

## Sicurezza nei Sistemi Distribuiti

### Aspetti di Sicurezza

- La sicurezza nei sistemi distribuiti deve riguardare tutti i componenti del sistema e coinvolge due aspetti principali:
  - Le comunicazioni tra utenti e processi
    - » *soluzione* : **canali sicuri**
  - Autorizzazione di utenti e processi
    - » *soluzione* : **controllo degli accessi**
- Meccanismi : **chiavi crittografiche** e **rimozione di utenti.**

## Minacce alla Sicurezza

- Intercettazione  
*(accessi non autorizzati)*
- Interruzione  
*(diniego di servizio - denial of service)*
- Modifica  
*(modifiche di dati non autorizzate)*
- Fabbricazione  
*(inserimenti di dati non autorizzati)*

## Politica di Sicurezza

- Un sistema distribuito sicuro ha bisogno di una  
**politica di sicurezza**  
che definisce  
*le azioni che le entità del sistema possono eseguire e quelle che sono proibite.*
- Una politica può essere realizzata tramite  
**meccanismi di sicurezza.**

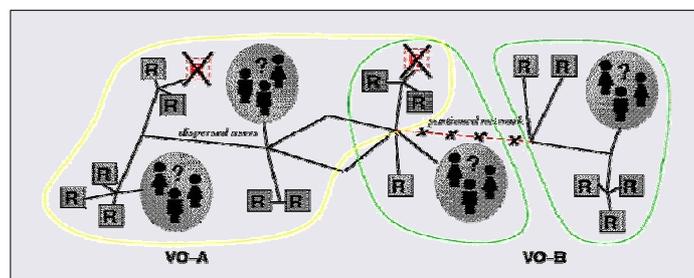
## Meccanismi di Sicurezza

- Crittografia
- Autenticazione
- Autorizzazione
- Auditing

## Esempio: Il Sistema Globus

Globus è un sistema per configurare e usare le Grid.

Una Grid è una infrastruttura di distributed computing. Il suo principale obiettivo è : *Condivisione di risorse & problem solving coordinato in organizzazioni virtuali dinamiche e multi-istituzionali.*



Domini Multipli in una Grid

## Esempio: Politica di Sicurezza di Globus

Le politiche di sicurezza di Globus sono basate sui seguenti principi.

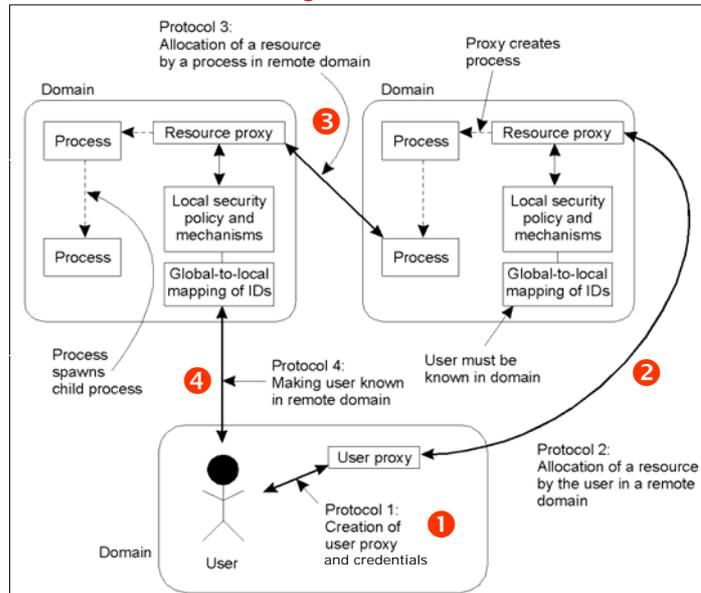
1. Il sistema consiste di diversi domini amministrativi.
2. Le operazioni locali sono soggette solo a politiche di sicurezza locali.
3. Le operazioni globali richiedono che il richiedente sia riconosciuto in tutti i domini interessati alle operazioni.
4. Le operazioni tra entità in domini differenti richiedono la mutua autenticazione.
5. L'autenticazione globale sostituisce la autenticazione locale.
6. Il controllo degli accessi alle risorse è soggetto alle politiche locali.
7. Gli utenti possono delegare loro diritti ai processi.
8. Un gruppo di processi nello stesso dominio può condividere le credenziali.

## Esempio: Globus Security Architecture (1)

- La *Globus security architecture* consiste di entità come utenti, processi, user proxies, e resource proxies.
- Uno **user proxy** è un processo che ha il permesso di agire per conto di un utente per un limitato periodo di tempo.
- Un **resource proxy** è un processo che in un dominio è usato per tradurre operazioni globali su una risorsa in operazioni locali che rispettano la politica di sicurezza del dominio.

## Esempio: Globus Security Architecture (2)

Diagramma della  
*Globus security  
architecture*  
e dei protocolli.

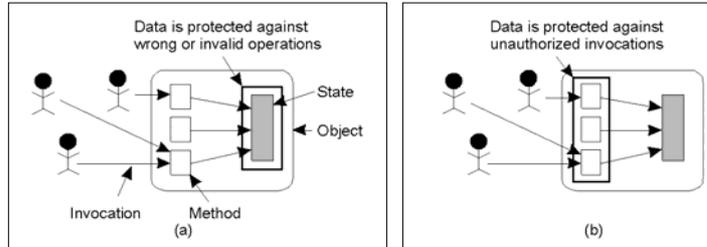


## Aspetti di Progettazione della Sicurezza

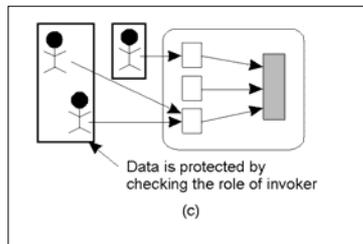
- Politiche di sicurezza possono essere implementate da servizi di sicurezza.
- Diversi aspetti devono essere considerati nella progettazione di politiche di sicurezza:
  - focus del controllo,
  - meccanismi e livelli,
  - semplicità.

## Focus del Controllo

Tre approcci per la protezione da minacce alla sicurezza.

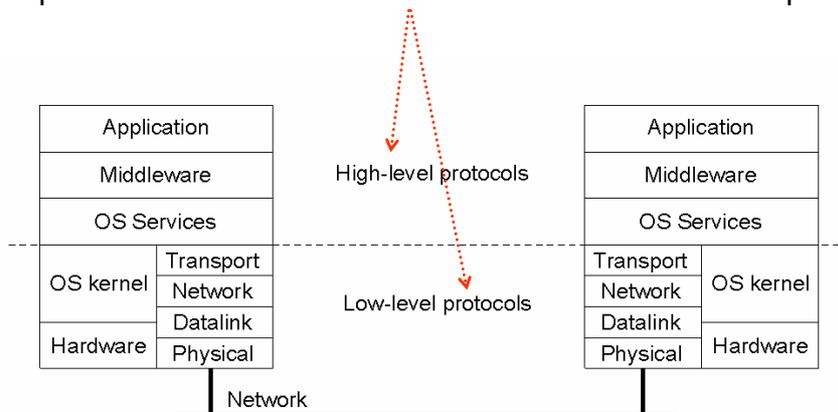


- (a) Protezione da operazioni non valide
- (b) Protezione da invocazioni non autorizzate
- (c) Protezione da utenti non autorizzati



## Livelli dei Meccanismi di Sicurezza (1)

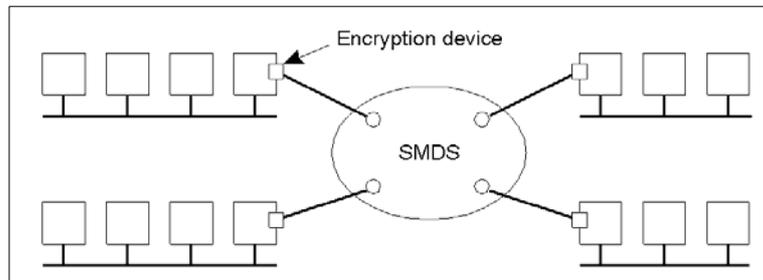
A quale livello i meccanismi di sicurezza devono essere posti?



L'organizzazione logica di un sistema distribuito in diversi livelli.

## Livelli dei Meccanismi di Sicurezza (2)

Esempio: connessione tramite Switched Multi-megabit Data Service



Siti differenti connessi tramite un servizio backbone su WAN.

NB: I meccanismi di sicurezza nei sistemi distribuiti sono generalmente posti a livello del middleware.

## Distribuzione dei Meccanismi di Sicurezza (1)

- Dipendenze tra servizi di sicurezza portano al concetto di

### **Trusted Computing Base**

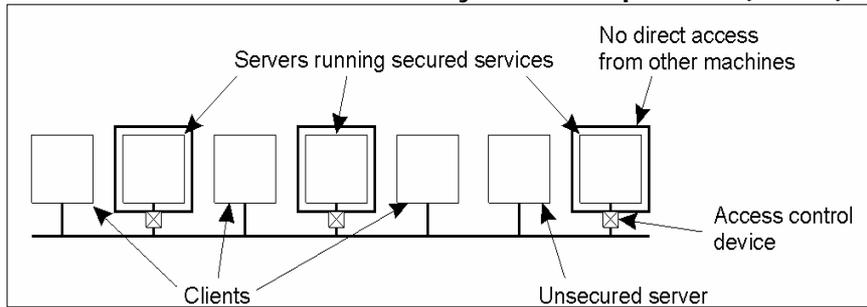
*l'insieme di tutti i meccanismi di sicurezza in un sistema distribuito che sono necessari per rispettare la sicurezza del sistema.*

- Una TCB in un sistema distribuito può includere i sistemi operativi locali dei vari nodi del sistema.
- Esempi: file system distribuito, middleware distribuito.

## Distribuzione dei Meccanismi di Sicurezza (2)

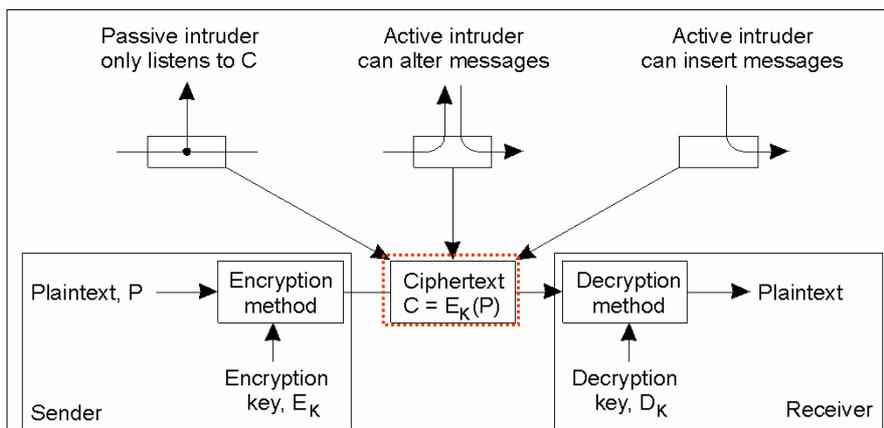
- Servizi di sicurezza possono essere isolati da altri tipi di servizi, riducendo la TCB ad un piccolo sottoinsieme di nodi.

### Reduced Interface for Secure Systems Components (RISCC)



Il principio della RISCC applicato ad un sistema distribuito sicuro

## Crittografia (1)



Intrusioni e ficcanaso nelle comunicazioni.

## Crittografia (2)

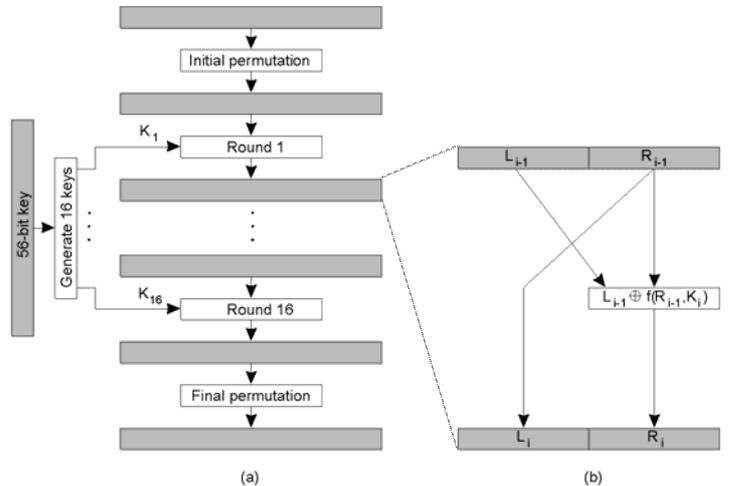
- **Sistema Crittografico Simmetrico** :  $P = D_K(E_K(P))$  : si usa la stessa chiave.
- **Sistema Crittografico Asimmetrico** :  $P = D_{K_D}(E_{K_E}(P))$  : si usano chiavi differenti (una pubblica e una privata): sistema a chiave pubblica

Notazione	Descrizione
$K_{A, B}$	Chiave segreta condivisa tra A e B
$K_A^+$	Chiave pubblica di A
$K_A^-$	Chiave privata di A

## Crittografia Simmetrica : DES (1)

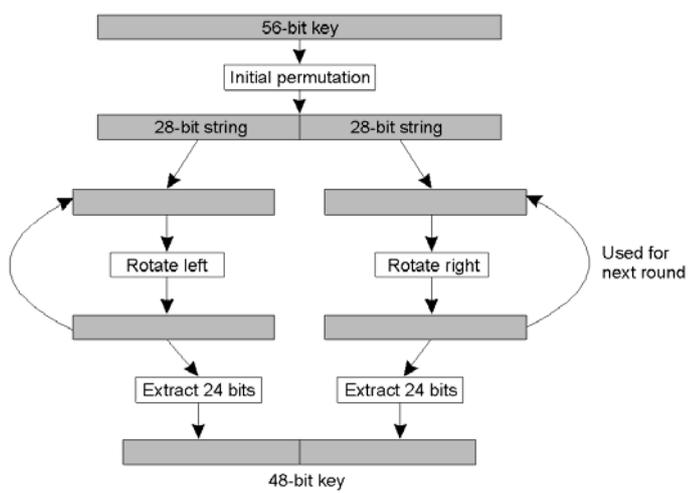
- Sistemi a cifratura simmetrica : *Data Encryption Standard* (**DES**) proposto nel 1970
- I dati sono cifrati operando su blocchi di 64 bit usando 16 chiavi di 48 bit derivati da una chiave master a 56 bit.
- Una *mangler function*  $f$  è usata per cifrare un blocco a 32 bit.
- Estensioni: Triple-DES, DESX e AES (standard)

## Crittografia Simmetrica : DES (2)



(a) Il principio del DES      (b) Schema di un ciclo di cifratura

## Crittografia Simmetrica : DES (3)



Dettagli della generazione di chiavi per ciclo nel DES

## Sistemi a Chiave Pubblica: RSA (1)

Sistema crittografico asimmetrico definito da Rivest, Shamir e Adleman (RSA) nel 1977, basato sul fatto che è molto complesso trovare i fattori primi di un numero di valore elevato.

- Ogni numero intero può essere scritto come il prodotto di numeri primi.
- Le chiavi pubbliche e private sono costruite a partire da numeri primi molto grandi.
- Decodificare RSA è equivalente a trovare quei numeri primi.

## Sistemi a Chiave Pubblica : RSA (2)

La generazione della chiave privata e della chiave pubblica è basata su quattro passi:

1. Selezione di due numeri primi grandi,  $p$  e  $q$
2. Calcolo di  $n = p \times q$  e  $z = (p - 1) \times (q - 1)$
3. Selezione di un numero  $d$  che è primo per  $z$
4. Calcolo del numero  $e$  tale che  $e \times d = 1 \pmod{z}$

$d$  può essere usato per la decifratura ed  $e$  per la cifratura.

Ogni messaggio viene diviso in blocchi  $m_i$  e

*Per cifrare un blocco:*  $c_i = m_i^e \pmod{n}$

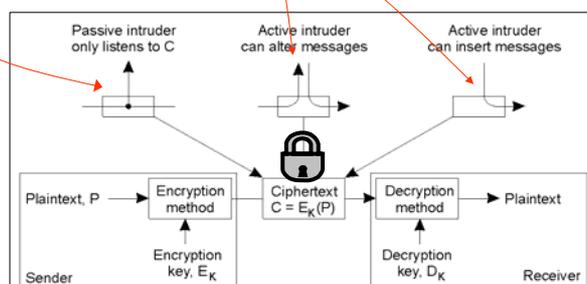
*Per decifrare un blocco:*  $m_i = c_i^d \pmod{n}$

## Sicurezza

- La sicurezza nei sistemi distribuiti coinvolge due aspetti principali:
  - Le comunicazioni tra utenti e processi
    - » *soluzione* : **canali sicuri**
  - Autorizzazione di utenti e processi
    - » *soluzione* : **controllo degli accessi**

## Canali Sicuri (1)

- Un canale sicuro protegge il mittente e il destinatario da
  - **Intercettazione**  
*accesso non autorizzato al canale*
  - **Modifica**  
*modifica non autorizzata di un messaggio*
  - **Fabbricazione**  
*inserimento non autorizzato di un messaggio*

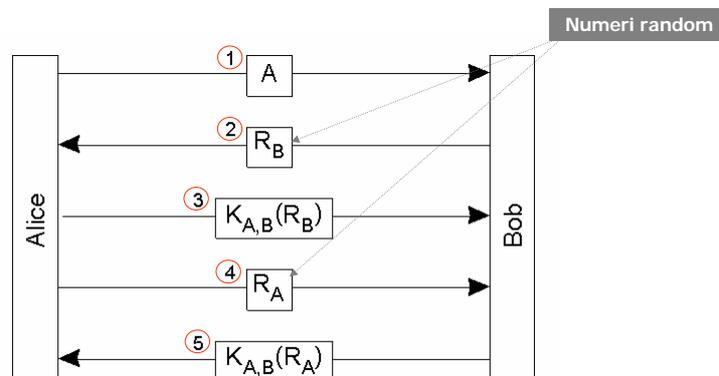


## Canali Sicuri (2)

- Autenticazione e integrità dei messaggi devono essere garantiti insieme.
- Viene usata una chiave segreta associata ad un canale: **session key**.
- Quando viene chiuso un canale la *session key* viene eliminata e distrutta.

## Autenticazione Basata su Chiave Segreta (1)

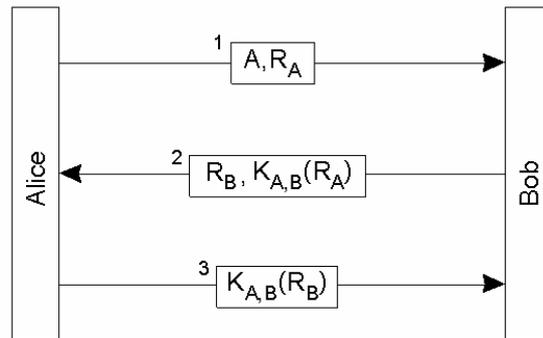
**Protocolli Challenge-Response** per la mutua autenticazione di due "parti" che condividono una chiave segreta



Autenticazione basata su una *shared secret key* ( $K_{A,B}$ )

## Autenticazione Basata su Chiave Segreta (2)

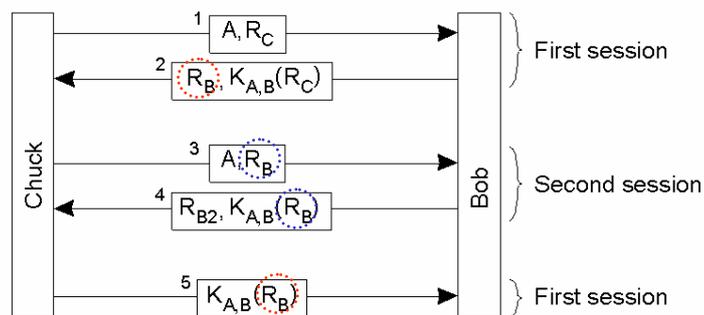
E' possibile realizzare una ottimizzazione del Protocollo Challenge-Response ??



Possibile protocollo di autenticazione basato su chiave segreta che usa solo 3 invece di 5 messaggi.

## Autenticazione Basata su Chiave Segreta (3)

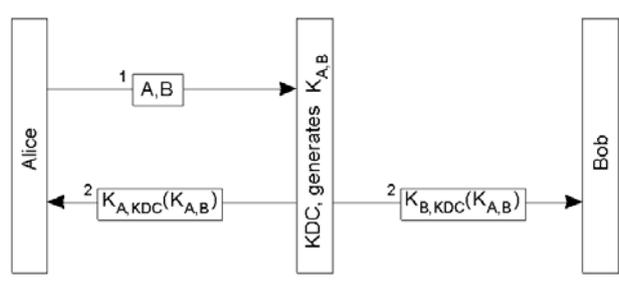
Il **Reflection Attack** fa fallire il protocollo a 3 messaggi.



Per risolvere questo problema **devono essere usati challenge differenti.**

## Autenticazione con Uso di un Key Distribution Center (1)

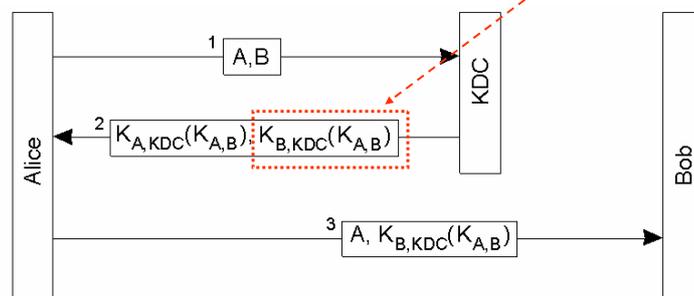
- In un sistema distribuito composto da  $N$  host, ogni host condivide  $N-1$  chiavi e globalmente sono necessarie  $N(N-1)/2$  chiavi segrete.
- In questo caso può essere usato un approccio centralizzato => il **Key Distribution Center (KDC)** gestisce solo  $N$  chiavi.



Il principio di uso di un KDC.

## Autenticazione con Uso di un Key Distribution Center (2)

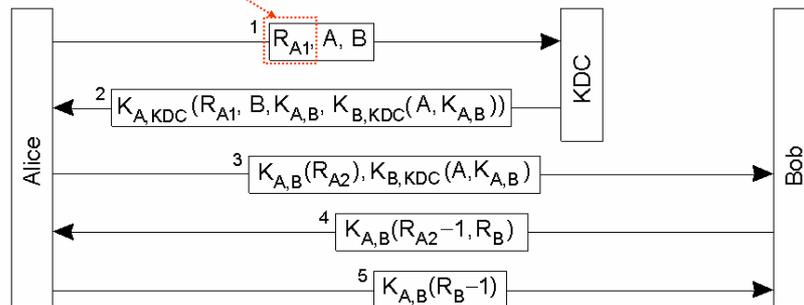
- Il KDC può passare  $K_{B,KDC}(K_{A,B})$  ad  $A$  affida ad  $A$  il compito di connettersi a  $B$ . Il messaggio è chiamato **ticket**.



Uso di un ticket per permettere ad Alice di connettersi a Bob.

## Autenticazione con Uso di un Key Distribution Center (3)

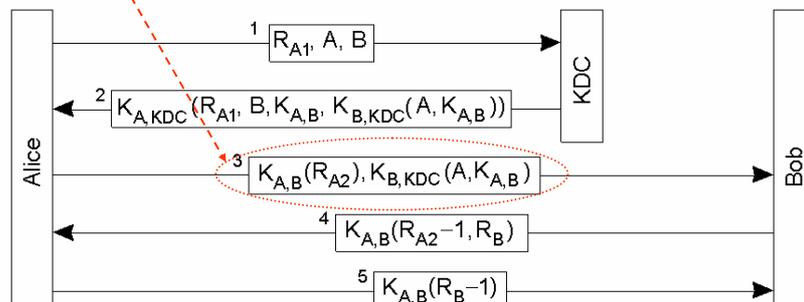
**Nonce:** numero random usato solo una volta.



Il protocollo di autenticazione di **Needham-Schroeder**.

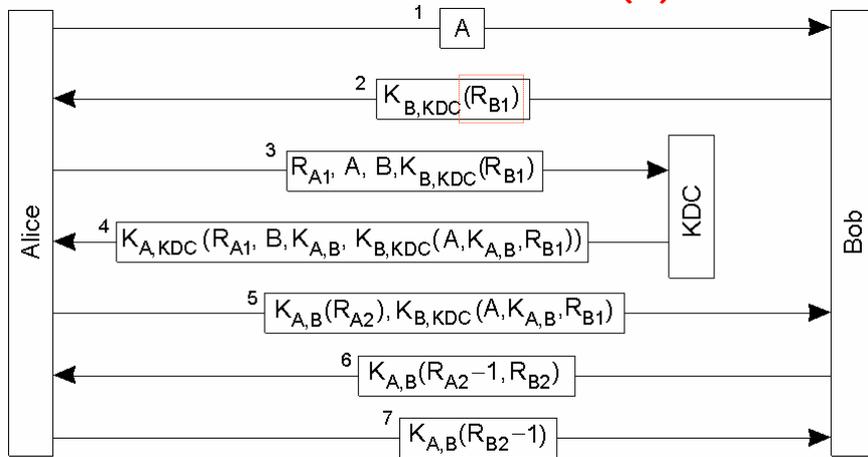
## Autenticazione con Uso di un Key Distribution Center (4)

- Il messaggio 3 deve essere correlato al messaggio 1 per evitare l'intrusione di un'altra entità.
- Soluzione: Un nonce inviato inizialmente da Bob deve essere usato nel messaggio di Alice.



Il protocollo di autenticazione di **Needham-Schroeder**.

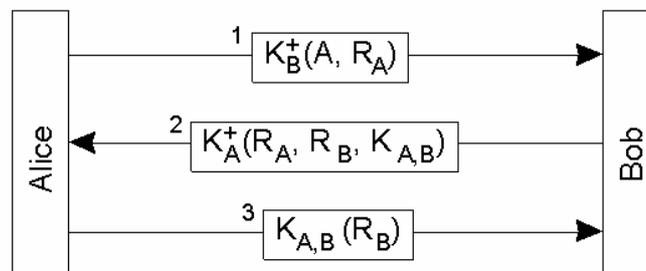
## Autenticazione con Uso di un Key Distribution Center (5)



Protezione da riuso scorretto di una chiave generata nella sessione precedente nel protocollo di Needham-Schroeder.

## Autenticazione con Crittografia a Chiave Pubblica

- Alice vuole definire un canale sicuro con Bob e ambedue posseggono la chiave pubblica dell'altro.
- Alice genera una session key usata per le successive comunicazioni.



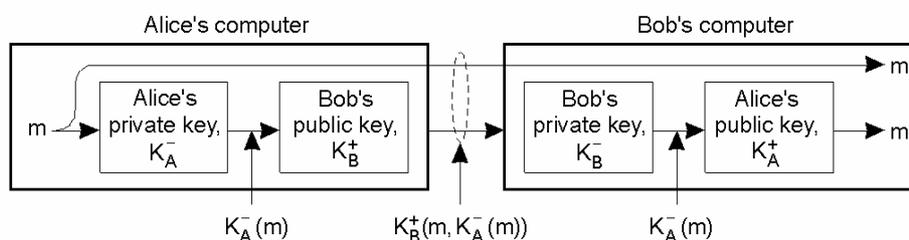
Mutua autenticazione in un sistema a chiave pubblica senza un KDC.

## Messaggi: Integrità e Confidenzialità

- Oltre all'autenticazione, un canale sicuro deve garantire:
  - **confidenzialità** contro l'intercettazione  
*usando crittografia*
  - **integrità dei messaggi** contro le modifiche  
*usando firme digitali*

## Firme Digitali (1)

- La firma digitale può essere usata per assicurare che eventuali modifiche al messaggio non passino inosservate.



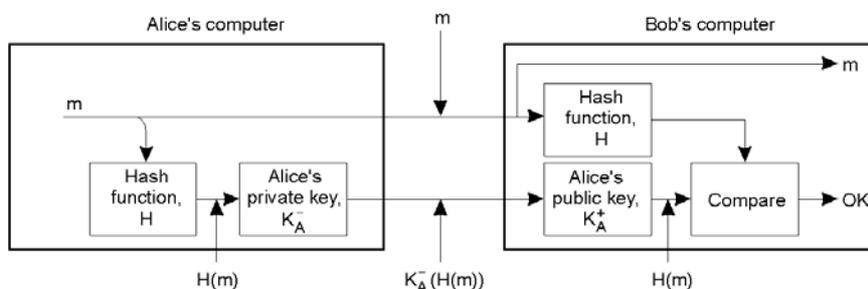
Firma digitale di un messaggio usando la crittografia a chiave pubblica

## Firme Digitali (2)

- Potenziali problemi:
  - La chiave privata di Alice deve rimanere segreta
    - » *L'integrità del messaggio è persa se la chiave viene rubata*
  - Cosa accade se Alice cambia la sua chiave ?
    - » *Una autorità centrale deve tenere traccia della modifica della chiave*
  - La cifratura del messaggio può costare molto in termini di necessità di elaborazione.
    - » *Può essere usato un estratto del messaggio*

## Firme Digitali (3)

- Un estratto di un messaggio è una stringa di bit di lunghezza fissata  $h$  ottenuta da un messaggio  $m$  per mezzo di una funzione hash crittografica  $H$ .
- Se  $m \neq m' \implies H(m') \neq H(m) = h$



Firma digitale di un messaggio usando un estratto del messaggio.

## Comunicazioni Sicure di Gruppo

- In un sistema distribuito è anche necessario disporre di comunicazioni sicure di gruppo tra più di due processi.
- Ad esempio,
  - **Quando un server deve comunicare con un insieme di clienti,**
  - **Quando diversi server che gestiscono repliche di dati devono comunicare usando operazioni di multicast**garantendo la sicurezza rispetto a modifiche, intercettazioni, e fabbricazioni.

## Comunicazioni Confidenziali di Gruppo

- Se deve essere assicurata la confidenzialità:
  - Uno schema semplice può essere basato su una chiave segreta condivisa tra i processi del gruppo. *(soluzione vulnerabile)*
  - Uno schema più complesso usa chiavi segrete tra coppie di processi. *(molte chiavi)*
  - Può essere usato uno schema a chiave pubblica, ogni processo nel gruppo ha la propria coppia  
*(chiave pubblica, chiave privata)*  
Sono necessarie *N coppie di chiavi.*

## Servizi Sicuri Replicati (1)

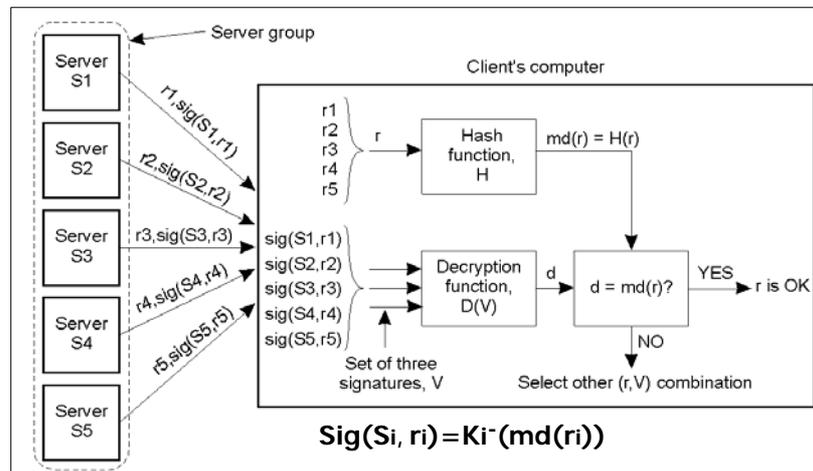
- **Obiettivo:** un cliente invia una richiesta ad un gruppo di server replicati e deve ricevere una risposta sicura da loro.
- Reiter ha proposto una soluzione per un server sicuro replicato che mantiene la trasparenza alla replicazione (i clienti non conoscono il numero delle repliche)
- La soluzione è basata sul **secret sharing**: *un gruppo di processi condivide un segreto, ma nessuno di loro conosce l'intero segreto.*

## Servizi Sicuri Replicati (2)

- Stiamo cercando una soluzione per cui al più  $c$  degli  $N$  server possono essere corrotti da una intrusione, ma una risposta corretta può essere comunque prodotta per un cliente.
- Ogni server  $S_i$  produce una risposta  $r_i$  che ritorna al cliente con una firma digitale  $Sig(S_i, r_i)$ . Con  $md(r_i)$  denotiamo l'estratto del messaggio calcolato dal server  $S_i$ .  
$$Sig(S_i, r_i) = K_i^{-1}(md(r_i))$$
- La soluzione è basata sull'uso di  
almeno  $c + 1$  firme digitali  
per costruire una firma valida per la risposta.

## Servizi Sicuri Replicati (3)

Un esempio di  $N=5$  server replicati capaci di tollerare  $c=2$  server corrotti. Viene usata una funzione  $D(V)$  di 3 firme.



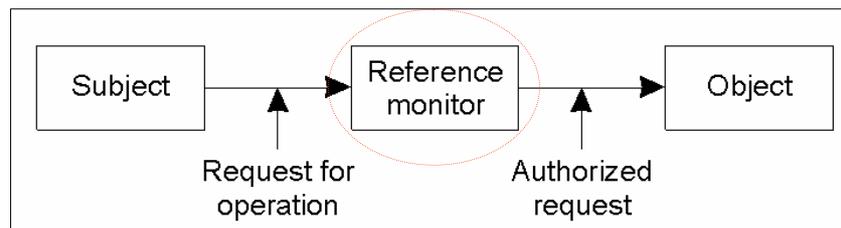
Condivisione di una firma segreta in a gruppo di server replicati

## Schema Basato su Threshold

- Questo metodo è detto  **$(c+1, N)$ -threshold scheme** (schema basato su un valore soglia).
- In questo schema, un messaggio può essere diviso in  $N$  parti, chiamate **shadows**, ed un gruppo di  $c+1$  shadows (con  $c+1 < N$ ) possono essere usate per ricostruire il messaggio.
- Ma usando solo fino a  $c$  shadows la ricostruzione del messaggio è impossibile
- Sono state proposte diverse implementazioni del  **$(c+1, N)$ -threshold scheme**.

## Problemi Generali nel Controllo degli Accessi

- **Controllo degli accessi** : verifiche dei diritti di accesso.
- **Autorizzazioni**: Concessioni dei diritti di accesso.



Modello generale del controllo degli accessi ad oggetti.

## Matrici di Controllo degli Accessi (1)

- I diritti di accesso sono modellati tramite una matrice degli accessi
- Un oggetto è rappresentato da una colonna e un soggetto da una riga.
- Una entry  $M[s,o]$  specifica quali operazioni il soggetto  $s$  può richiedere su un oggetto  $o$ .

object \ subject	O <sub>1</sub>	O <sub>2</sub>	O <sub>3</sub>	O <sub>4</sub>
S <sub>1</sub>	read		read	
S <sub>2</sub>				print
S <sub>3</sub>		read	execute	
S <sub>4</sub>	read write		read write	

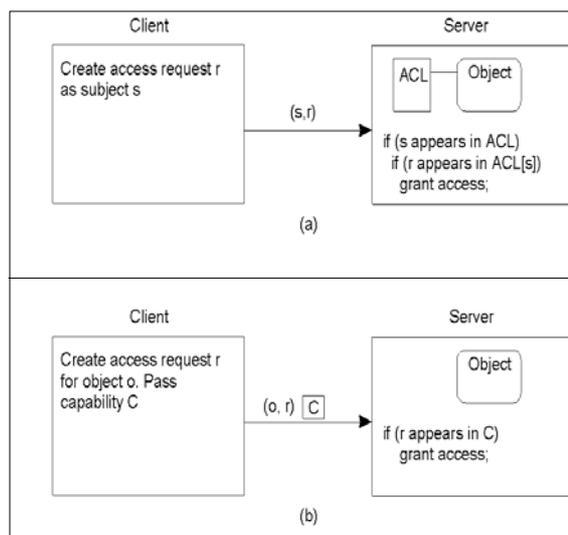
## Matrice di Controllo degli Accessi (2)

- Una matrice degli accessi non va realizzata tramite una matrice quando molti oggetti e soggetti devono essere gestiti: molti elementi potrebbero essere nulli.
- Due soluzioni principali:
  - **Access Control List (ACL)** : ogni oggetto mantiene una lista dei diritti dei soggetti
  - **Capabilities** : ogni soggetto mantiene una lista di capabilities che ha per gli oggetti

## Matrice di Controllo degli Accessi (3)

**Confronto tra ACL e capabilities per la protezione degli oggetti.**

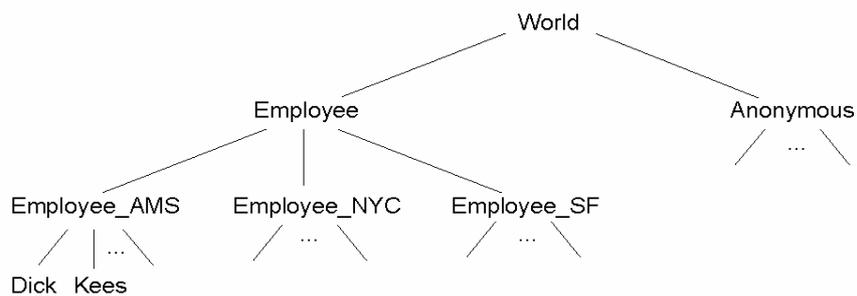
- (a) Usando una **ACL**
- (b) Usando **capabilities**



## Domini di Protezione (1)

ACL e capability list possono essere limitate se si usano i domini di protezione

Un **dominio di protezione** è un insieme di coppie  
(oggetto, diritti di accesso)



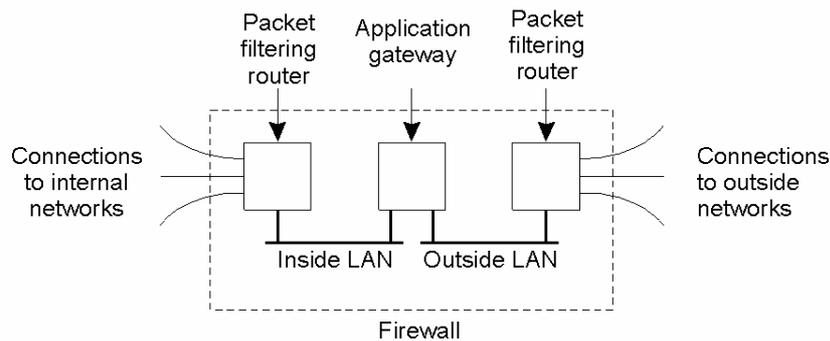
Organizzazione gerarchica dei domini di protezione come gruppi di utenti

## Domini di Protezione (2)

- Altri meccanismi per il controllo degli accessi efficiente sono:
  - subject certificates
  - accessi basati sui ruoli
  - raggruppamento di oggetti
  - combinazione di domini di protezione e raggruppamento di oggetti.

## Firewall (1)

- Quando alcune entità in un sistema distribuito non rispettano le stesse regole altri meccanismi sono necessari per realizzare sistemi sicuri (ad esempio, un firewall).



Una tipica implementazione di un firewall.

## Firewall (2)

- Un firewall è una risorsa critica in un sistema distribuito.
- Non dovrebbe mai fallire!
- Un firewall distribuito dovrebbe essere più sicuro di un firewall centralizzato.

## Codice Mobile Sicuro

- Quando un agente mobile si sposta sulla rete :
  1. L'agente dovrebbe essere protetto da host "malevoli" che tentano di modificare il suo comportamento.
  2. Un host dovrebbe essere protetto da agenti con intenzioni ostili.
- Sono quindi necessari modelli di controllo degli accessi.

## Proteggere gli Agenti (1)

- Quando un agente si sposta su un host, quell'host non dovrebbe poter modificare o alterare il codice e l'informazione dell'agente.
- Se non è possibile garantire nessuna modifica, almeno gli agenti dovrebbero essere organizzati in modo tale che le **modifiche siano rilevate**.
- Questo approccio è usato in **Ajanta** usando
  - *read-only state*,
  - *append-only logs*, e
  - *selective revealing*.

## Proteggere gli Agenti (2)

- Un **read-only state** è una **collezione di dati firmati** dal proprietario dell'agente.
- Un estratto crittografato di un **messaggio viene associato** ad un read-only state.
- Un host può controllare la correttezza dello stato controllando l'estratto del messaggio firmato dello stato originale e confrontandolo con esso.

## Proteggere gli Agenti (3)

- Gli **Append-only logs** impediscono di modificare o cancellare i dati ma permettono di aggiungere dei dati ai log.
- I dati sono memorizzati nel **log** usando la chiave pubblica del proprietario e una funzione checksum.
- Quando i dati sono acceduti dal proprietario (perchè l'agente ritorna dal suo proprietario), questo può controllare la loro sicurezza usando la sua chiave privata.

## Proteggere gli Agenti (4)

- Il **Selective revealing** dello stato è costruito usando un array di dati in cui ogni elemento è definita per un server specifico.
- I dati nell'array sono crittografati usando la chiave pubblica del server corrispondente.
- L'array globale è firmato dal proprietario per garantire la sua integrità.

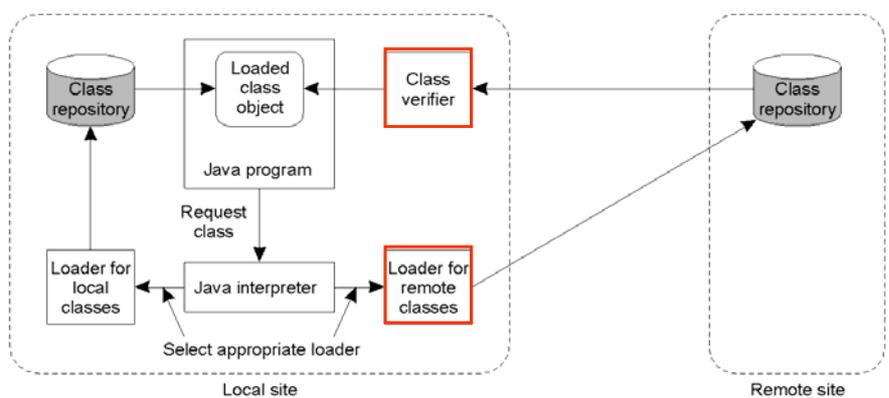
## Proteggere gli Host (1)

- Gli host devono essere protetti da agenti, applet, e simili che avessero intenzioni ostili.
- Il codice malevolo deve essere identificato **prima dell'esecuzione**, dopo sarebbe troppo tardi!
- Tecniche usate:
  - *sandbox*,
  - *playground*, e
  - *stack introspection*.

## Proteggere gli Host (2)

- Un **sandbox** permette di controllare l'esecuzione delle istruzioni ed è implementato inserendo istruzioni aggiuntive per controllare gli agenti durante l'esecuzione.
- Java permette l'implementazione di sandbox tramite l'approccio basato sulle classi.
  1. Sono usati solo class loader sicuri.
  2. Una verifica del byte-code viene usata per controllare istruzioni che potrebbero corrompere lo stack o la memoria.
  3. E' usato un security manager per controlli durante l'esecuzione per controllare accessi a file e a dati.

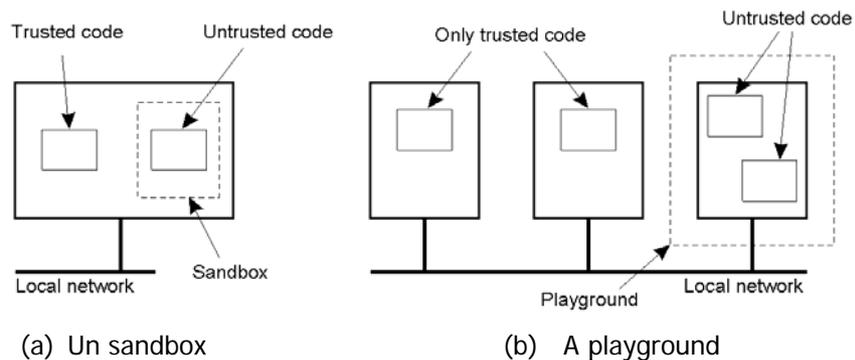
## Proteggere gli Host (3)



L'organizzazione di un sandbox in Java.

## Proteggere gli Host (4)

Un **playground** è un host separato esclusivamente riservato ad eseguire codice mobile. Nessun codice remoto verrà eseguito su host esterni al playground.



## Gestione delle Autorizzazioni

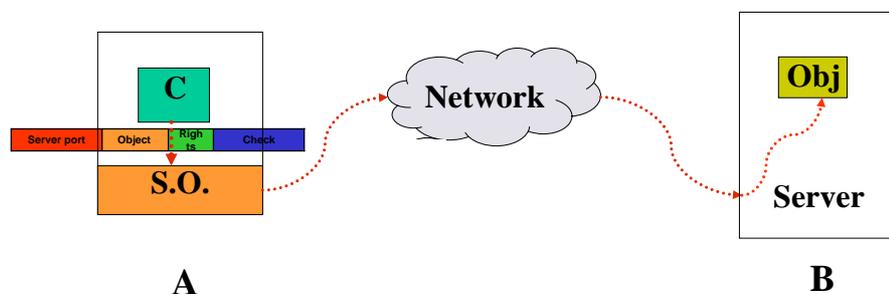
- Due semplici schemi usati nei sistemi distribuiti per gestire i diritti di accesso sono:
  - Creazione di account di ogni utente su ogni host, (*es., network operating system*)
  - creazione di un singolo account su un server centrale.

## Capabilities e Attribute Certificates (1)

- Uno schema più flessibile può essere basato su **capability con possessori associate ad una risorsa**.
- Esistono differenti implementazioni delle capability.
- Ad esempio, **Amoeba** è un sistema distribuito basato su oggetti.
- Ogni oggetto risiede su un server e i clienti possono accedere oggetti remoti usando dei proxy.

## Capabilities e Attribute Certificates (2)

- In Amoeba, per invocare una operazione su un oggetto, un cliente passa una capability al S.O. locale che localizza il server per un oggetto e invia una RPC al server.



## Capabilities e Attribute Certificates (3)

- Una capability contiene l'**indirizzo del server**, l'**object id**, i **diritti di accesso** e un campo di controllo (**check**).

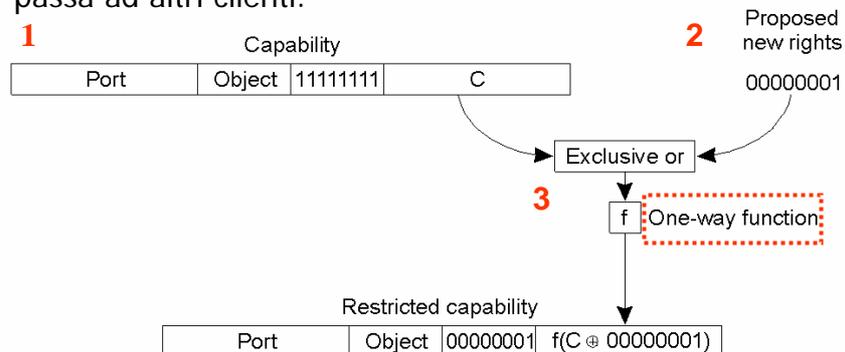
Una capability in Amoeba



Il campo **check** viene mantenuto anche sul server per una verifica della integrità della capability

## Capabilities e Attribute Certificates (4)

Una **restricted capability** può essere creata dal server con minori diritti quando un cliente ne fa richiesta e la passa ad altri clienti.



Generazione di una *restricted capability* da una capability del proprietario.

## Capabilities e Attribute Certificates (5)

- Quando una capability ritorna al server, esso può notare che non è una capability del proprietario e verifica il campo check.
- Un client non può aggiungere diritti ad una restricted capability.
- La protezione è implementata tramite una **funzione one-way** (non invertibile).
- Una modifica al campo check sarà così rilevata.

## Capabilities e Attribute Certificates (6)

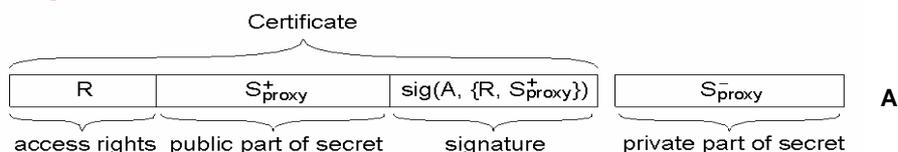
- Alcuni sistemi distribuiti usano un **attribute certificate**.
- Le liste delle coppie (*attributo, valore*) degli **attribute certificate** indicano i diritti di accesso che il possessore ha rispetto alla risorsa identificata.
- Gli attribute certificates sono gestiti dalle **autorità degli attribute certificates** che possono essere diverse dai server che gestiscono le risorse e sono usate per firmare i diritti di accesso contenuti in un certificato.

## Delegation (1)

- La **Delegation** dei diritti di accesso è una tecnica utile per le applicazioni distribuite:  
il passaggio dei diritti da un processo ad un altro è importante in molti casi (es., stampante, compilatore, file handler) per evitare trasferimento di dati.
- I **Delegation proxies** sono token usati per passare diritti da un processo ad un altro processo che potrà operare su una risorsa.
- La Delegation può essere **esplicita** ("A dichiara che B ha i diritti D") o **implicita** ("Il possessore ha i diritti D").
- Il secondo caso necessita di una implementazione più sicura.

## Delegation (2)

- **Un Delegation Proxy** contiene due parti
  - I diritti di accesso delegati
  - La parte pubblica del segreto per autenticare il possessore
  - La firma del processo che ha creato il proxy (A) per proteggerlo contro eventuali modifiche,
  - la parte privata del segreto per autenticare il possessore

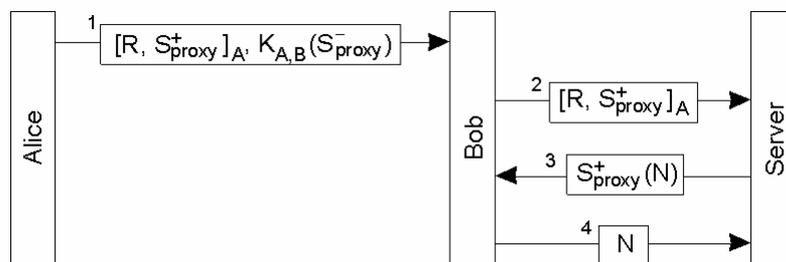


La struttura generale di un delegation proxy.

## Delegation (3)

Uso di un proxy per delegare e provare la proprietà di diritti di accesso.

- Bob chiede al server di operare su un oggetto il cui proprietario (Alice) gli ha passato i diritti di accesso.
- Bob usa la parte privata del segreto  $S_{\text{proxy}}$  per autenticare se stesso come il corretto possessore del certificato.



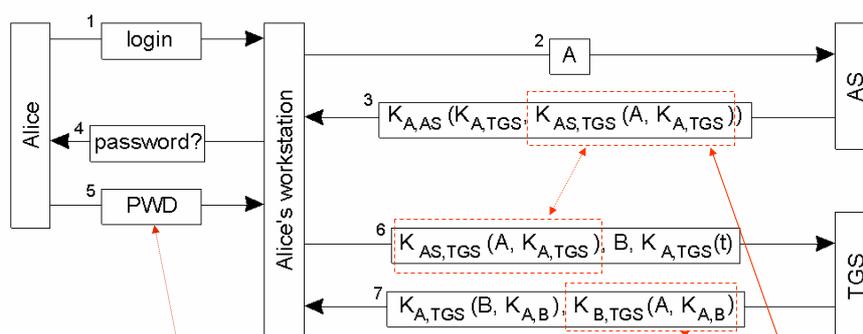
## Esempio: Kerberos (1)

- **Kerberos** è un sistema sviluppato al MIT per supportare l'implementazione della sicurezza in sistemi distribuiti. In particolare, Kerberos assiste i clienti si stabilire canali sicuri con un server.
- Kerberos è basato sul protocollo di autenticazione di **Needham-Schroeder**.
- Kerberos usa due componenti principali:
  - L'Authentication Server (AS)
  - Il Ticket Granting Service (TGS).

## Esempio: Kerberos (2)

- L'**Authentication Server (AS)** autentica un utente e fornisce una chiave da essere usata per rendere sicuri i canali con un server.
- Il **Ticket Granting Service (TGS)** stabilisce canali sicuri con un server usando ticket (chiavi segrete crittografate).

## Esempio: Kerberos (3)

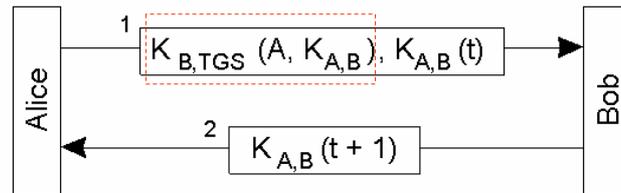


La password (PWD) è usata per generare  $K_{A,AS}$

**tickets**

Autenticazione in Kerberos.

## Esempio : Kerberos (4)



Creazione di un canale sicuro in Kerberos.