UNIK4250 Security in Distributed Systems
UNIK University Graduate Center
Spring 2012

Lecture 3

Key Distribution and User Authentication

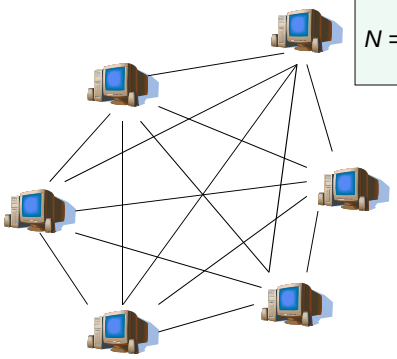
Leif Nilsen

UNIK

Outline

- What is key management?
- Kerberos?
- X.509
- PKI
- Identity federation

The key management challenge



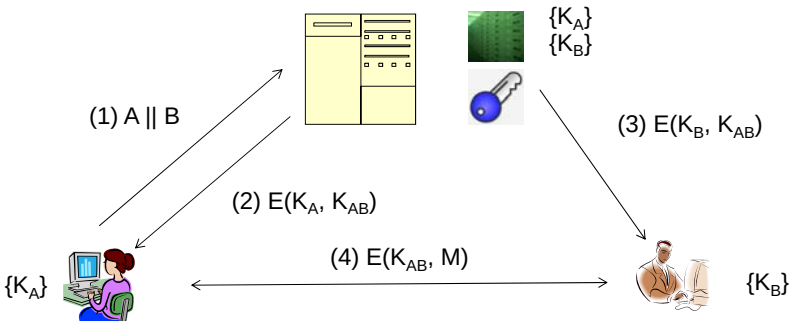
Number of keys in network with *n* nodes:

$N = (n\text{-}1) + (n\text{-}2) + \ldots\ldots\ldots + 1 = n(n\text{-}1)/2$

n	*N*
6	15
300	44 850
10.000	49 995 000

Possible solutions

- Group keys
- Key Distribution Centre (KDC)



Elements of key management

- Key planning
- Generate key
- Register key
- Distribute key
- Install key
- Key storage
- Key usage
- Key update
- Archive key
- Destroy key
- Key accounting

Kerberos

- Framework for user authentication and key management in distributed networks
- Development started at MIT as part of the Athena project (1983-1991)
- How to ensure that only authorized users were given access to a specific service
- Released in two versions 4 and 5
- Primary based on symmetric cryptographic techniques

" Kerberos is the three-headed dog that guarded the entrance to Hades" –Ancient Greek myth.

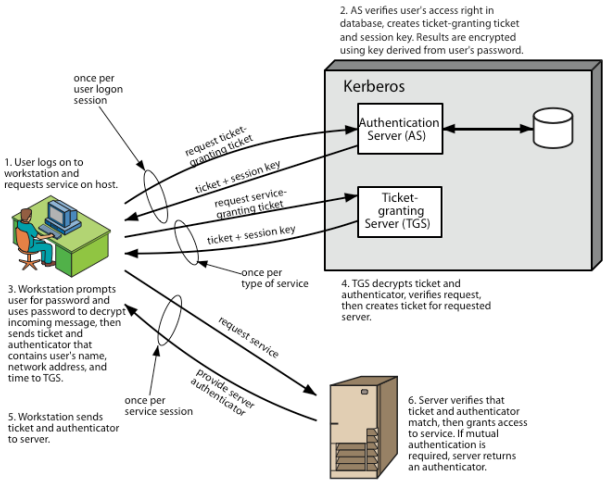
Kerberos Requirements

- Its first report identified requirements as:
  - secure
  - reliable
  - transparent
  - scalable
- Implemented using an authentication protocol based on Needham-Schroeder

Kerberos v4 Overview

- a basic third-party authentication scheme
- have an Authentication Server (AS)
  - users initially negotiate with AS to identify self
  - AS provides a non-corruptible authentication credential (ticket granting ticket TGT)
- have a Ticket Granting server (TGS)
  - users subsequently request access to other services from TGS on basis of users TGT
- using a complex protocol using DES

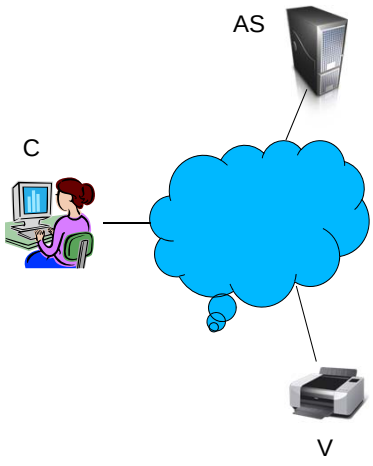
Kerberos 4 Overview



Protocol 1st version

(1) C → AS: $ID_C \| P_C \| ID_V$
(2) AS → C: Ticket
(3) C → V: $ID_C \| Ticket$

Ticket = $E(K_V,[ID_C \| AD_C \| ID_V])$



Protocol 2nd version

Once per user logon session:

(1) C → AS: $ID_C \| ID_{TGS}$

(2) AS → C: $E(K_C, Ticket_{TGS})$

Once per type of session:

(3) C → TGS: $ID_C \| ID_V \| Ticket_{TGS}$

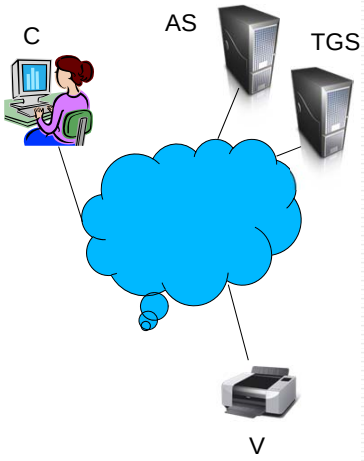
(4) TGS → C: $Ticket_V$

Once per service session:

(5) C → V: $Ticket_V$

$Ticket_{TGS} = E(K_{TGS}, [ID_C \| AD_C \| ID_{TGS} \| TS_1 \| Lifetime_1])$

$Ticket_V = E(K_V, [ID_C \| AD_C \| ID_V \| TS_2 \| Lifetime_2])$



Kerberos v4 Dialogue

(1) C → AS $ID_c \| ID_{tgs} \| TS_1$

(2) AS → C $E(K_c, [K_{c,tgs} \| ID_{tgs} \| TS_2 \| Lifetime_2 \| Ticket_{tgs}])$

$Ticket_{tgs} = E(K_{tgs}, [K_{c,tgs} \| ID_C \| AD_C \| ID_{tgs} \| TS_2 \| Lifetime_2])$

(a) Authentication Service Exchange to obtain ticket-granting ticket

(3) C → TGS $ID_v \| Ticket_{tgs} \| Authenticator_c$

(4) TGS → C $E(K_{c,tgs}, [K_{c,v} \| ID_v \| TS_4 \| Ticket_v])$

$Ticket_{tgs} = E(K_{tgs}, [K_{c,tgs} \| ID_C \| AD_C \| ID_{tgs} \| TS_2 \| Lifetime_2])$

$Ticket_v = E(K_v, [K_{c,v} \| ID_C \| AD_C \| ID_v \| TS_4 \| Lifetime_4])$

$Authenticator_c = E(K_{c,tgs}, [ID_C \| AD_C \| TS_3])$

(b) Ticket-Granting Service Exchange to obtain service-granting ticket

(5) C → V $Ticket_v \| Authenticator_c$

(6) V → C $E(K_{c,v}, [TS_5 + 1])$ (for mutual authentication)

$Ticket_v = E(K_v, [K_{c,v} \| ID_C \| AD_C \| ID_v \| TS_4 \| Lifetime_4])$

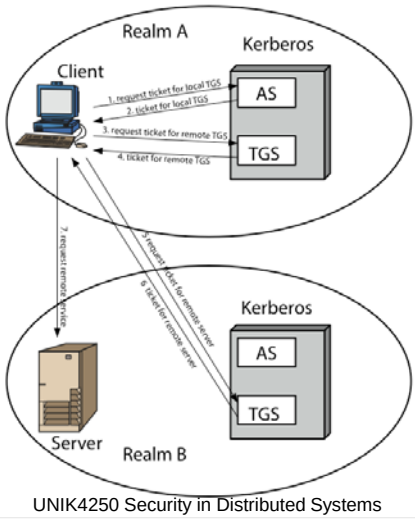
$Authenticator_c = E(K_{c,v}, [ID_C \| AD_C \| TS_5])$

(c) Client/Server Authentication Exchange to obtain service

Kerberos Realms

- a Kerberos environment consists of:
  - a Kerberos server
  - a number of clients, all registered with server
  - application servers, sharing keys with server
- this is termed a realm
  - typically a single administrative domain
- if have multiple realms, their Kerberos servers must share keys and trust

Kerberos Realms



Kerberos Version 5

- developed in mid 1990's
- specified as Internet standard RFC 1510
- provides improvements over v4
  - addresses environmental shortcomings
    - encryption alg, network protocol, byte order, ticket lifetime, authentication forwarding, interrealm auth
  - and technical deficiencies
    - double encryption, non-std mode of use, session keys, password attacks

Kerberos v5 Dialogue

(1) C → AS Options || ID_c || $Realm_c$ || ID_{tgs} || $Times$ || $Nonce_1$

(2) AS → C $Realm_c$ || ID_C || $Ticket_{tgs}$ || E(K_c, [$K_{c,tgs}$ || $Times$ || $Nonce_1$ || $Realm_{tgs}$ || ID_{tgs}])

$Ticket_{tgs}$ = E(K_{tgs}, [$Flags$ || $K_{c,tgs}$ || $Realm_c$ || ID_C || AD_C || $Times$])

(a) Authentication Service Exchange to obtain ticket-granting ticket

(3) C → TGS Options || ID_v || $Times$ || || $Nonce_2$ || $Ticket_{tgs}$ || $Authenticator_c$

(4) TGS → C $Realm_c$ || ID_C || $Ticket_v$ || E($K_{c,tgs}$, [$K_{c,v}$ || $Times$ || $Nonce_2$ || $Realm_v$ || ID_v])

$Ticket_{tgs}$ = E(K_{tgs}, [$Flags$ || $K_{c,tgs}$ || $Realm_c$ || ID_C || AD_C || $Times$])

$Ticket_v$ = E(K_v, [Flags || $K_{c,v}$ || $Realm_c$ || ID_C || AD_C || Times])

$Authenticator_c$ = E($K_{c,tgs}$, [ID_C || $Realm_c$ || TS_1])

(b) Ticket-Granting Service Exchange to obtain service-granting ticket

(5) C → V Options || $Ticket_v$ || $Authenticator_c$

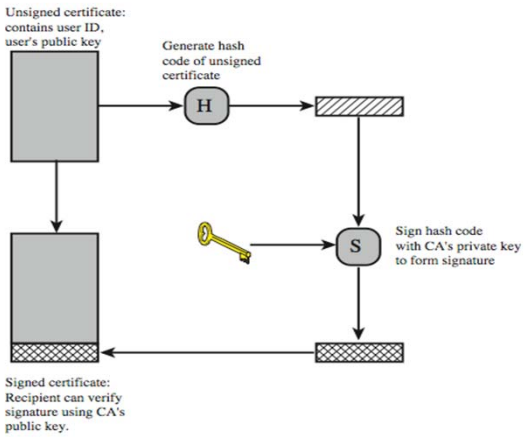
(6) V → C $E_{K_{c,v}}$ [TS_2 || Subkey || Seq#]

$Ticket_v$ = E(K_v, [Flags || $K_{c,v}$ || $Realm_c$ || ID_C || AD_C || Times])

$Authenticator_c$ = E($K_{c,v}$, [ID_C || $Realm_c$ || TS_2 || $Subkey$ || $Seq\#$])

(c) Client/Server Authentication Exchange to obtain service

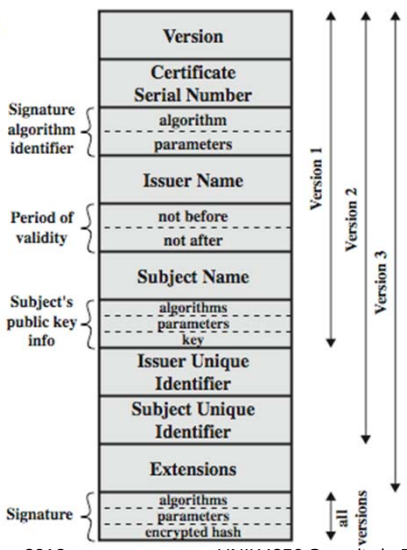
X.509 Certificate Use



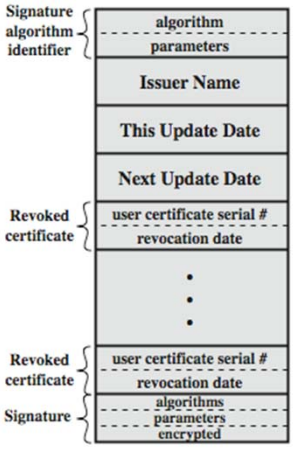
X.509 Certificates

- issued by a Certification Authority (CA), containing:
  - version V (1, 2, or 3)
  - serial number SN (unique within CA) identifying certificate
  - signature algorithm identifier AI
  - issuer X.500 name CA)
  - period of validity TA (from - to dates)
  - subject X.500 name A (name of owner)
  - subject public-key info Ap (algorithm, parameters, key)
  - issuer unique identifier (v2+)
  - subject unique identifier (v2+)
  - extension fields (v3)
  - signature (of hash of all fields in certificate)
- notation CA<<A>> denotes certificate for A signed by CA

X.509 Certificates



(a) X.509 Certificate



(b) Certificate Revocation List

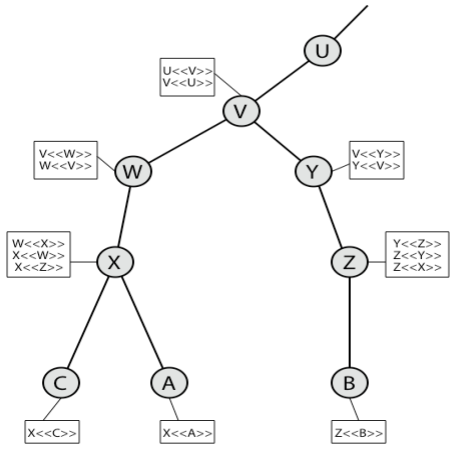
Obtaining a Certificate

- any user with access to CA can get any certificate from it
- only the CA can modify a certificate
- because cannot be forged, certificates can be placed in a public directory

CA Hierarchy

- if both users share a common CA then they are assumed to know its public key
- otherwise CA's must form a hierarchy
- use certificates linking members of hierarchy to validate other CA's
  - each CA has certificates for clients (forward) and parent (backward)
- each client trusts parents certificates
- enable verification of any certificate from one CA by users of all other CAs in hierarchy

CA Hierarchy Use



Certificate Revocation

- certificates have a period of validity
- may need to revoke before expiry, eg:
  1. user's private key is compromised
  2. user is no longer certified by this CA
  3. CA's certificate is compromised
- CA's maintain list of revoked certificates
  - the Certificate Revocation List (CRL)
- users should check certificates with CA's CRL

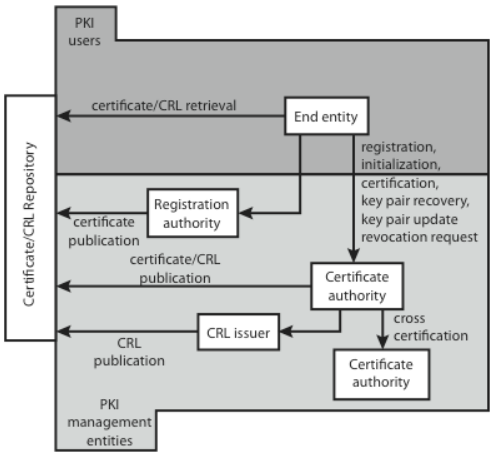
X.509 Version 3

- has been recognised that additional information is needed in a certificate
  - email/URL, policy details, usage constraints
- rather than explicitly naming new fields defined a general extension method
- extensions consist of:
  - extension identifier
  - criticality indicator
  - extension value

Certificate Extensions

- key and policy information
  - convey info about subject & issuer keys, plus indicators of certificate policy
- certificate subject and issuer attributes
  - support alternative names, in alternative formats for certificate subject and/or issuer
- certificate path constraints
  - allow constraints on use of certificates by other CA's

Public Key Infrastructure



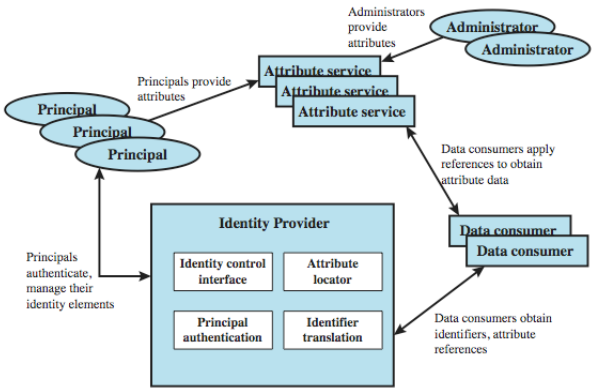
PKIX Management

- functions:
  - registration
  - initialization
  - certification
  - key pair recovery
  - key pair update
  - revocation request
  - cross certification
- protocols: CMP, CMC

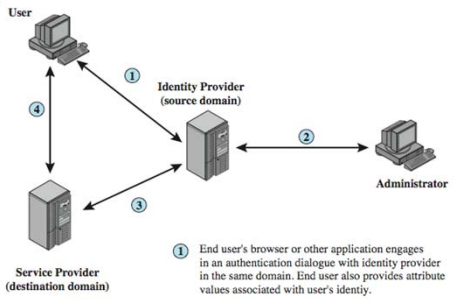
Federated Identity Management

- use of common identity management scheme
  - across multiple enterprises & numerous applications
  - supporting many thousands, even millions of users
- principal elements are:
  - authentication, authorization, accounting, provisioning, workflow automation, delegated administration, password synchronization, self-service password reset, federation
- Kerberos contains many of these elements

Identity Management



Identity Federation

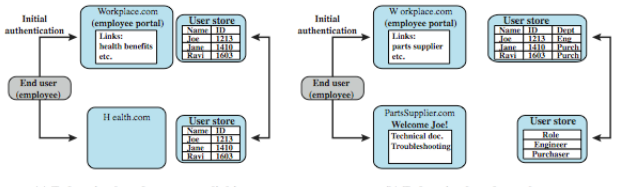


1. End user's browser or other application engages in an authentication dialogue with identity provider in the same domain. End user also provides attribute values associated with user's identiy.
2. Some attributes associated with an identity, such as allowable roles, may be provided by an administrator in the same domain.
3. A service provider in a remote domain, which the user wishes to access, obtains identity information, authentication information, and associated attributes from the identity provider in the source domain.
4. Service provider opens session with remote user and enforces access control restrictions based on user's identity and attributes.

Standards Used

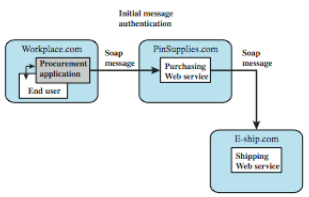
- Security Assertion Markup Language (SAML)
  - XML-based language for exchange of security information between online business partners
- part of OASIS (Organization for the Advancement of Structured Information Standards) standards for federated identity management
  - e.g. WS-Federation for browser-based federation
- need a few mature industry standards

Federated Identity Examples



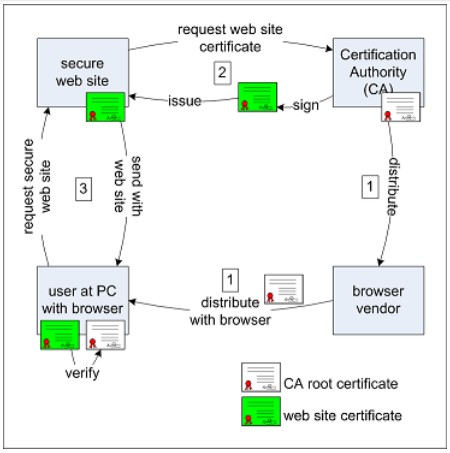
(a) Federation based on account linking

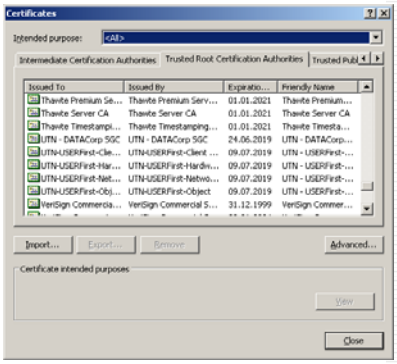
(b) Federation based on roles



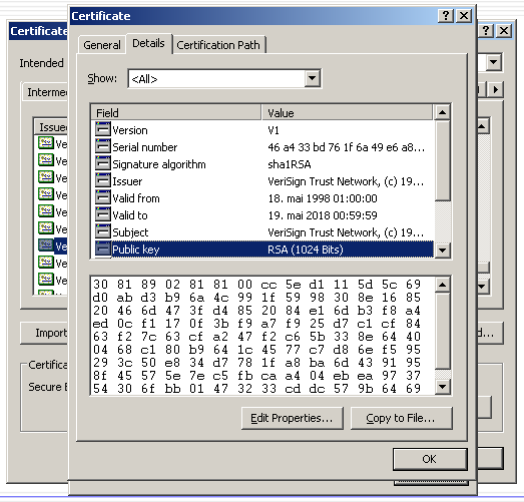
(b) Chained Web Services

Net shopping model

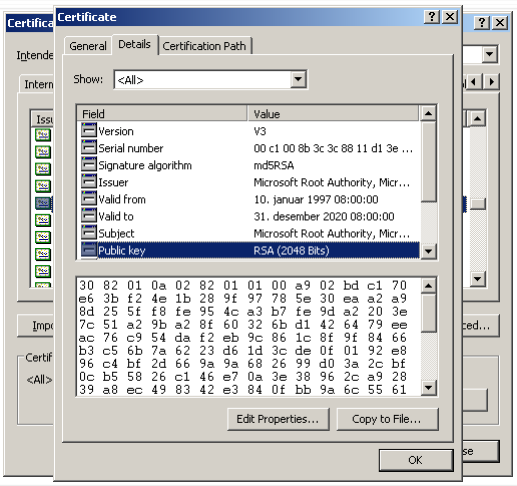




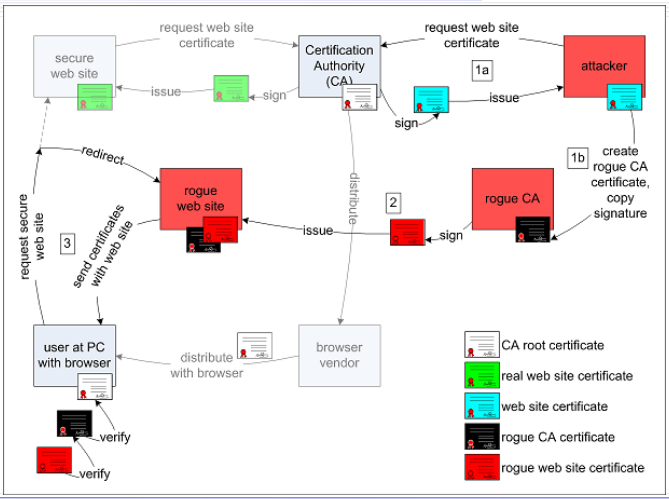
Typical certificate



Obs.....



Attacks based on MD5-collisions



Summary

- have considered:
  - Models for key management
  - Kerberos
  - X.509
  - PKI
  - Identity management