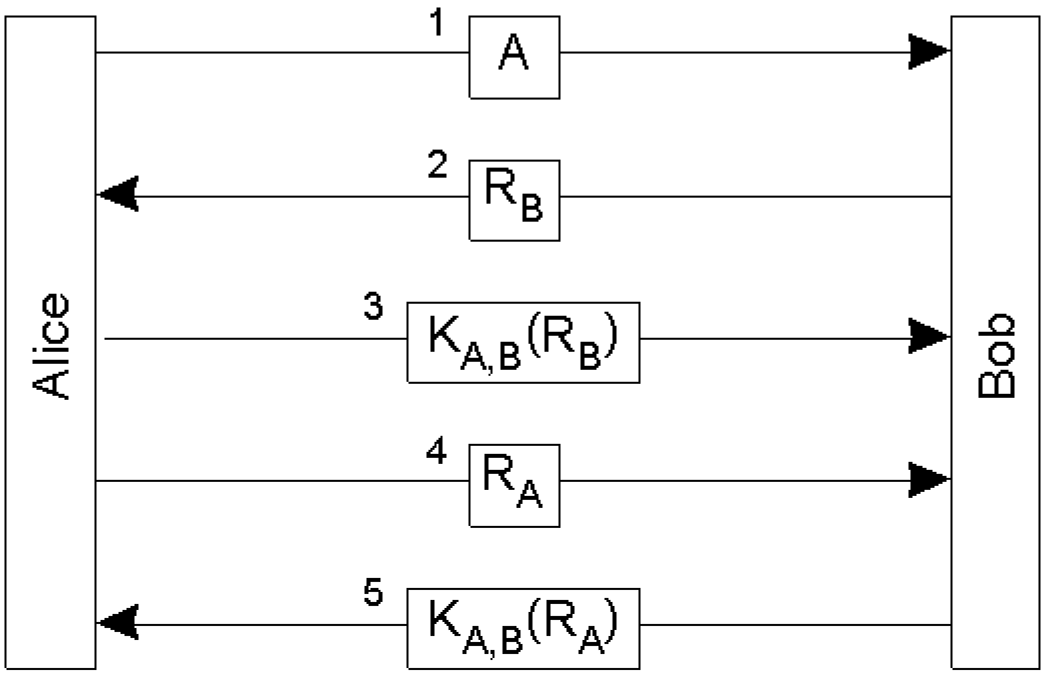
Fabrication (auth. and integrity).

Note that *authentication* and *message integrity* as technologies rely on each other.

Applications of Cryptography

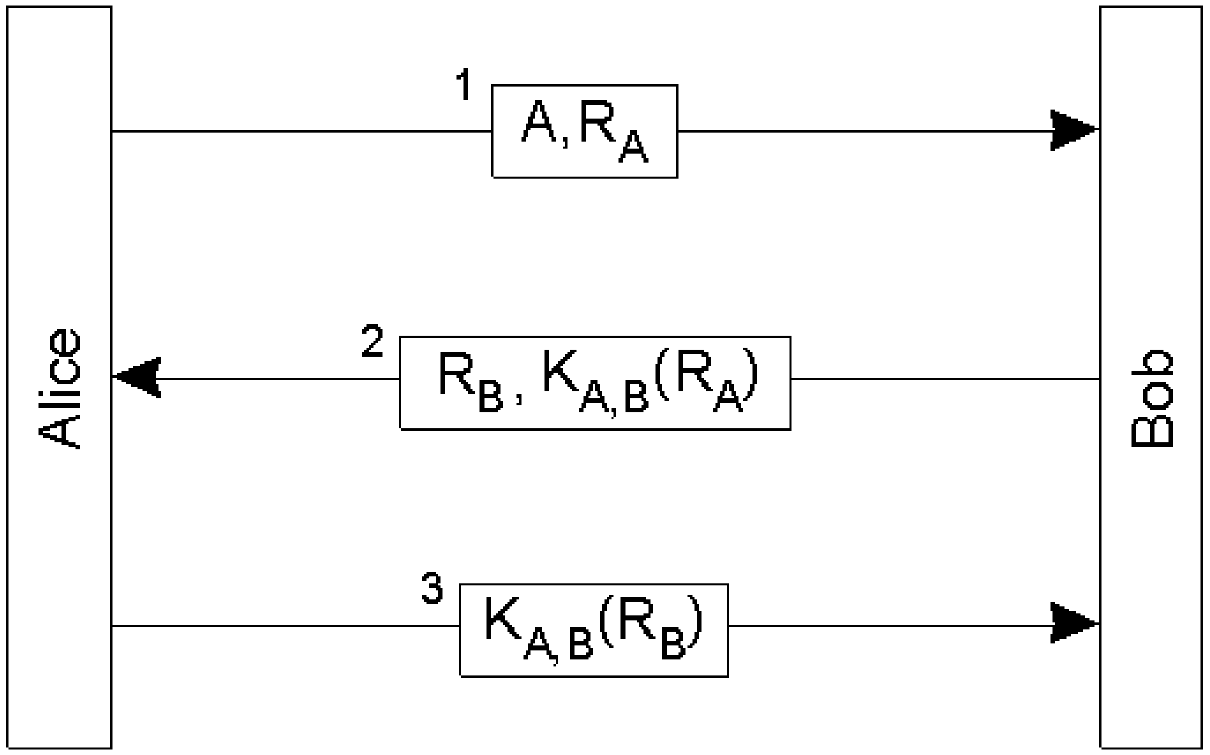
1. Authentication.
2. Message Integrity.
3. Confidentiality.

Authentication



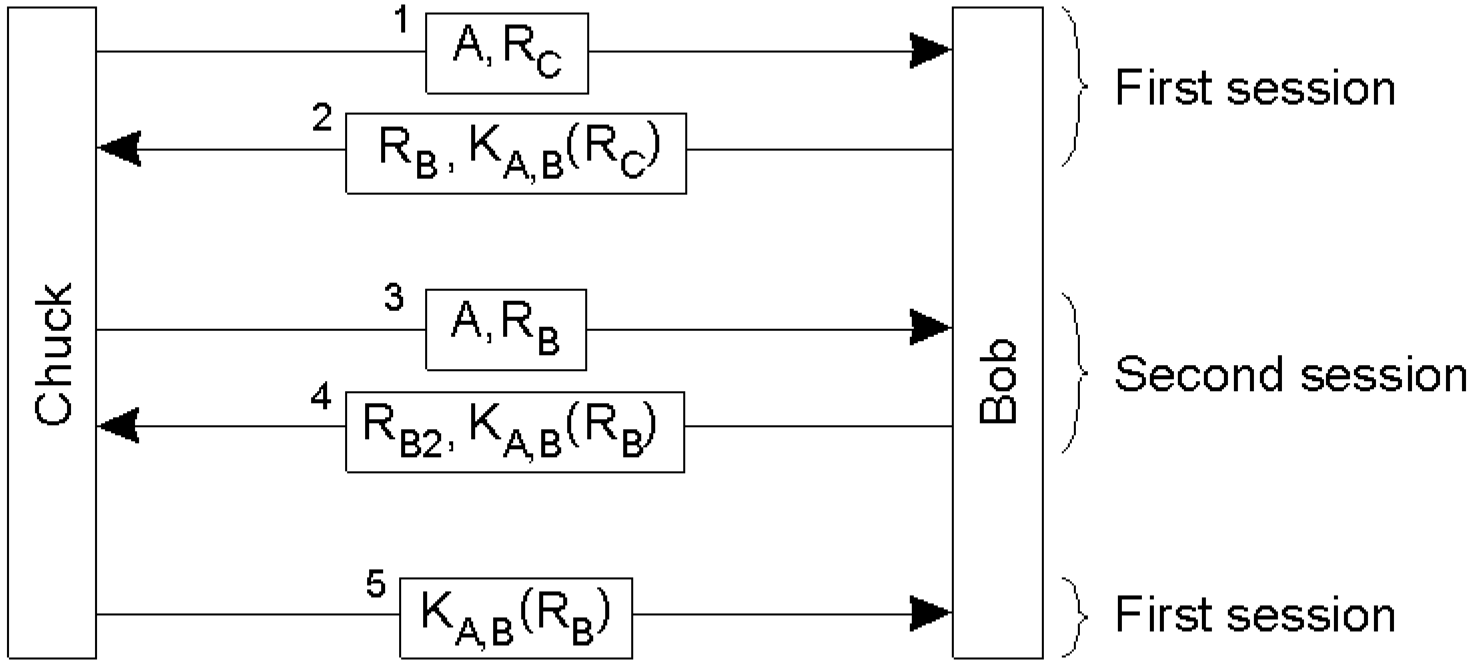
Authentication based on a *shared secret key*, using a 'challenge response' protocol. Note: *R* is a random number.

An Authentication Optimization??



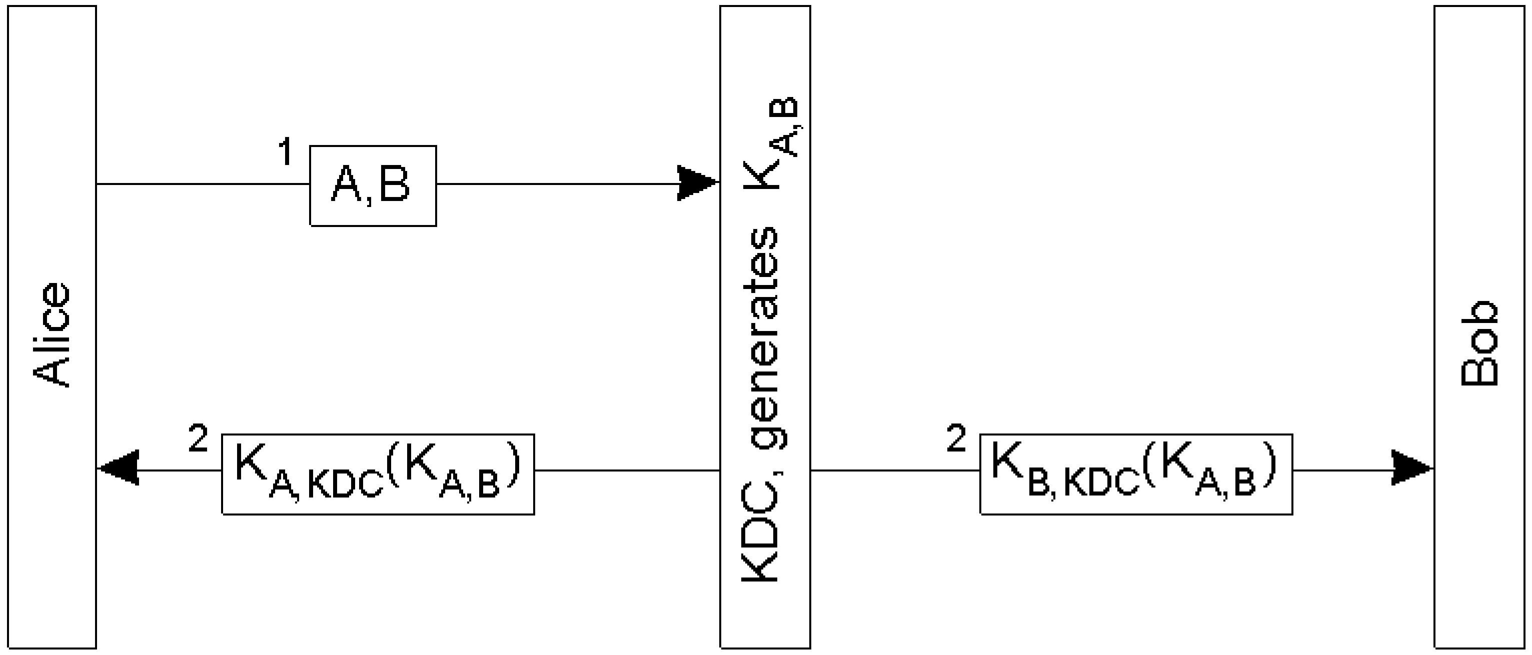
Authentication based on a shared secret key, but using three instead of five messages. *Is this better??*

An Authentication Attack!!



The ‘reflection attack’. Chuck wants Bob to think he is Alice, so he starts up a second session to *trick* Bob.

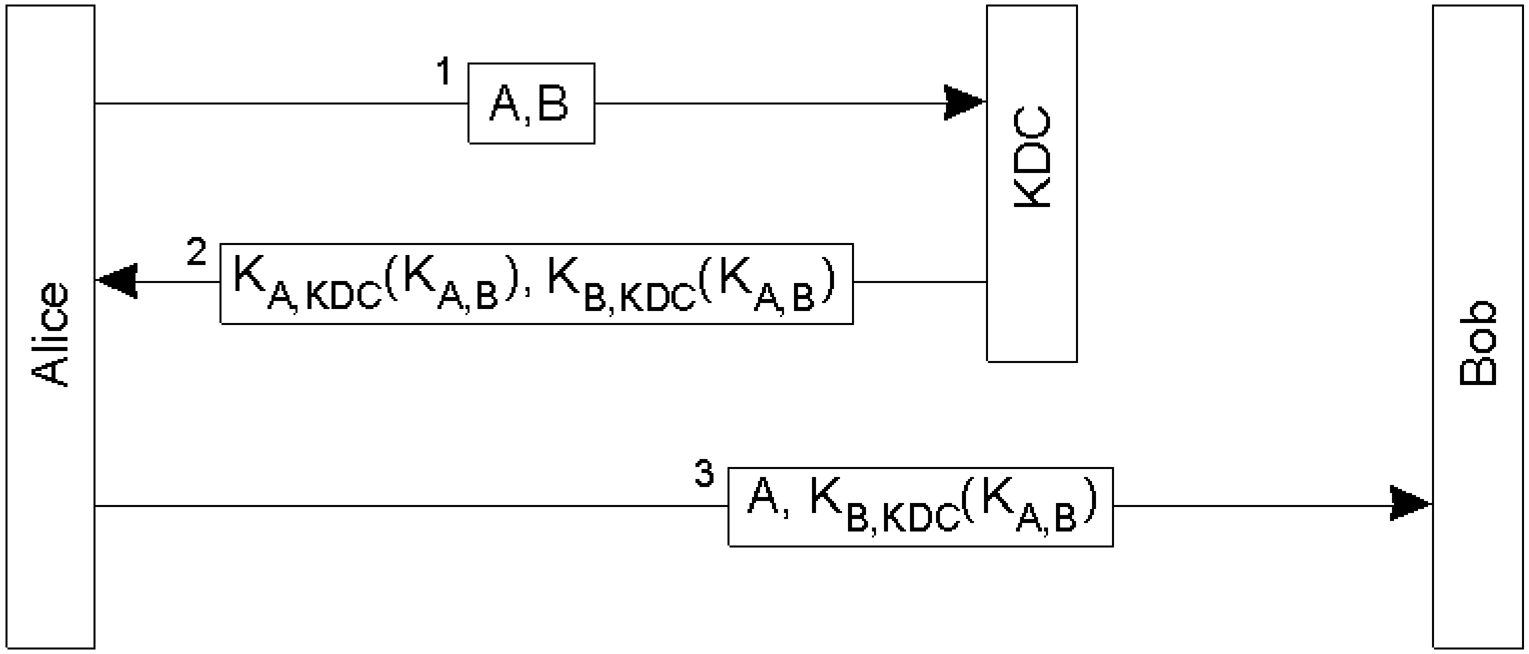
Authentication Using a Key Distribution Center



The principle of using a KDC.

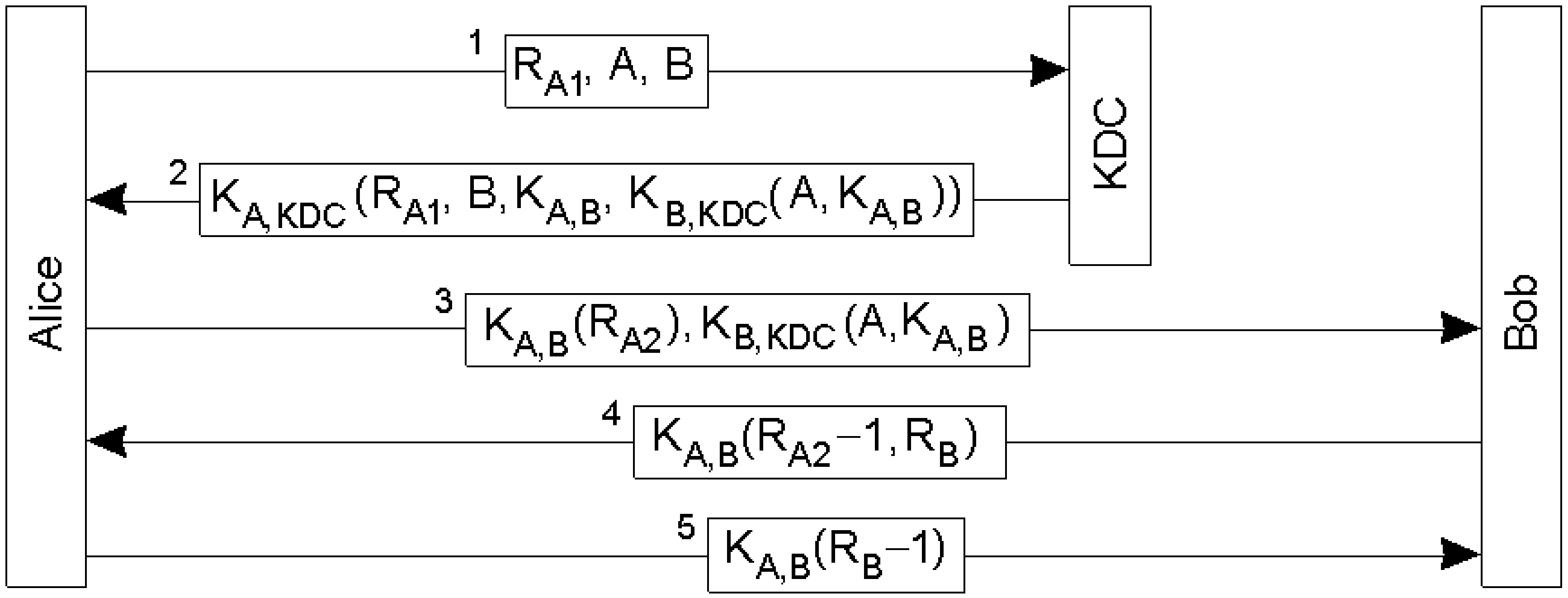
This scales better than the alternative: N keys in the system as opposed to N(N-1)/2 keys!

A Ticket-Granting Key Distribution Center



Using a *ticket* and letting Alice set up her own connection to Bob, which is an accepted improvement to the previous protocol.

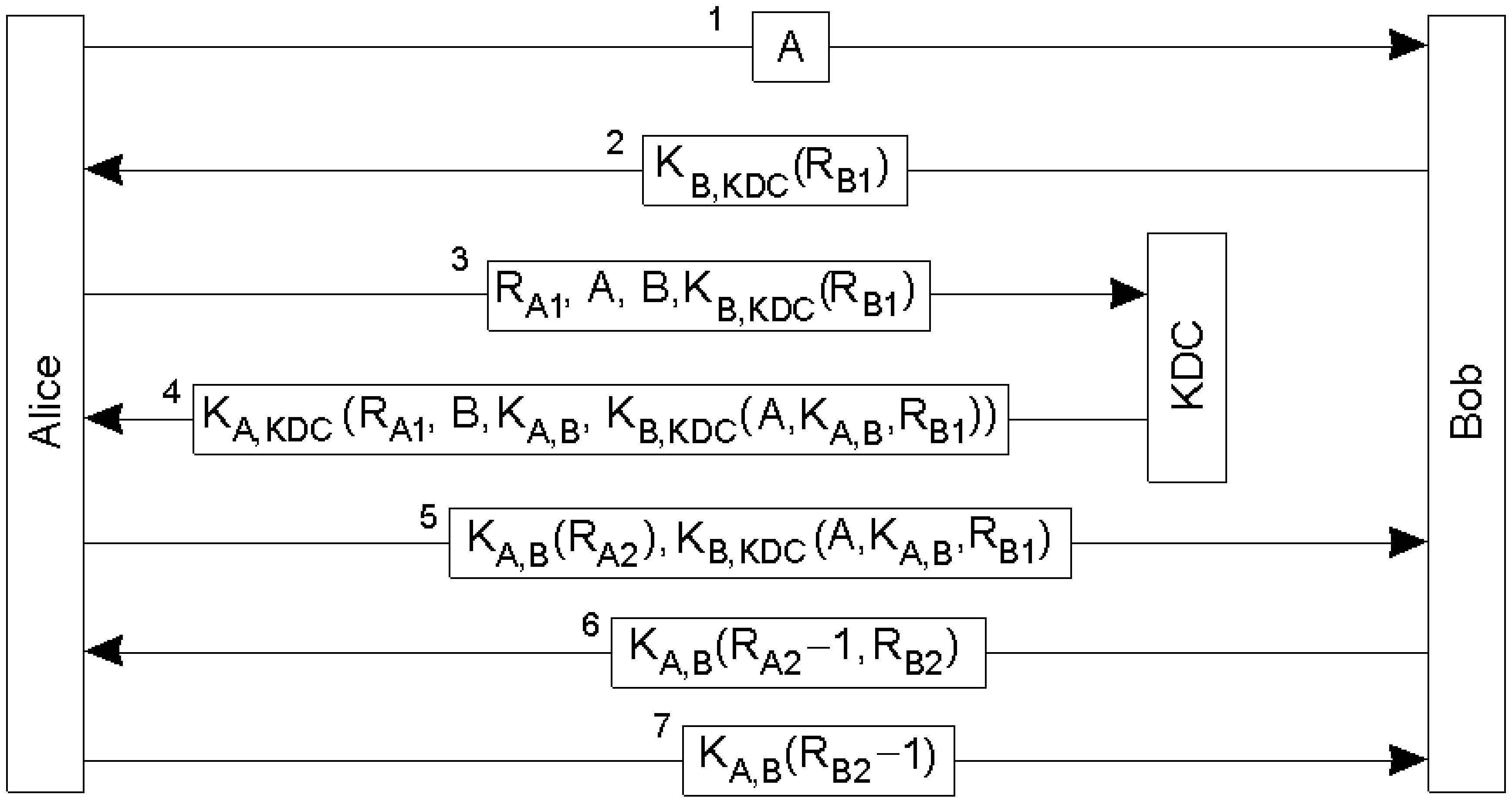
The ‘Multi-way Challenge Response’ Protocol


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.

Note: R_{A1} is a “nonce” – a random number that is used only once. In this protocol, it serves to link pairs of messages together and guards against ‘replay attacks’.

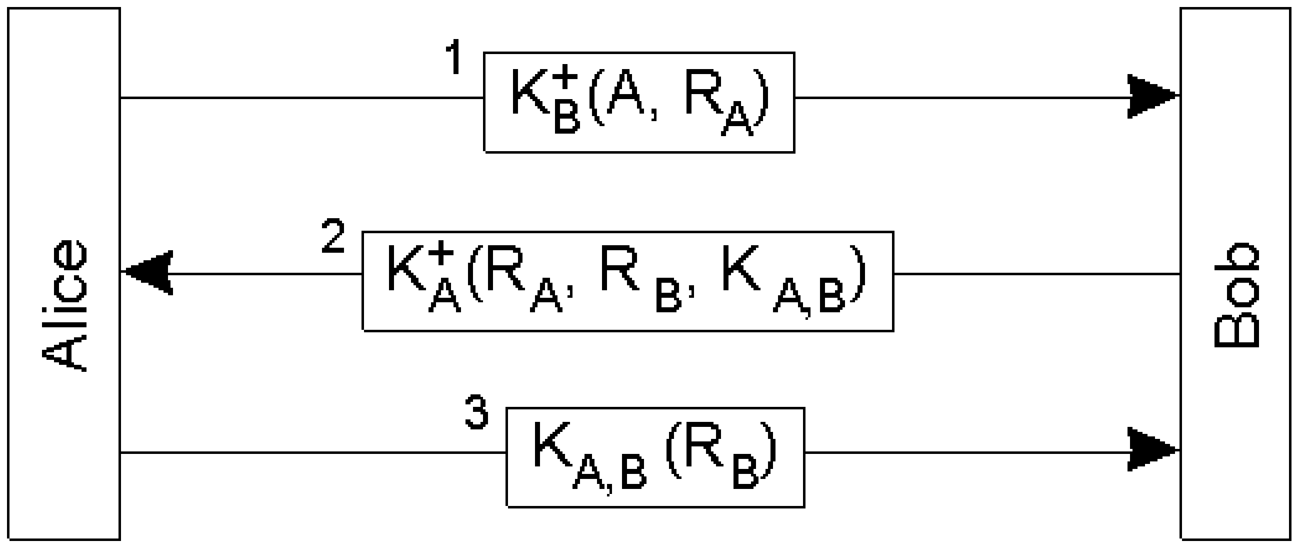
Avoiding Session Key Reuse



Alice
Bob
KDC
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.

Note that the KDC is missing … but, this assumes that some mechanism exists to *verify* everyone's public key.

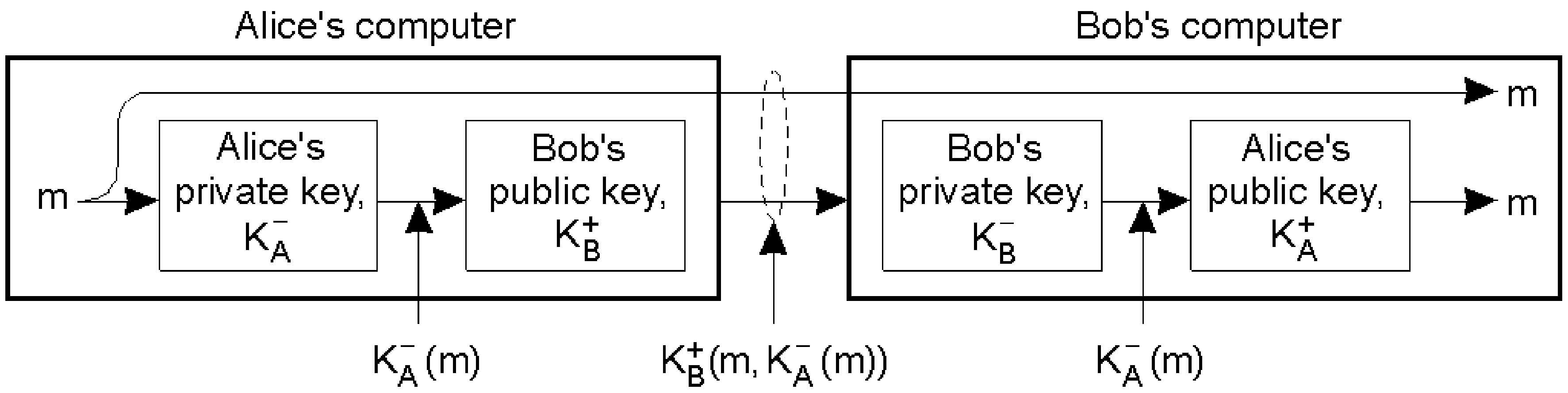
More on Secure Channels

In addition to *authentication*, a secure channel also requires that messages are *confidential*, and that they maintain their *integrity*.

For example: Alice needs to be sure that Bob cannot change a received message and claim it came from her. And Bob needs to be sure that he can prove the message was sent by/from Alice, just in case she decides to deny ever having sent it in the first place.

Solution: Digital Signing.

Digital Signatures

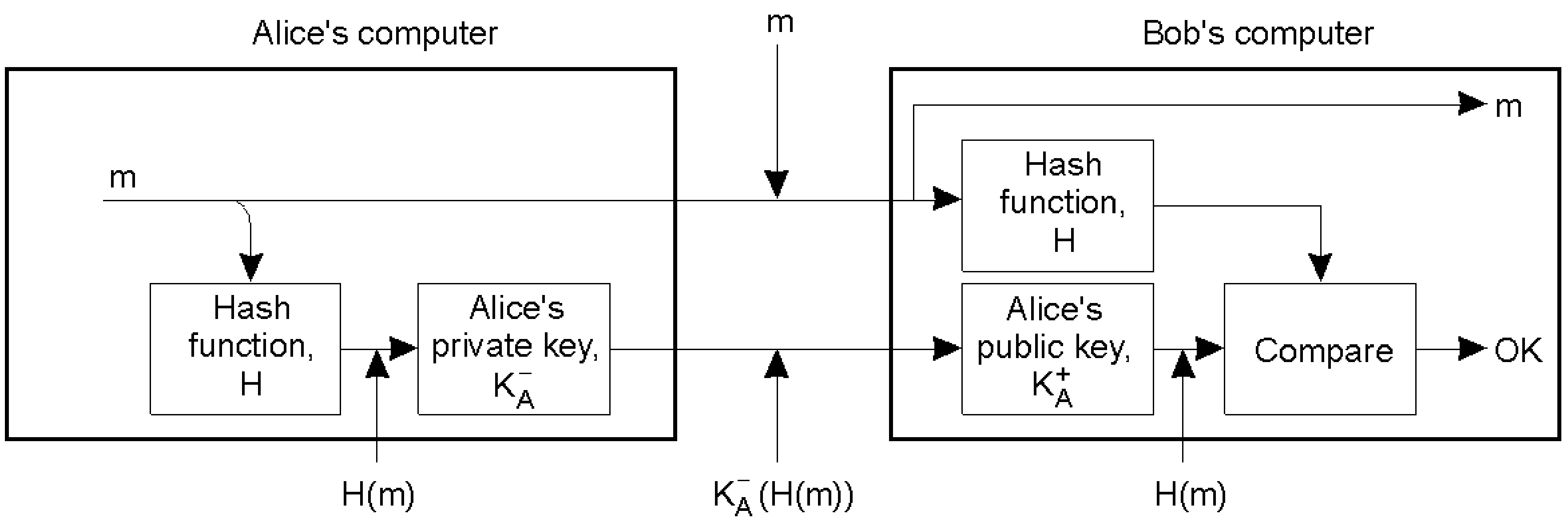

Alice's computer
Bob's computer
m
Alice's private key, K_A^-
Bob's public key, K_B^+
Bob's private key, K_B^-
Alice's public key, K_A^+
m
m
$K_A^-(m)$
$K_B^+(m, K_A^-(m))$
$K_A^-(m)$

Digital signing a message using public-key cryptography.

This is implemented in the RSA technology.

Note: the entire document is encrypted/signed - this can sometimes be a costly overkill.

Digital Signature Digests



Digitally signing a message using a message digest.

Message is sent as *plaintext*. However, the digest can be used to assure Bob of message integrity.

Secure Group Communications

Within Distributed Systems, it is often necessary to enable a secure channel between a group of processes (not just 2).

Think of replicated processes …

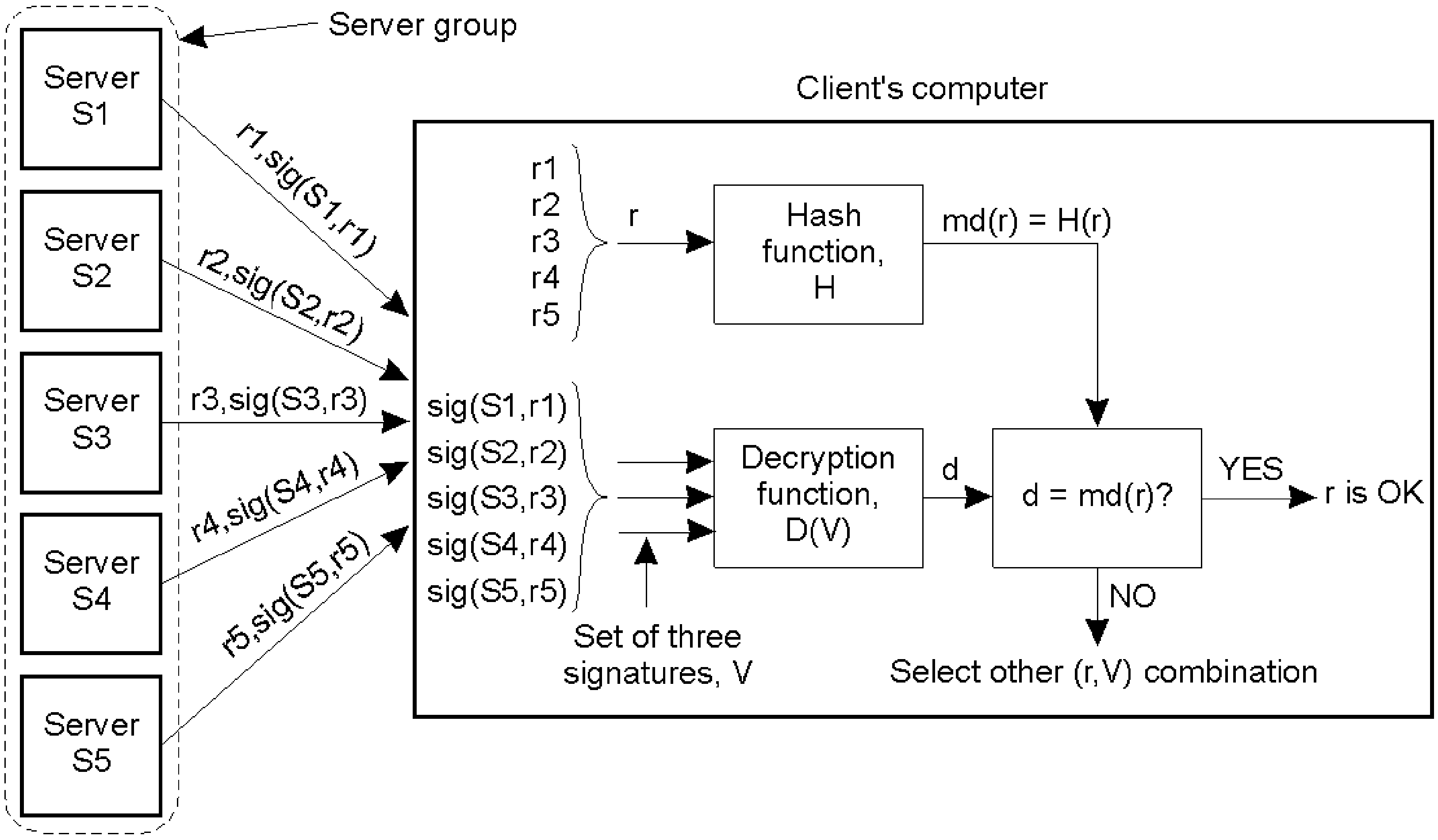
Solution 1: all replicas use same shared key!!!

Solution 2: each pair share a secret key – this results in N(N-1)/2 key pairs in the system!!

Solution 3: use public-key crytpography – which results in N key pairs in the system!

But … what happens if a replicated process is compromised?

Secure Replicated Services



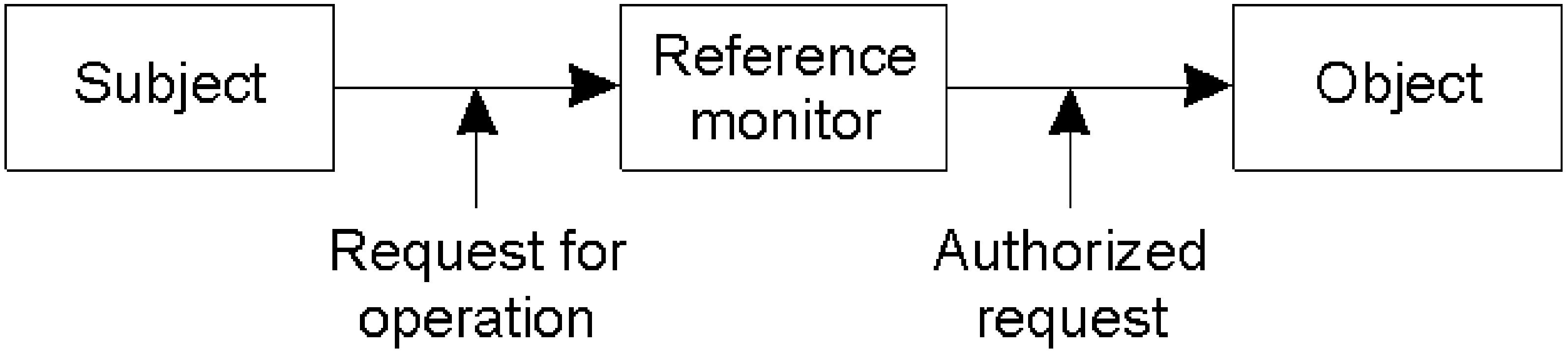
Sharing a secret signature in a group of replicated servers.

Access Control

Authorization is the process of
“granting access rights”
to a user/process.

Access Control is the process of
“verifying access rights”
for an authorized user/process.

General Model for Access Control



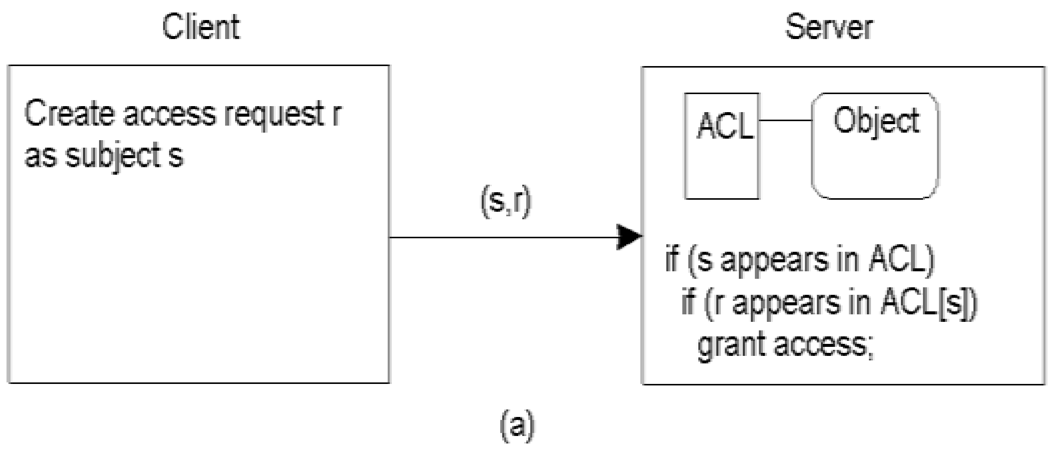
General model of controlling access to objects, via a tamper-proof *reference monitor*.

Access Control Matrix

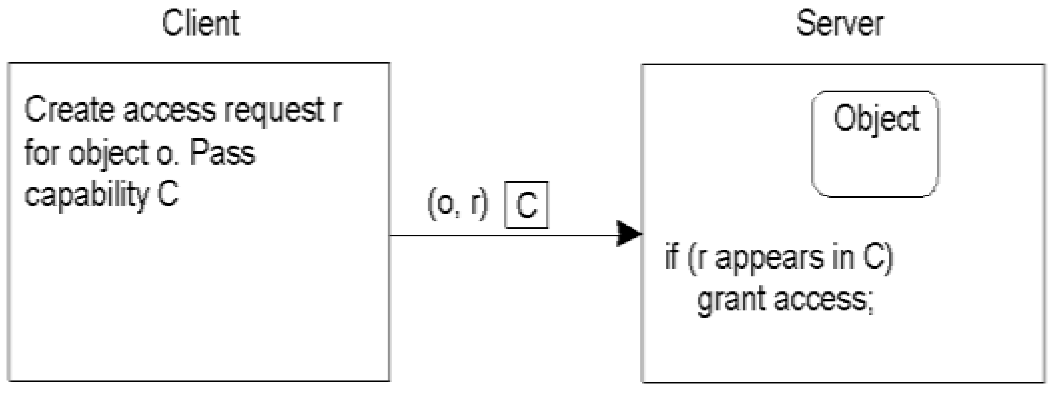
Comparison between ACLs and capabilities for protecting objects.

a) Using an ACL.

b) Using capabilities.

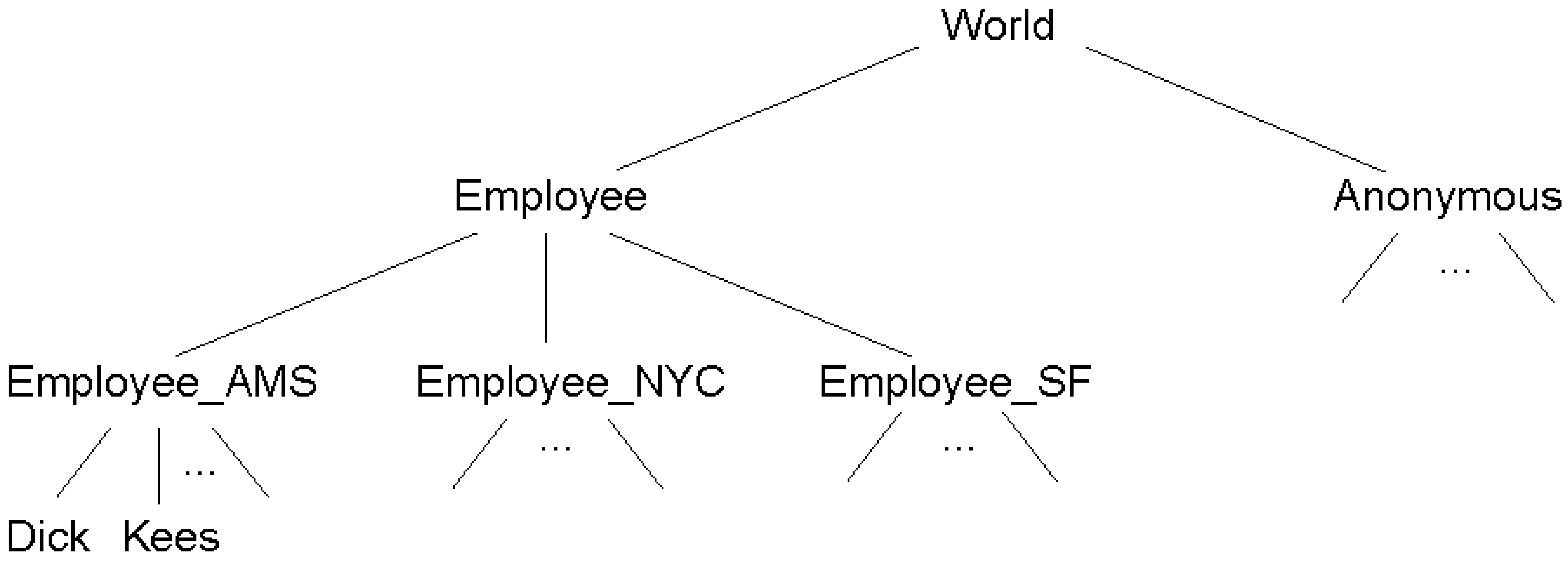


(a)



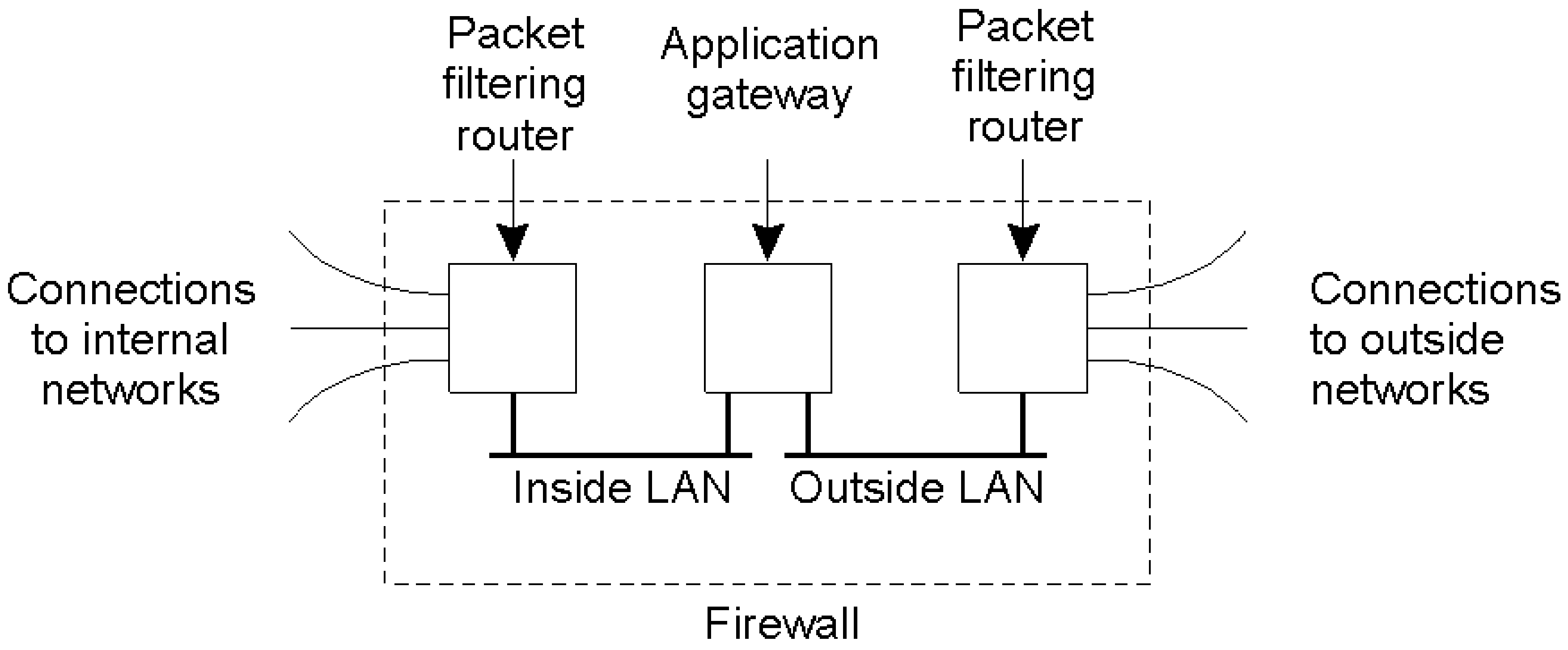
(b)

Protection Domains



The hierarchical organization of protection domains as groups of users.

Firewalls



A common implementation of a firewall, which come in one of three types: *packet-filtering*, *application-level* (*proxy*) or *circuit-level*.

Securing Mobile Code

Two main issues:

Securing against *malicious agents* that attempt to damage a mobile agent environment.

Securing a mobile agent from a *malicious environment* that attempts to interfere with the working of the mobile agent.

More on Securing Mobile Code

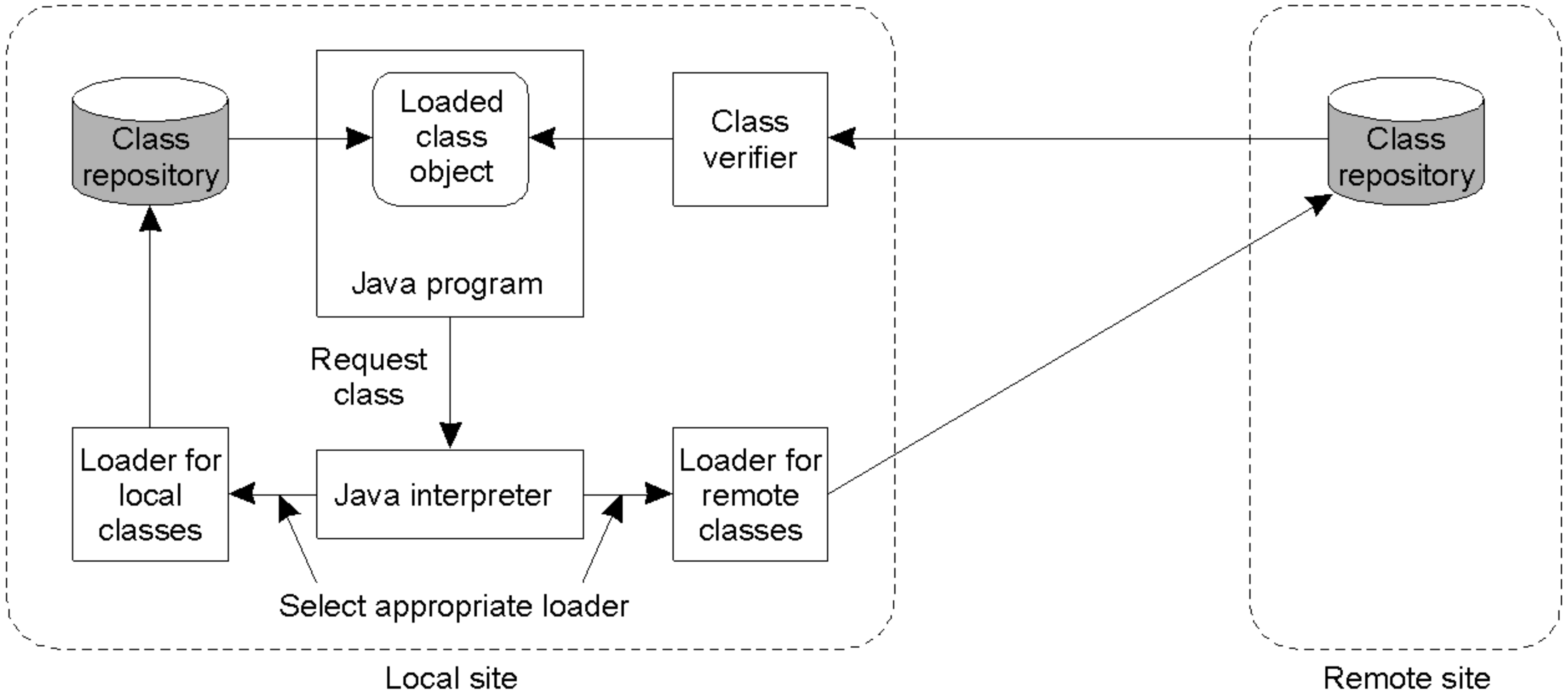
It has proved impossible to protect an agent against all types of attack.

The emphasis is on being able to detect modifications (or tampering) to an agent.

The *Ajanta System* uses public-key technologies to implement this idea:

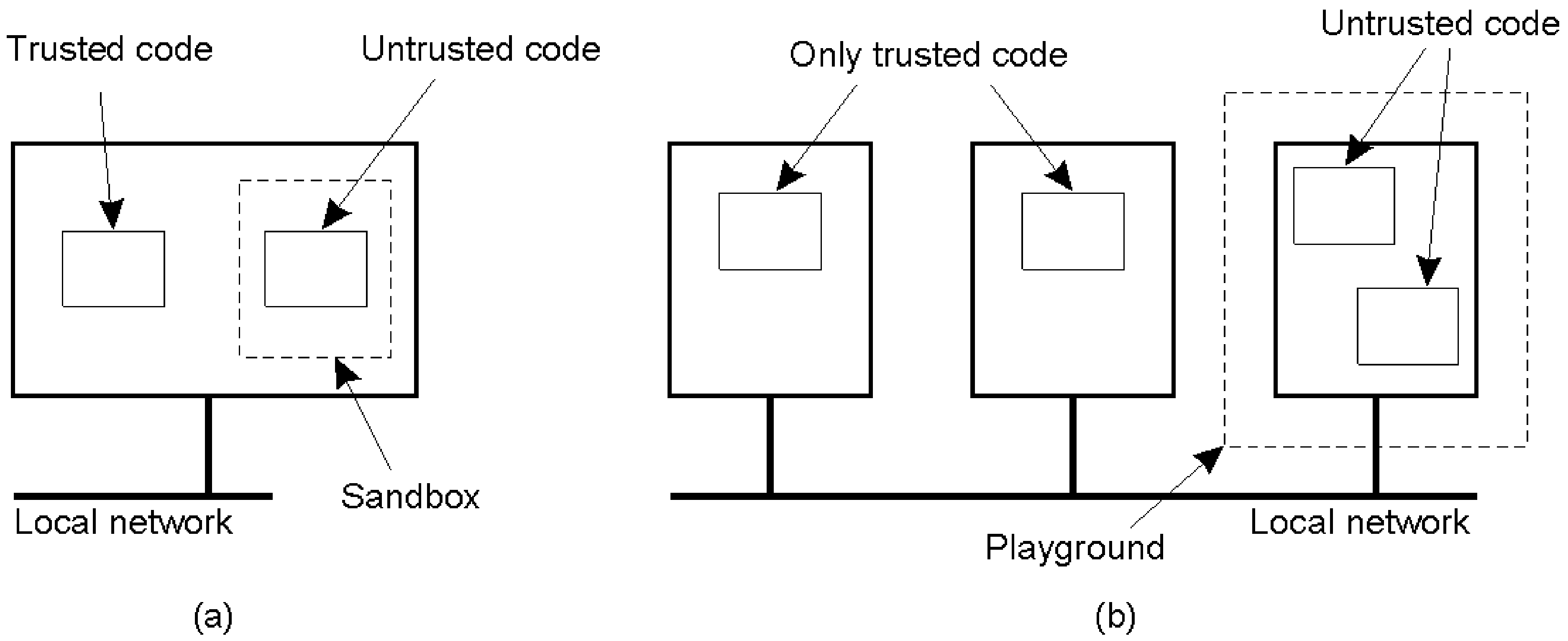
1. Read-only state.
2. Append-only logs.
3. Selective Revealing.

Protecting the Target (1)



The organization of a Java sandbox – each instruction is fully controlled by the receiving agent.

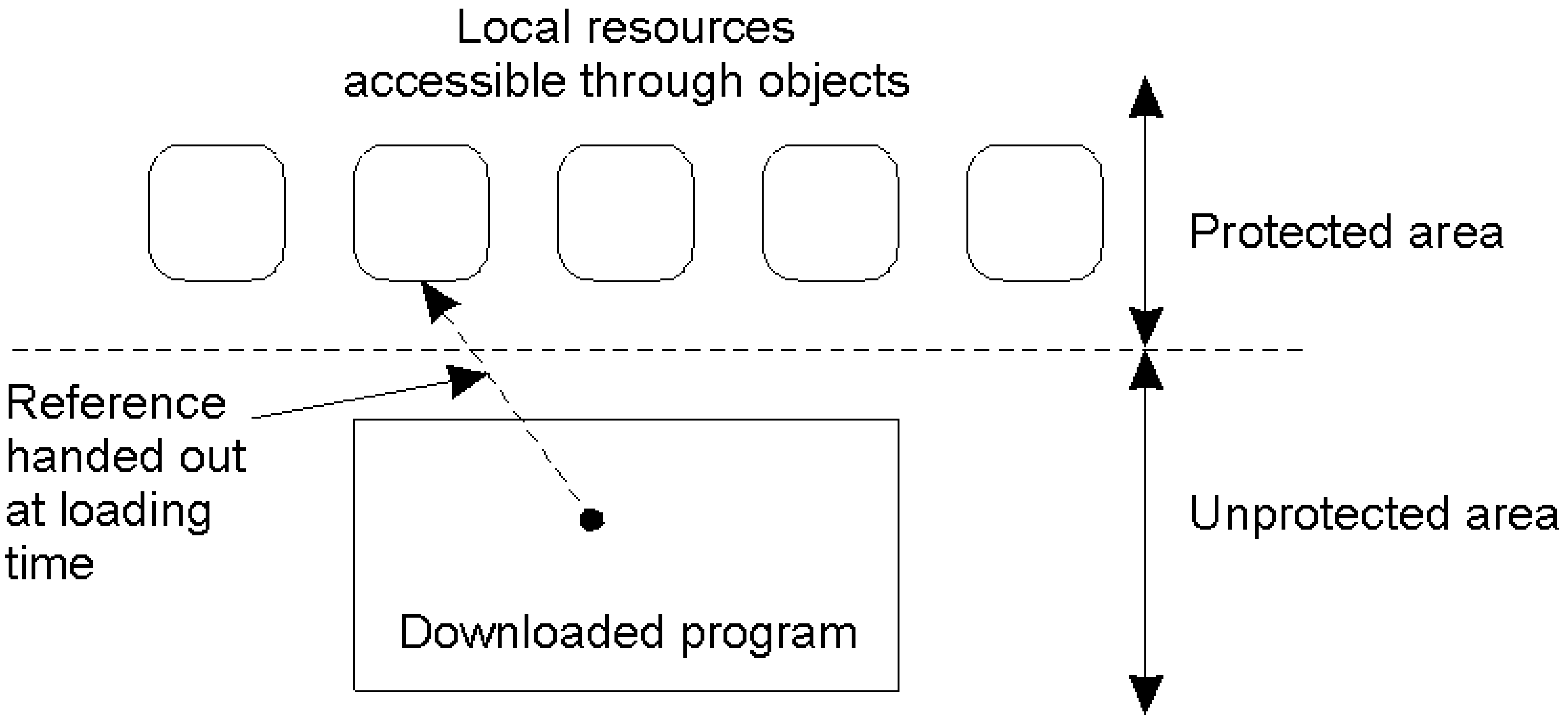
Protecting the Target (2)



a) A sandbox – part of the same computer.

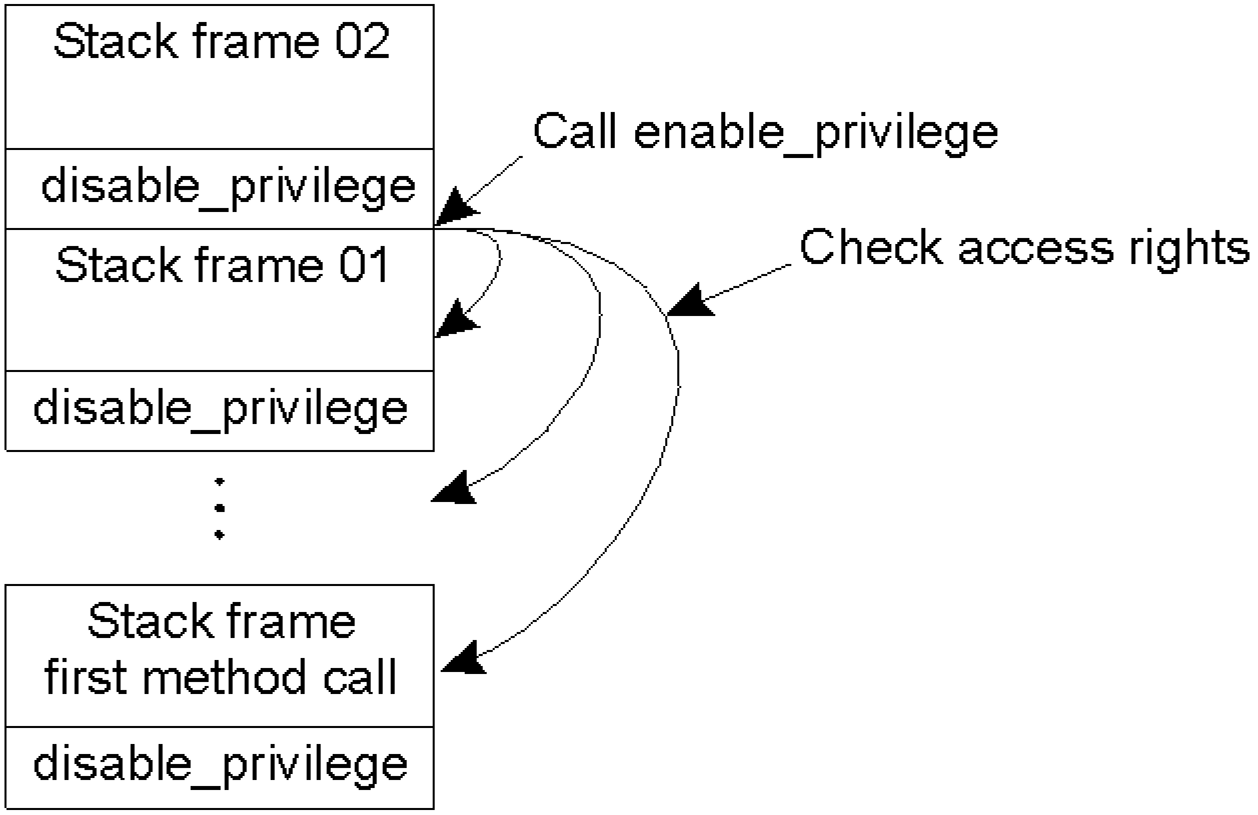
b) A playground – a separate computer that exists solely to securely execute the downloaded code.

Protecting the Target (3)


Local resources accessible through objects
Protected area
Reference handed out at loading time
Unprotected area
Downloaded program

The principle of using Java object references as capabilities.

Protecting the Target (4)



The principle of stack introspection.

Other techniques include: code signing and name space management.

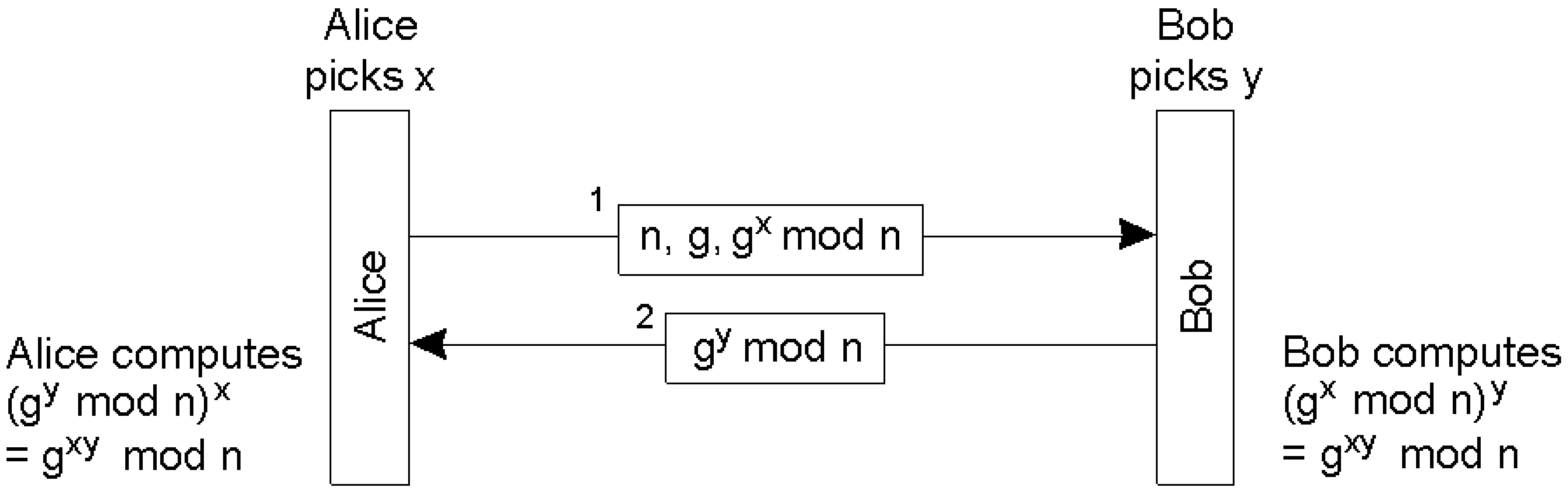
Security Management

Key issues:

How are (session) keys distributed securely?

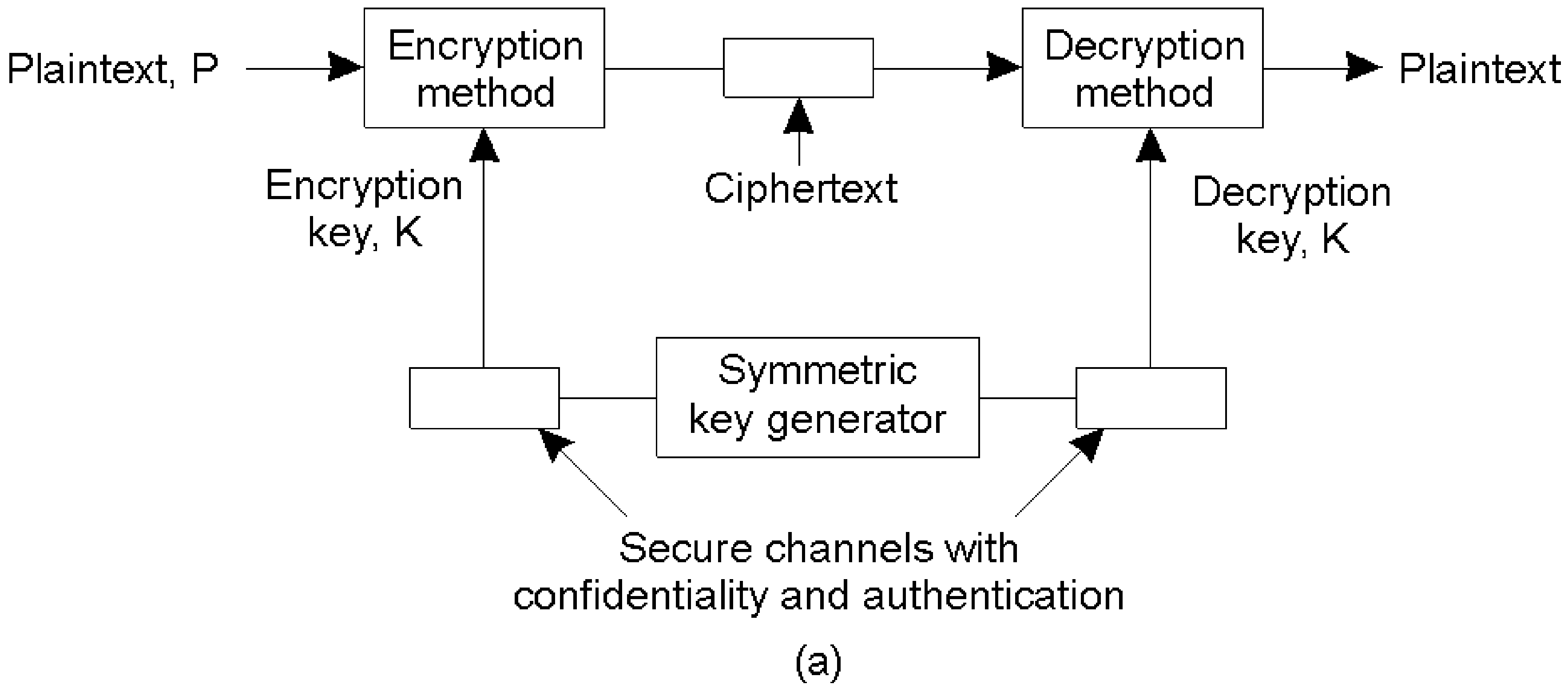
How can we be sure we are using the correct public-key?

Key Establishment



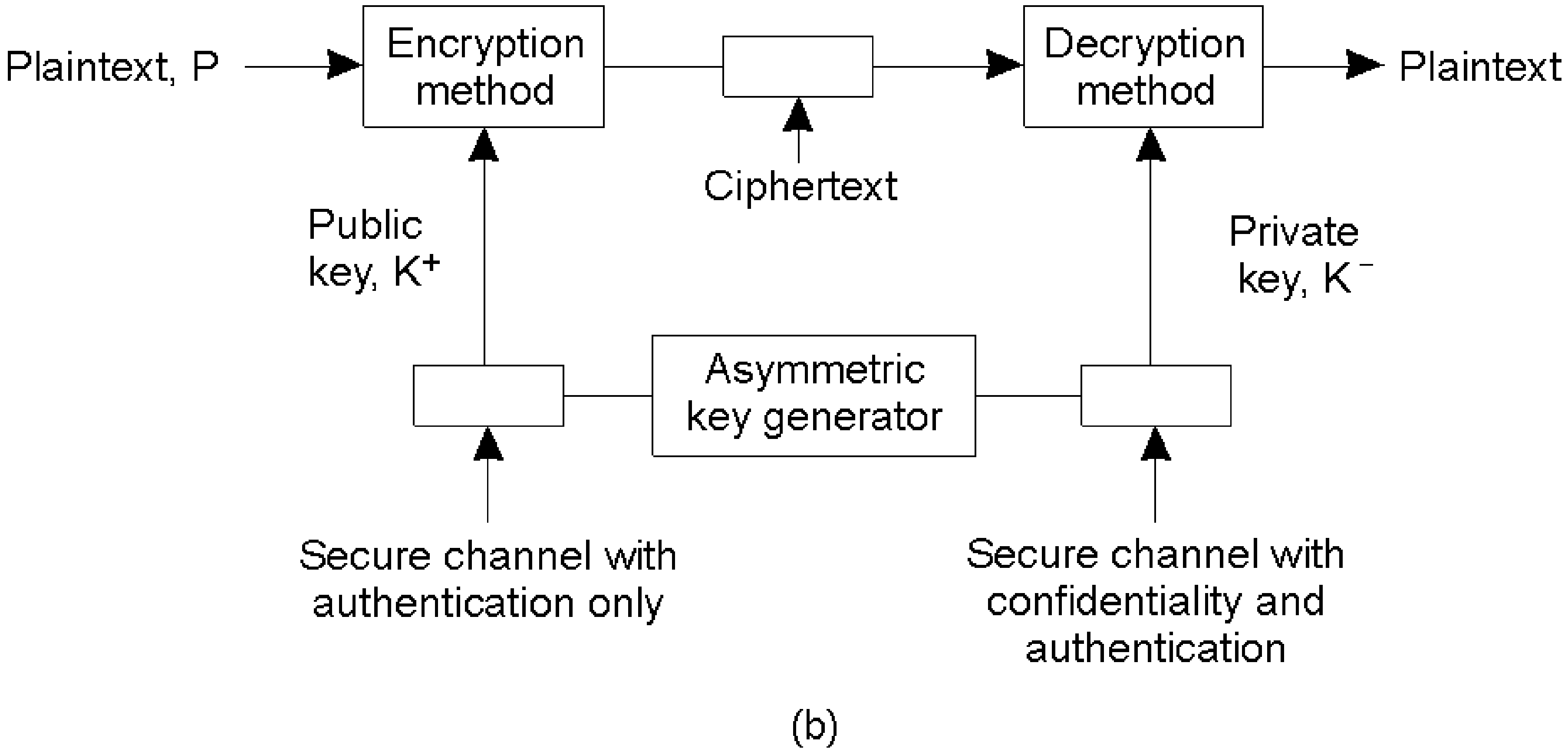
The principle of Diffie-Hellman key exchange – establishing a shared secret-key across an insecure channel.

Key Distribution (1)



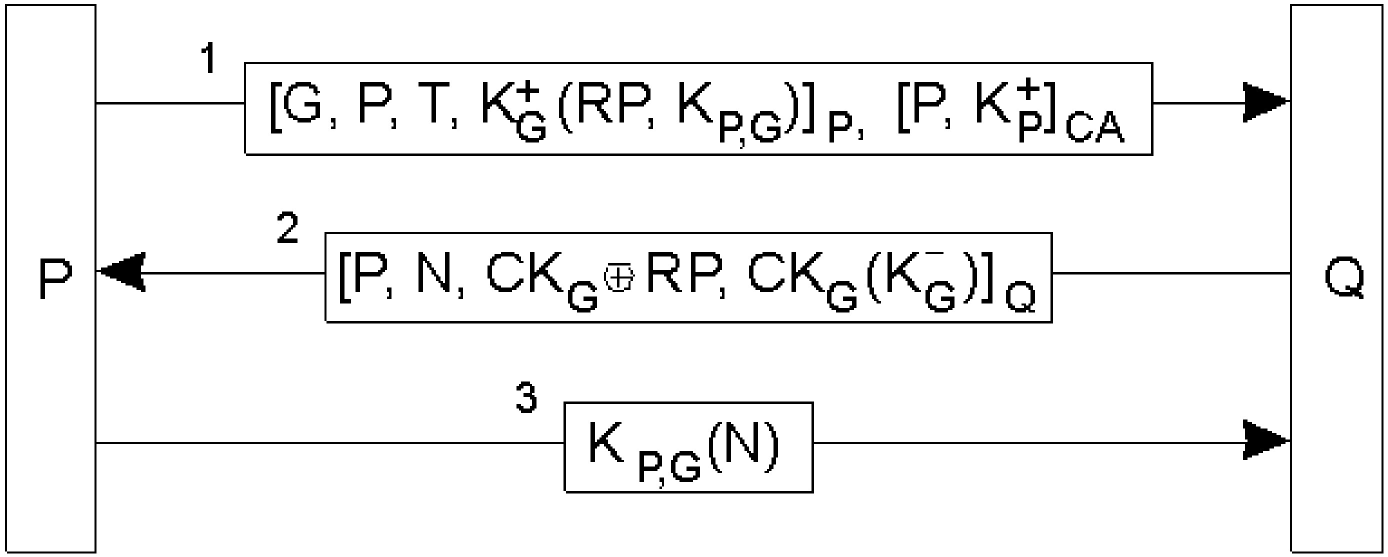
Secret-key distribution

Key Distribution (2)



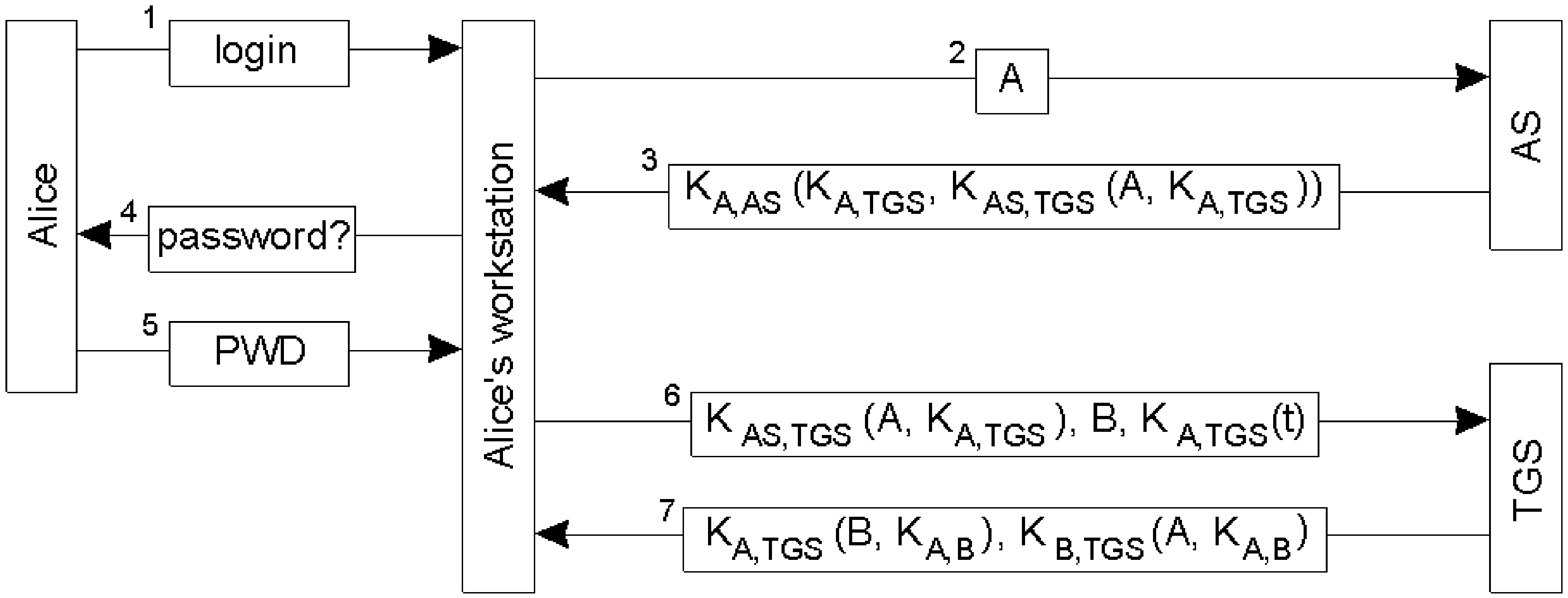
Public-key distribution (see also [menezes.a96]).

Secure Group Management


1
$[G, P, T, K_G^+(RP, K_{P,G})]_P, \ [P, K_P^+]_{CA}$
2
$[P, N, CK_G \oplus RP, CK_G(K_G^-)]_Q$
3
$K_{P,G}(N)$
P
Q

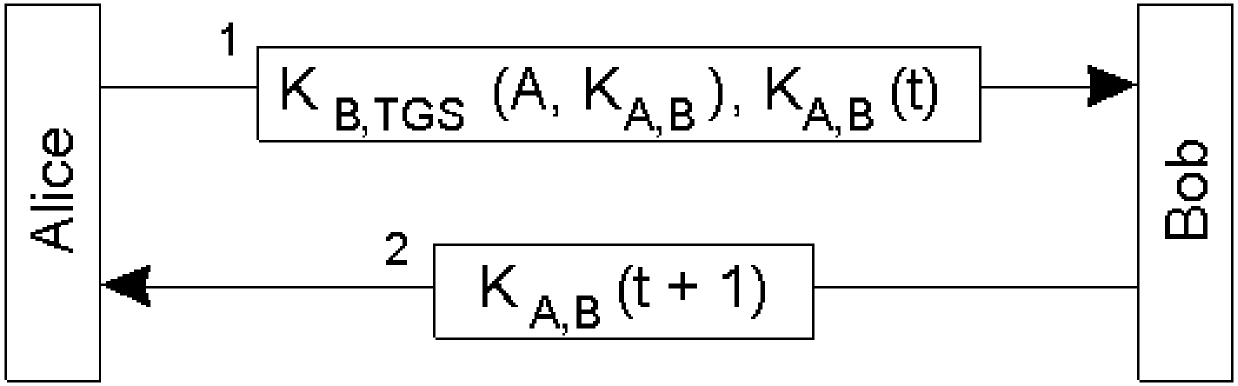
Securely admitting a new group member.

Example: Kerberos (1)


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

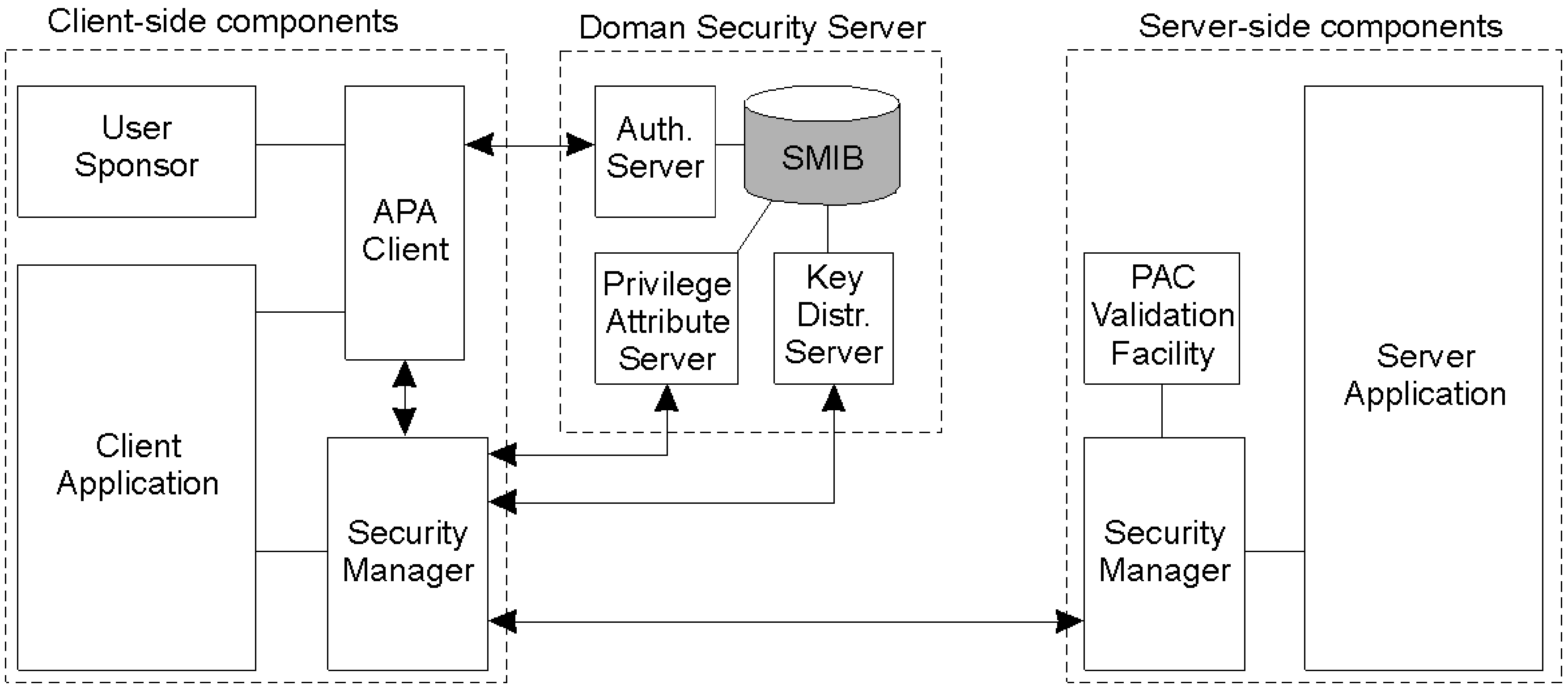
Authentication in Kerberos.

Example: Kerberos (2)



Setting up a secure channel in Kerberos.

SESAME Components



Overview of components in SESAME.

Privilege Attribute Certificates (PACs)

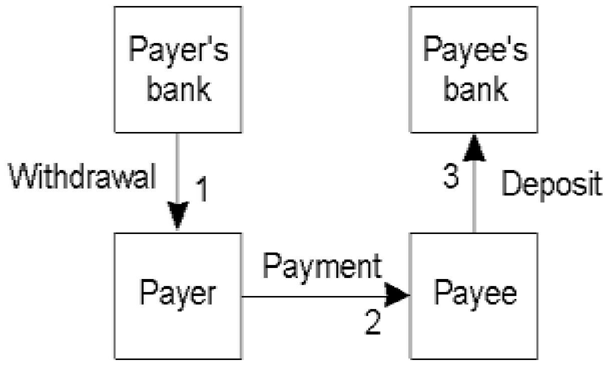
Field	Description
Issuer domain	Name the security domain of the issuer
Issuer identity	Name the PAS in the issuer's domain
Serial number	A unique number for this PAC, generated by the PAS
Creation time	UTC time when this PAC was created
Validity	Time interval when this PAC is valid
Time periods	Additional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

The organization of a SESAME Privilege Attribute Certificate.

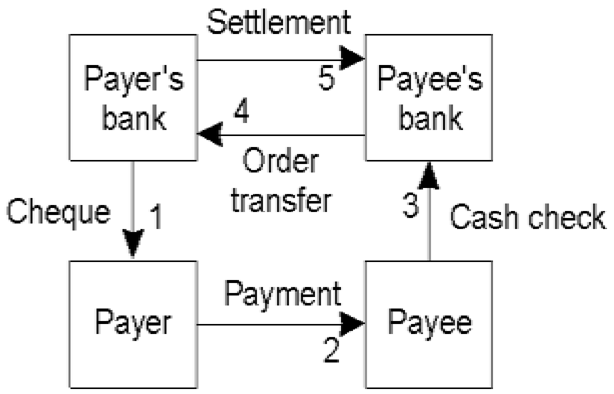
Electronic Payment Systems (1)

Payment systems based on direct payment between customer and merchant.

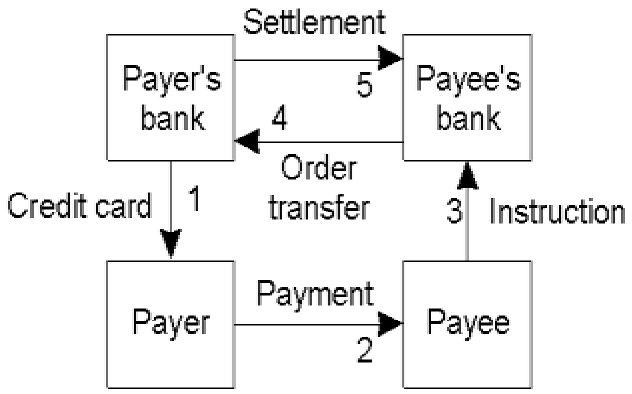
a) Paying in cash.
b) Using a check.
c) Using a credit card.



(a)

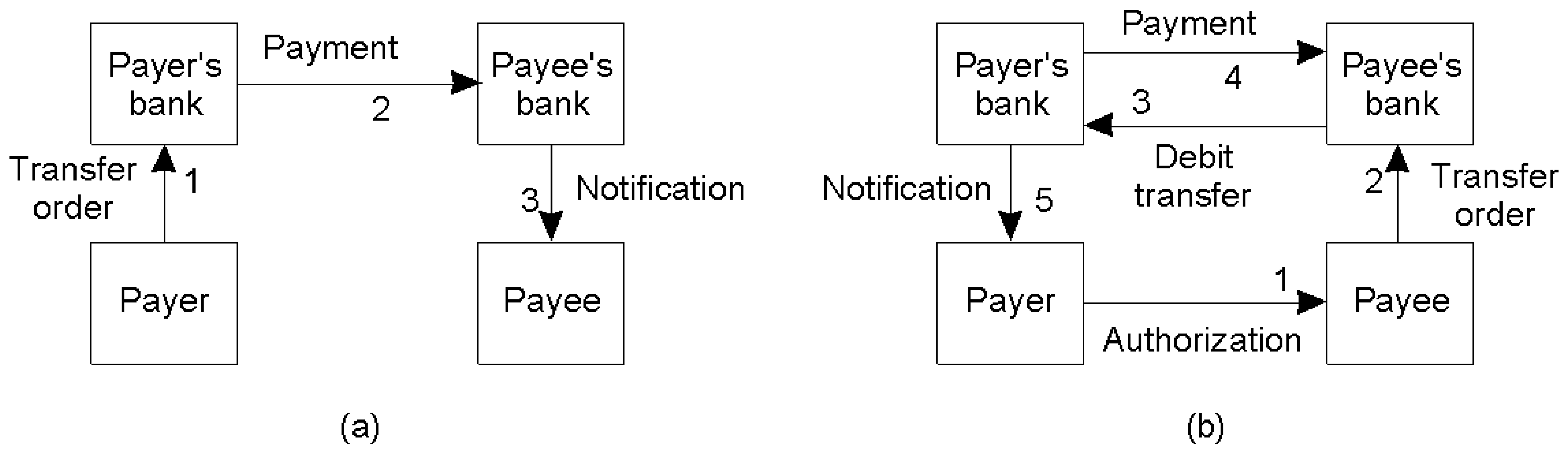


(b)



(c)

Electronic Payment Systems (2)

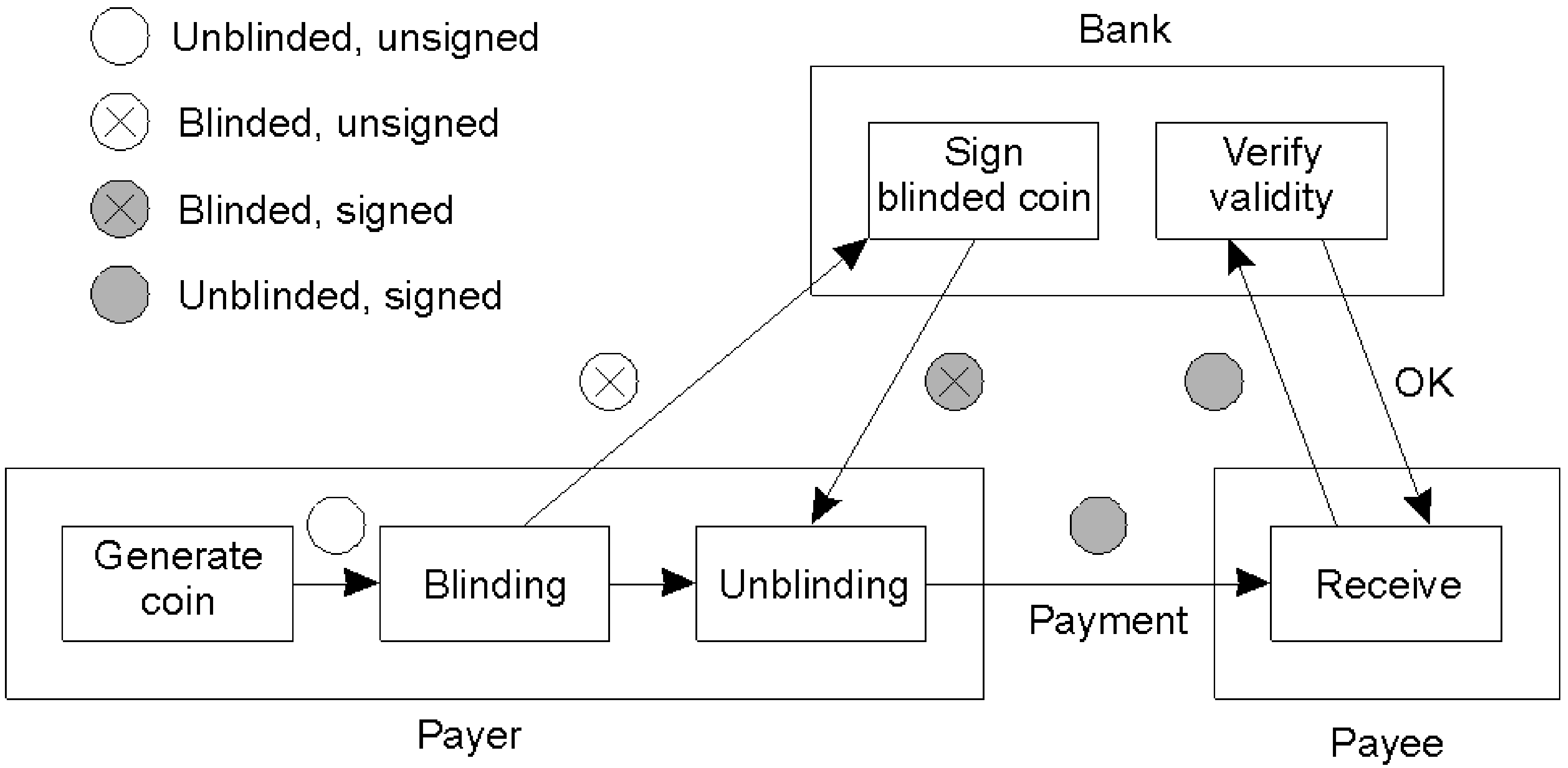


Payment systems based on money transfer between banks.

a) Payment by money order.

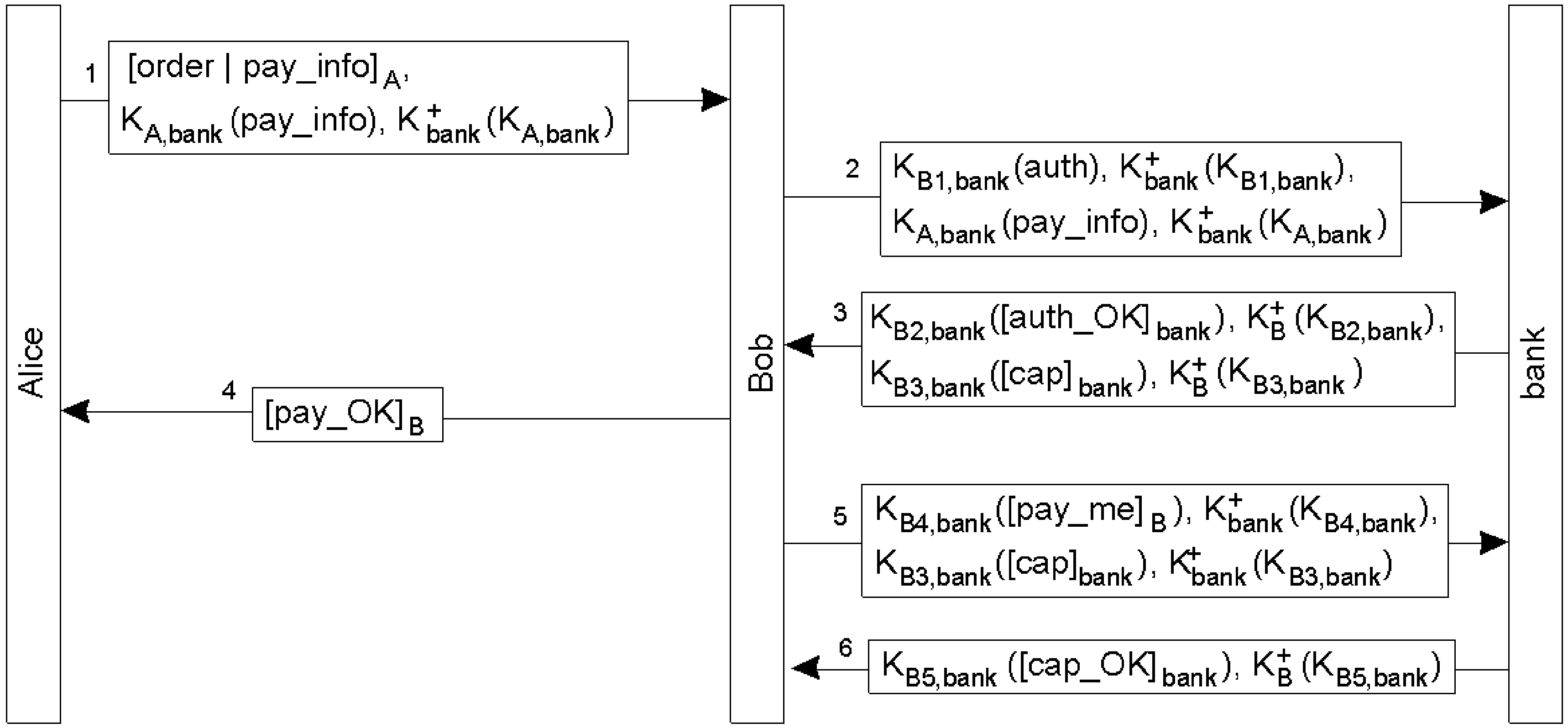
b) Payment through debit order.

E-cash



The principle of anonymous electronic cash using blind signatures.

Secure Electronic Transactions (SET)



Alice
Bob
bank
1
$[order \mid pay\_info]_A$, $K_{A,bank}(pay\_info)$, $K^+_{bank}(K_{A,bank})$
2
$K_{B1,bank}(auth)$, $K^+_{bank}(K_{B1,bank})$, $K_{A,bank}(pay\_info)$, $K^+_{bank}(K_{A,bank})$
3
$K_{B2,bank}([auth\_OK]_{bank})$, $K^+_B(K_{B2,bank})$, $K_{B3,bank}([cap]_{bank})$, $K^+_B(K_{B3,bank})$
4
$[pay\_OK]_B$
5
$K_{B4,bank}([pay\_me]_B)$, $K^+_{bank}(K_{B4,bank})$, $K_{B3,bank}([cap]_{bank})$, $K^+_{bank}(K_{B3,bank})$
6
$K_{B5,bank}([cap\_OK]_{bank})$, $K^+_B(K_{B5,bank})$

The different steps in SET.

Summary

Security plays an extremely important role in Distributed Systems.

It is very difficult to get right.

There are *three main issues*:

1. Establishing a Secure Channel.
2. Providing for Access Control.
3. Managing Key Distribution.