KEY DISTRIBUTION AND USER AUTHENTICATION

Date 2016/03/28

Alnazif Mohammed Alnour

(alnadeef@yahoo.com)
Key Management and Distribution

No Singhalese, whether man or woman, would venture out of the house without a bunch of keys in his hand, for without such a talisman he would fear that some devil might take advantage of his weak state to slip into his body.

—The Golden Bough, Sir James George Frazer
Key Management and Distribution

- topics of cryptographic key management / key distribution are complex
- cryptographic, protocol, & management issues
- symmetric schemes require both parties to share a common secret key
- public key schemes require parties to acquire valid public keys
- have concerns with doing both
Key Distribution

- symmetric schemes require both parties to share a common secret key
- issue is how to securely distribute this key
- whilst protecting it from others
- frequent key changes can be desirable
- often secure system failure due to a break in the key distribution scheme
Key Distribution

given parties A and B have various key distribution alternatives:

1. A can select key and physically deliver to B
2. third party can select & deliver key to A & B
3. if A & B have communicated previously can use previous key to encrypt a new key
4. if A & B have secure communications with a third party C, C can relay key between A & B
Kerberos

- trusted key server system from MIT
- provides centralised private-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

- 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 v4 Dialogue

(1) $C \rightarrow AS \quad ID_c \parallel ID_{tgs} \parallel TS_1$

(2) $AS \rightarrow C \quad E(K_c, [K_{c,tgs} \parallel ID_{tgs} \parallel TS_2 \parallel Lifetime_2 \parallel Ticket_{tgs}])$
   
   $Ticket_{tgs} = E(K_{tgs}, [ID_c \parallel AD_c \parallel ID_{tgs} \parallel TS_2 \parallel Lifetime_2])$

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

(3) $C \rightarrow TGS \quad ID_v \parallel Ticket_{tgs} \parallel Authenticator_c$

(4) $TGS \rightarrow C \quad E(K_{c,tgs}, [K_{c,v} \parallel ID_v \parallel TS_4 \parallel Ticket_v])$
   
   $Ticket_{tgs} = E(K_{tgs}, [K_{c,tgs} \parallel ID_c \parallel AD_c \parallel ID_{tgs} \parallel TS_2 \parallel Lifetime_2])$

   $Ticket_v = E(K_v, [ID_c \parallel AD_c \parallel ID_v \parallel TS_4 \parallel Lifetime_4])$

   $Authenticator_c = E(K_{c,tgs}, [ID_c \parallel AD_c \parallel TS_3])$

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

(5) $C \rightarrow V \quad Ticket_v \parallel Authenticator_c$

(6) $V \rightarrow C \quad E(K_{c,v}, [TS_5 + 1])$ (for mutual authentication)
   
   $Ticket_v = E(K_v, [ID_c \parallel AD_c \parallel ID_v \parallel TS_4 \parallel Lifetime_4])$

   $Authenticator_c = E(K_{c,v}, [ID_c \parallel AD_c \parallel TS_3])$

   (c) Client/Server Authentication Exchange to obtain service
Kerberos 4 Overview

1. User logs on to workstation and requests service on host.
2. AS verifies user's access right in database, creates ticket-granting ticket and session key. Results are encrypted using key derived from user's password.
3. Workstation prompts user for password and uses password to decrypt incoming message, then sends ticket and authenticator that contains user's name, network address, and time to TGS.
4. TGS decrypts ticket and authenticator, verifies request, then creates ticket for requested server.
5. Workstation sends ticket and authenticator to server.
6. Server verifies that ticket and authenticator match, then grants access to service. If mutual authentication is required, server returns an authenticator.
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

1. Request ticket for local TGS
2. Ticket for local TGS
3. Request ticket for remote TGS
4. Ticket for remote TGS
7. Request remote service
6. Ticket for remote service
5. Request remote service
4. Ticket for remote TGS
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

Unsigned certificate:
contains user ID,
user's public key

Generate hash code of unsigned certificate

H

Sign hash code
with CA's private key
to form signature

S

Signed certificate:
Recipient can verify
signature using CA's
public key.
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 identity.

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
Thank you