

Autenticazione a Livello Applicazione

Autenticazione

1

Motivazioni per l'Autenticazione

- L'autenticazione interviene in ambienti aperti, in cui client di vario genere richiedono di accedere a vari servizi
 - L'autenticazione consente l'accesso ai soli utenti autorizzati
- Le workstation utente potrebbero non essere fidate
- Altre soluzioni (password, firewall, etc) potrebbero non essere sufficienti

Autenticazione

2

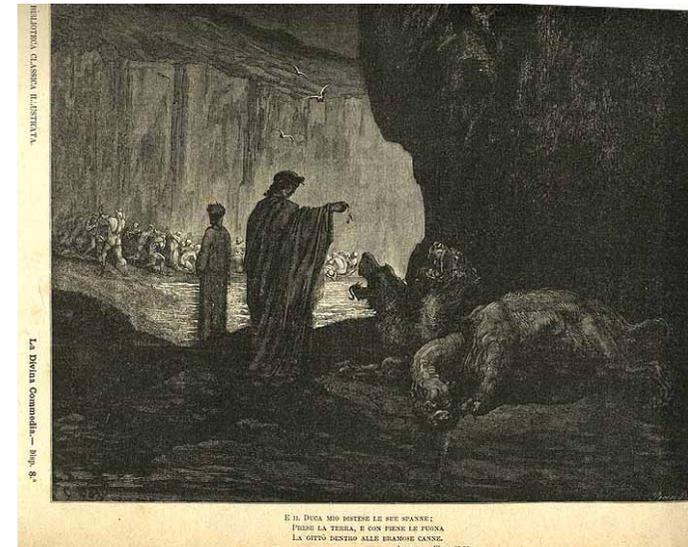
Due approcci

- Kerberos
- X.509

Autenticazione

3

KERBEROS



Autenticazione

4

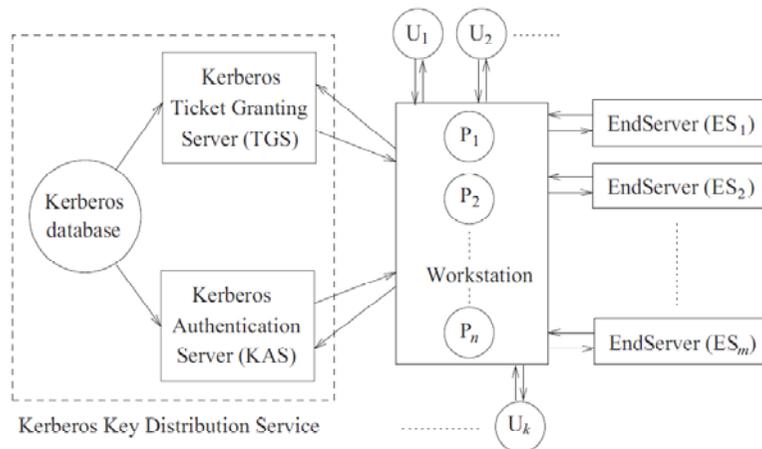
Generalità (1)

- Kerberos è traslitterazione latina del mostro della mitologia greca Κέρβερος
 - In italiano Cerbero
 - È un mostro a tre teste posto a guardia dell'Ade
- Aspetti chiave:
 - Tre teste
 - Impedire ai **non autorizzati** l'accesso all'Ade

Generalità (2)

- Kerberos è stato sviluppato al MIT all'interno del progetto Athena
- È un sistema di autenticazione distribuita che permette a un client di dimostrare la propria identità a un server senza inviare dati attraverso la rete
- È basato su un server centralizzato, la cui funzione è quella di garantire
 - l'autenticazione dei client per i server
 - l'autenticazione dei server per i client

Architettura (1)



Architettura (2)

- Comprende tre componenti logiche principali (tre teste)
 - **Workstation** su cui sono eseguiti i processi P_1, P_2, \dots, P_n , per conto dell'insieme di utenti U_1, U_2, \dots, U_k
 - **Kerberos Key Distribution Service** (o solo **Kerberos**), a sua volta costituito da
 - Kerberos Authentication Server – KAS
 - Ticket Granting Server - TGS
 - Un insieme di **End Server** ES_1, ES_2, \dots, ES_m

Architettura (3)

- All'interno della componente Kerberos c'è il DB delle chiavi dei client e dei server
 - Ogni chiave è la versione crittografata secondo DES della passwd dell'utente che accede a un client
- I nomi (ID) dei client/server sono le uniche informazioni identificative che circolano in chiaro sulla rete

Funzionamento (1)

- Affinché un client possa usare un servizio deve produrre al server che lo fornisce un **ticket** precedentemente ottenuto da Kerberos
 - Il ticket è una stringa di bit, crittografata mediante la chiave privata del server, che contiene l'identità del client che ha fatto la richiesta
 - Il server che ha ricevuto un ticket è certo dell'identità del client

Funzionamento (2)

- Per ogni servizio richiesto è necessario un ticket
 - All'inizio della sessione il client potrebbe **non** conoscere l'insieme di tutti i servizi che dovrà richiedere
- Per risolvere il problema il client potrebbe richiedere all'inizio **tutti i possibili ticket**, uno per ogni possibile servizio
 - Soluzione inefficiente

Funzionamento (3)

- Per superare il problema il client ottiene da Kerberos al momento del login un ticket a lungo termine chiamato **authentication ticket**
 - È usato dal client ogni volta che deve accedere a un servizio, inviandolo al TGS di Kerberos
 - TGS riconosce l'identità grazie al ticket di autenticazione e fornisce il ticket per il servizio

Funzionamento (4)

- Oltre al ticket, il client riceve da Kerberos anche una session key con cui crittografare le comunicazioni verso il server

Caratteristiche

- Il **server di autenticazione centralizzato** autentica
 - gli utenti nei confronti del server
 - il server nei confronti degli utenti
- È basato su un metodo di **crittografia convenzionale**
- Attualmente sono disponibili Version 4 & **Version 5** (RFC 1510)

Requisiti Soddisfatti

- **Secure** – impedire il masquerading
- **Reliable** – se Kerberos non è disponibile allora tutti i servizi che lo richiedono sono indisponibili
- **Transparent** – l'autenticazione trasparente all'utente
- **Scalable** – supportare numero indefinito di client/server

Simple Client Authentication (1)

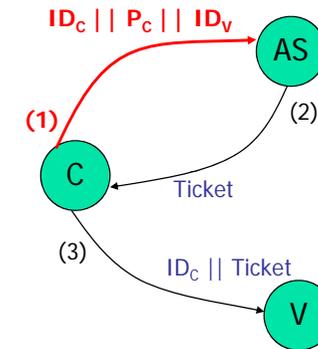
- In una rete senza autenticazione c'è il rischio di **impersonation**:
 - l'utente malintenzionato si presenta come se fosse un altro
- Soluzione 1
 - Il server ha bisogno di confermare l'identità di ogni client
 - Il sistema non è scalabile

Simple Client Authentication (2)

• Soluzione 2

- Usare un **authentication server** (AS)
- AS conosce le passwd di tutti gli utenti
- Condivide una chiave segreta con ogni server

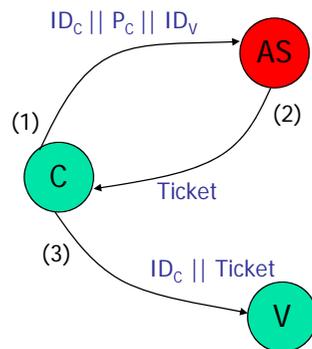
Simple Kerberos



$$\text{Ticket} = E_{K_V}[\text{ID}_C \parallel \text{AD}_C \parallel \text{ID}_V]$$

- User **logs on** and requests access to server V
- Client module requests user **password**
- Sends **message** to the AS with user's ID, server's ID and user's password

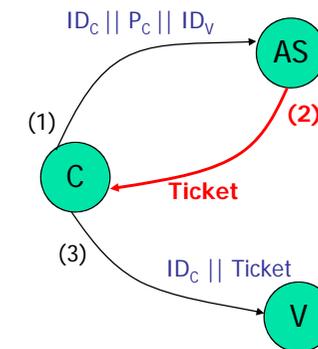
Simple Kerberos



$$\text{Ticket} = E_{K_V}[\text{ID}_C \parallel \text{AD}_C \parallel \text{ID}_V]$$

- AS checks database to see if user has supplied the proper password and is permitted to access server V
- If authentic, then creates a ticket containing user's ID, network address, and server's ID

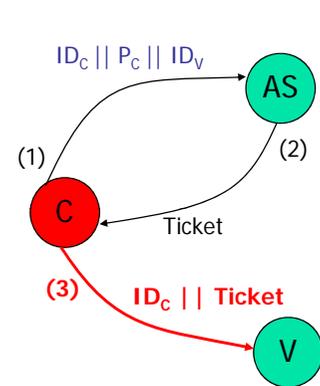
Simple Kerberos



$$\text{Ticket} = E_{K_V}[\text{ID}_C \parallel \text{AD}_C \parallel \text{ID}_V]$$

- **Ticket is encrypted** using the secret key shared by the AS and the server V
- **Send** ticket back to C
- Because the ticket is encrypted, it **cannot be altered** by C or an attacker

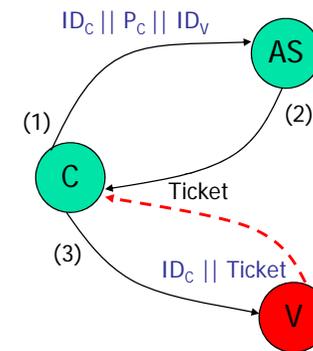
Simple Kerberos



$$Ticket = E_{K_V}[ID_C || AD_C || ID_V]$$

- C can now **apply to V for service**
- C sends message to V with **user's ID and the ticket**
- Server's ID_V is included so that the server can **verify** it has decrypted the ticket properly
- Ticket is encrypted to **prevent capture** or forgery

Simple Kerberos



$$Ticket = E_{K_V}[ID_C || AD_C || ID_V]$$

- V decrypts the ticket and **verifies** that the user ID_C in the ticket is the same as in the message
- AD_C in the message guarantees it came from **original requesting workstation**
- Finally, V **grants** the requested service

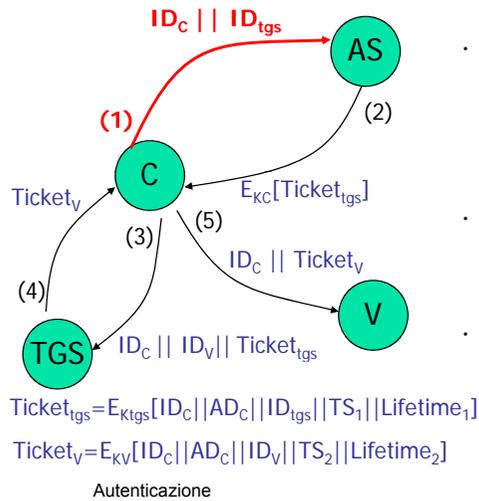
Problema

- Le password viaggiano in chiaro

Ticket Granting Server (TGS)

- Il TGS produce ticket per gli utenti già autenticati da AS
- L'utente dapprima richiede un ticket $Ticket_{tgs}$ a AS e lo salva nella workstation
- Il client richiede un servizio al TGS
- TGS produce il ticket $Ticket_V$ specifico per quel servizio
- Il client lo salva e lo usa quando necessario

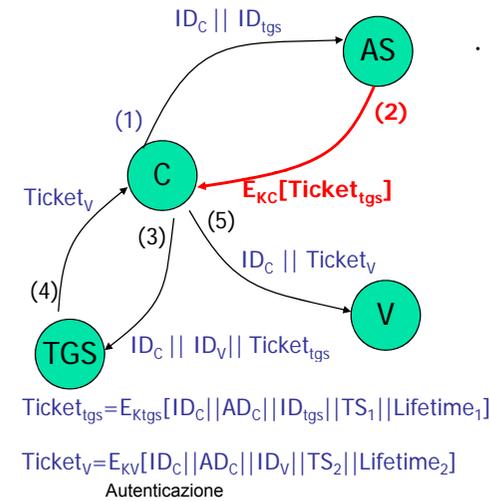
Simple Kerberos w/TGS



- Client requests a ticket granting ticket on behalf of user
- Sends user's ID and the ID of the TGS
- Indicates request for TGS service

25

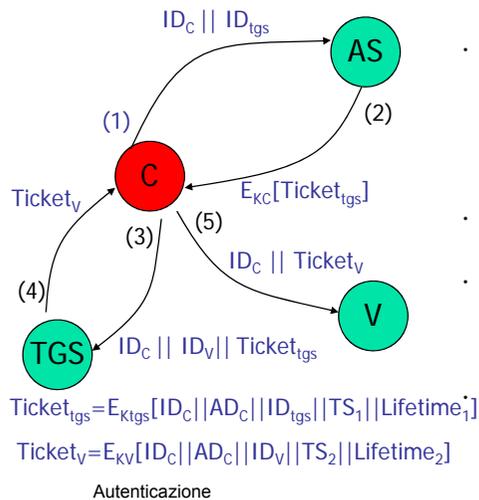
Simple Kerberos w/TGS



- AS responds with a ticket that is encrypted with a key from user's password

26

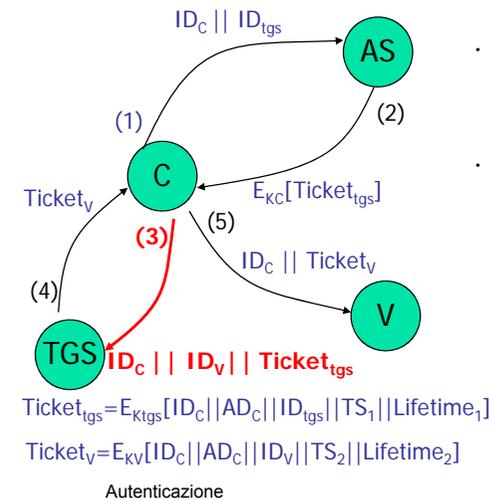
Simple Kerberos w/TGS



- Client prompts user for password, generates key and decrypts message
- Ticket is recovered!
- No need to transmit password in plaintext
- Ticket(tgs) is reusable

27

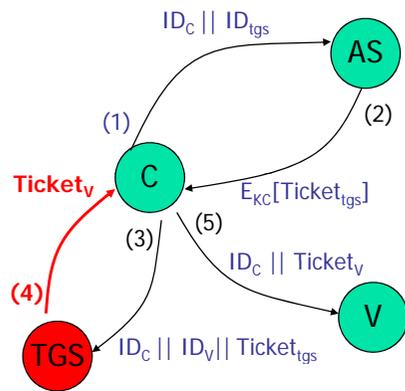
Simple Kerberos w/TGS



- Client requests a service granting ticket
- Sends message to TGS containing user's ID, ID of the desired service and the ticket granting ticket

28

Simple Kerberos w/TGS



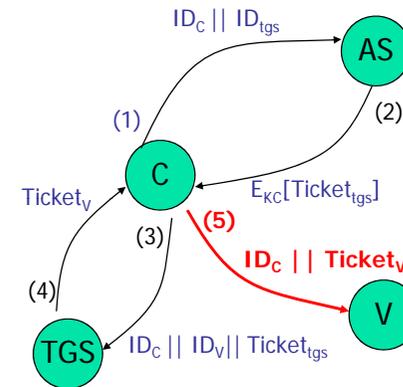
$Ticket_{TGS} = E_{K_{TGS}}[ID_C || AD_C || ID_{TGS} || TS_1 || Lifetime_1]$
 $Ticket_V = E_{K_V}[ID_C || AD_C || ID_V || TS_2 || Lifetime_2]$

Autenticazione

29

- TGS decrypts the incoming ticket and looks for presence of its ID
- Checks **lifetime** and **authenticates** the user
- If user permitted **access**, sends **service granting ticket**

Simple Kerberos w/TGS



$Ticket_{TGS} = E_{K_{TGS}}[ID_C || AD_C || ID_{TGS} || TS_1 || Lifetime_1]$
 $Ticket_V = E_{K_V}[ID_C || AD_C || ID_V || TS_2 || Lifetime_2]$

Autenticazione

30

- Client requests access to service on behalf of the user
- **Sends** user's ID and service granting ticket
- This can happen repeatedly without prompting for password

Version 4 Authentication

- Problemi:
 - **Lifetime** associate a un TGS
 - Se troppo corto, allora è necessario inserire più volte la passwd
 - Se troppo lungo, allora potrebbe essere intercettato
 - Autenticazione del server all'utente
 - Falsi server potrebbero farsi riconoscere come veri

Autenticazione

31

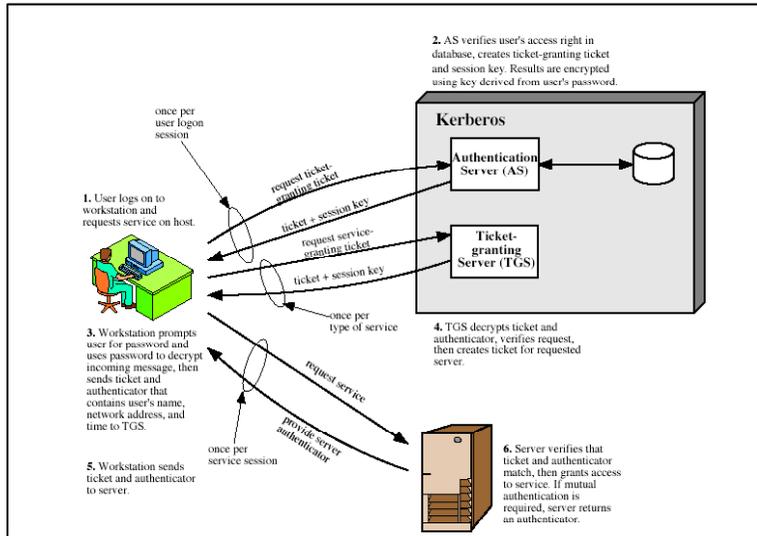
Version 4 Authentication

- **Session Key**
 - Usata per crittografare i messaggi, $K_{C,TGS}$ e $K_{C,V}$
- **Authenticator**
 - Crittografato con la session key
 - Contiene user ID indirizzo del client e un timestamp
 - È usato una sola volta: short lifetime

Autenticazione

32

Overview of Kerberos



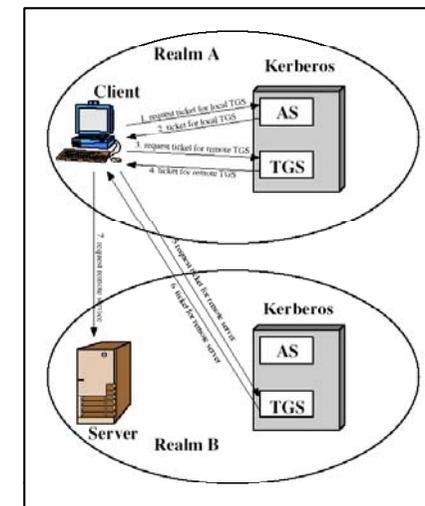
Kerberos Realms

- Per **realm** si intende una collezione di client e server all'interno di un'unica amministrazione, tali che Kerberos
 - conosce tutti gli user ID e le passwd
 - condivide una chiave segreta con ogni server

Kerberos Realms

- Gli utenti di un realm possono aver bisogno di accedere a servizi di un altro realm
- Kerberos in ogni realm condivide una secret key con gli altri
- Il server Kerberos in un realm deve fidarsi del Kerberos server degli altri realm

Richiesta Servizi in altro Realm



Kerberos Realms

- Problemi quando molti realms
- Dati N realms, ci devono essere $N(N-1)/2$ secure key tra I server Kerberos

Kerberos Version 5

- Non fa uso della crittografia DES
 - Può usare qualsiasi tecnica
- Lifetime dei ticket di lunghezza arbitraria
- Permette il forwarding dell'autenticazione

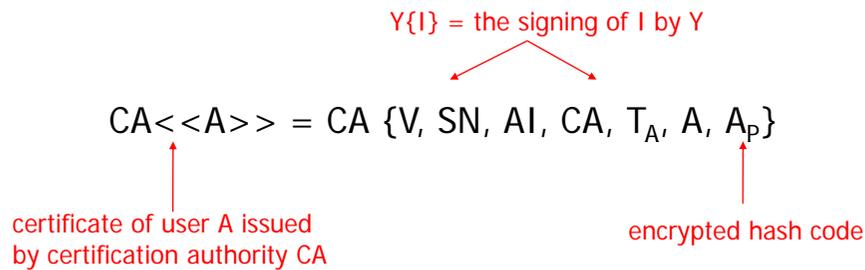
X.509 Authentication Service

- È parte della serie di specifiche X.500, che definisce il directory service
- Proposto nel 1988, V2-1993, V3-1995
- Basato su crittografia a chiave pubblica e firma digitale
- Definisce un framework per i servizi di autenticazione
- Prevede un repository di **public key certificates**
- Usato in S/MIME, IPsec, SSL e SET

Certificati

- Il certificato rappresenta il nucleo centrale dello schema X.509
- Ogni **certificato** contiene la **public key** di un utente ed è firmato con la **private key** di un'autorità fidata
- Un **certificato** è associato a ogni utente

Certificate Notation



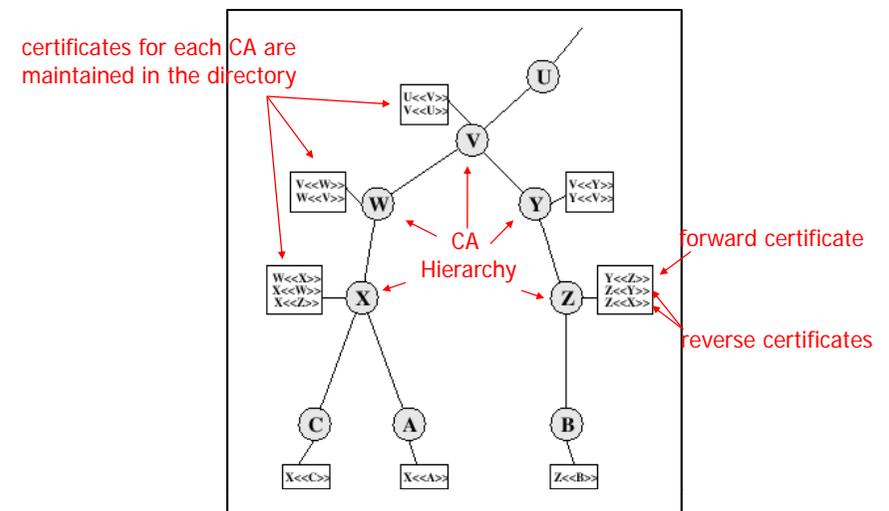
Certificato: Caratteristiche (1)

- A partire dalla public key del CA si deve poter risalire alla public key dell'utente certificato
- Solo il CA può modificare il certificato
- È memorizzato in una directory senza particolare protezione

Certificato: Caratteristiche (2)

- CA è **common trust** per tutti gli utenti certificati
- Ogni utente può trasmettere il proprio certificato ad altri
- Messaggi certificati sono sicuri dall'**eavesdropping**
- Non tutti gli utenti possono essere certificati alla stessa CA

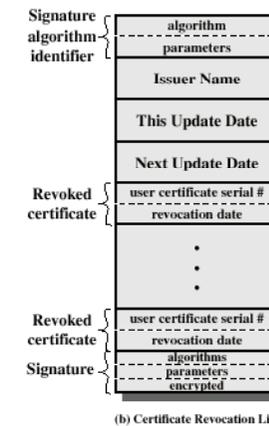
Chain of Certificates



Revoca di Certificati

- I certificati hanno un period di validità
- Possono essere **revoked** perché:
 - La chiave dell'utente è compromessa
 - L'utente non è più certificato presso la CA
 - Il certificato dalla CA è compromesso
- CA registra un elenco dei certificati revocati

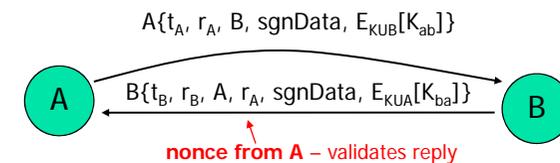
Certificate Revocation List (CRL)



Procedure di Autenticazione

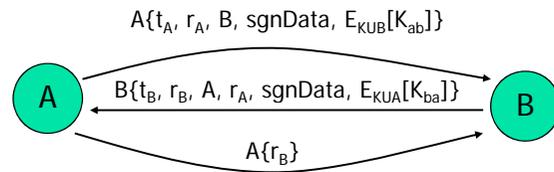
- X.509 prevede tre procedure che usano firme a chiave pubblica
- Si possono applicare a svariati contesti applicativi
- Si assume che ogni parte conosca la public key dell'altra

Two Way Authentication



- Establishes the identity of B and that the reply message was generated by B
- The message was intended for A
- Establishes the integrity and originality of the reply
- Both parties verify the identity of the other

Three Way Authentication



- Final message from A to B is included, with a signed copy of the nonce r_B
- No need for timestamps; each sides echoes back a nonce to prevent replay
- Used when no synchronized clocks available

Autenticazione

49

X.509 Version 3 Requirements

- Subject field needs to convey more information about the key owner
- Subject field needs more info for applications: IP address, URL
- Indicate security policy information (IPSec)
- Set constraints on certificate applicability – limit damage from faulty CA
- Identify separately different keys used by the same owner at different times – key life cycle management

Autenticazione

50

X.509 Version 3 Extensions

- Added optional extensions rather than fixed fields
- {extension id, criticality indicator, extension value}
- Three main categories:
 - Key and policy information – *EDI only*
 - Certificate subject and issuer attributes – *alternative names*
 - Certification Path Constraints - *restrictions*

Autenticazione

51