Authentication Applications

CSCI 454/554

- will consider authentication functions
- developed to support application-level authentication & digital signatures
- Kerberos – a symmetric-key based authentication service
- X.509 directory authentication service
Kerberos

- developed from Project Athena at MIT
- provides centralised symmetric-key third-party authentication in a distributed network
  - allows users access to services distributed through network
  - without needing to trust all workstations
  - rather all trust a central authentication server
- two versions in use: 4 & 5

Kerberos Requirements

- first published report identified its requirements as:
  - security
  - reliability
  - transparency
  - scalability
- implemented using an authentication protocol based on Needham-Schroeder
Needham-Schroeder Protocol

A \{K_a\}, B \{K_b\}, and KDC \{K_a, K_b, K_c, \ldots\}

1. A \rightarrow KDC: A, B, N_1

2. KDC \rightarrow A: \{N_1, B, K_s, \{K_s, A\}K_b\}K_a

3. A \rightarrow B: \{K_s, A\}K_b \leftarrow Ticket

4. B \rightarrow A: \{N_2\}K_s

5. A \rightarrow B: \{f(N_2)\}K_s

Kerberos 4 Overview

- Authentication Server (AS)
  - users initially negotiate with AS to identify self
  - AS provides a non-corruptible authentication credential ("session key" and ticket-granting ticket TGT)
- Ticket Granting Server (TGS)
  - users subsequently request access to other services from TGS on basis of users TGT
- AS & TGS play the role of KDC
Why AS&TGS, not KDC?

- user’s master key is derived from its password
- it is not secure to keep master key at the user side all the time
- master key is used to get a “session” key from AS, then is discarded at the user side
- TGT : {“session key”, user name, timestamp, and TGT’s expiration time}K_{tgs}
- user communicates with TGS for remote services

How it works

- Login: obtain a session key and TGT from AS
  - Alice —> AS: Alice, TGS
    - AS creates “session key” Sa and {TGT}K_{tgs}
  - AS —> Alice: {Sa, {TGT}K_{tgs}, T_{stamp}}K_A
    1. Alice uses password to derive KA
    2. decrypt { “session key”, {TGT}K_{tgs} }KA
    3. discard KA
How it works (Con’t)

- **Remote access:**
  - Alice → TGS: \{TGT\}_Kgs || name of server (Bob) || authenticator
    - Authenticator: \{A, network address, time of day\} encrypted with “session key” \text{Sa}
  - TGS → Alice: \{B, Ks, ticket to B, T\text{stamp}\}_Sa
    - Ticket to access server B: \{Ks, A, T\text{stamp}, Life\}_KB
  - Alice → Bob: \{Ks, A, T\text{stamp}, Life\}_KB || Authenticator
  - Bob → Alice: \{T\text{stamp}+1\}_Ks \{A, net-address, time of day\}_Ks

### Kerberos 4 Overview

1. User logs on to workstation and requests services on host.
2. AS verifies user’s access right in database, creates ticket-granting ticket and session key. Result are encrypted using key derived from user's password.
3. Workstation requests user for password and uses protocol to decrypt incoming message; then sends ticket and user's name to TGS.
   - TGS decrypts ticket and authenticates, verifies request, then creates ticket for requested service.
4. Workstation sends ticket and authenticator to server.
5. User logs out from host.

\begin{itemize}
  \item Kerberos
  \item Authentication Server (AS)
  \item Ticket-granting Server (TGS)
  \item Kerberos
\end{itemize}
More words about tickets

- Ticket-granting ticket is obtained as part of login process
- If login session lasts longer than lifetime of TGT
  - Run `kinit` to obtain a new TGT
- All tickets are automatically destroyed when user logs out

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 each other
Kerberos Version 5

- developed in mid 1990’s
- provides improvements over v4
- Problems in version 4
  - environmental shortcomings
    - encryption alg, network protocol, byte order, ticket lifetime, authentication forwarding, inter-realm authentication
  - technical deficiencies
    - double encryption, non-std mode of use, session keys, password attacks
- specified as Internet standard RFC 1510
X.509 Authentication Service

- part of CCITT X.500 directory service standards
  - distributed servers maintaining info about users
- defines framework for authentication services
  - directory may store public-key certificates
  - with public key of user
  - signed by certification authority
- also defines authentication protocols
- uses public-key crypto & digital signatures
  - algorithms not standardised, but RSA recommended

X.509 Certificates

- issued by a Certification Authority (CA), containing:
  - version (1, 2, or 3)
  - serial number (unique within CA) identifying certificate
  - signature algorithm identifier
  - issuer X.500 name (CA)
  - period of validity (from - to dates)
  - subject X.500 name (name of owner)
  - subject public-key info (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
Figure 14.3 Public-Key Certificate Use
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<<A>>: the certificate of user A issued by certification authority CA

CA Hierarchy

- Scalability issue: a very large user group
  - CA hierarchy (tree)
- A node (X) of hierarchy is a CA, its parent node is another CA (Y)
  - each CA has forward certificate: Y<<X>> and reverse certificate X<<Y>>
- each user trusts parent CA’s certificates
- enable verification of any certificate from one CA by users of all other CAs in hierarchy
CA Hierarchy Use

Path of A acquiring the certificate of B:
X<<W>>W<<V>>V<<Y>>Y<<Z>>Z<<B>>

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 certs 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
- each extension consist of:
  - extension identifier
  - criticality indicator: if can be safely ignored
  - extension value

Certificate Extensions

- key and policy information
  - convey additional 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

Summary

- have considered:
  - Kerberos trusted key server system
  - X.509 authentication and certificates