Trusted Intermediaries

- Problem: authentication for large networks
- Solution #1
  - Key Distribution Center (KDC)
  - Based on secret key cryptography
- Solution #2
  - Public Key Infrastructure (PKI)
  - Based on public key cryptography

KDC-based Trusted Intermediaries

- Representative system
  - Kerberos
Background

- The authentication problem
  - Assume an open distributed environment in which users at workstations wish to access services on servers distributed throughout the network.
  - Restrict access to authorized users and to be able to authenticate requests for service.

Can we rely on workstation for authentication service?

- Three threats
  - A user may gain access to a particular workstation and pretend to be another user operating from that workstation.
  - A user may alter the network address of a workstation so that the requests sent from the altered workstation appear to come from the impersonated workstation.
  - A user may eavesdrop on exchanges and use a replay attack to gain entrance to a server or to disrupt operations.
Authentication Service Provided by Kerberos

- A centralized authentication service
  - Authenticate users to services
  - Authenticate services to users
  - Servers are relieved of the burden of maintaining authentication information.

- Facts about Kerberos
  - Rely exclusively on conventional encryption.
    - Public key based Kerberos has been considered.
  - Stateless
    - Kerberos server doesn’t maintain the state information about any entities being authenticated.

Requirements for Kerberos

- Secure
  - A network eavesdropper should not be able to obtain the necessary to impersonate a user.

- Reliable
  - Kerberos should be highly available and should employ a distributed server architecture.

- Transparent
  - The user shouldn’t be aware that authentication is taking place.

- Scalable
  - The system should be capable of supporting large numbers of clients and servers.

The Kerberos Protocol

- Outline of the introduction to the Kerberos protocol
  - A simple authentication protocol
  - A more secure authentication protocol
  - Kerberos Version 4 authentication protocol
A Simple Authentication Protocol

- Use an authentication server (AS)
- Basic idea: use a ticket to authenticate a user to a server.

Protocol
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]

A Simple Authentication Protocol (Cont’d)

- Advantages
  - A centralized authentication service

- Weaknesses
  - A user needs to enter a password for every different service.
  - Password is transmitted in plaintext.

A More Secure Authentication Protocol

- A new server: ticket-granting server (TGS)
- Protocol
  - Once per user logon session
    1) C→AS: ID_C || ID_TGS
    2) AS→C: E_K_C[Ticket_TGS]
  - Once per type of service
    3) C→TGS: ID_C || ID_V || Ticket_TGS
    4) TGS→C: Ticket_TGS
  - Once per service session
    5) C→V: ID_C || Ticket_TGS
    - Ticket_TGS = E_K_TGS[ID_C || AD_C || ID_TGS || TS_1 || lifetime_1]
    - Ticket_V = E_K_V[ID_C || AD_V || ID_V || TS_2 || lifetime_2]
A More Secure Authentication Protocol (Cont’d)

- Ticket-granting ticket (TGT): Ticket<sub>tgs</sub>.
- Service-granting ticket: Ticket<sub>v</sub>.
- Weaknesses
  - Replay attack: No authentication of the valid ownership of the tickets.
  - No authentication of the servers.

- What are the components in the tickets?
- Why do we have them?

Kerberos Version 4 Protocol

- Basic idea to address the previous weaknesses
  - Session key
    - Authentication of the valid ownership of the tickets
    - Provide authentication of servers.

Kerberos Version 4 Protocol (Cont’d)

- Authentication Service Exchange: to obtain ticket-granting ticket
  1) C→AS: ID<sub>C</sub> || ID<sub>adm</sub> || TS<sub>1</sub>
  2) AS→C: E<sub>K<sub>tgs</sub></sub> [K<sub>C</sub> || ID<sub>adm</sub> || TS<sub>2</sub> || Lifetime<sub>2</sub> || Ticket<sub>tgs</sub>]
  - Ticket<sub>tgs</sub> = E<sub>K<sub>tgs</sub></sub> [K<sub>C</sub> || ID<sub>C</sub> || AD<sub>C</sub> || ID<sub>adm</sub> || TS<sub>1</sub> || Lifetime<sub>1</sub>]
Kerberos Version 4 Protocol (Cont’d)

- Ticket-Granting Service Exchange: to obtain service-granting ticket
  3) $C \rightarrow TGS$: ID$_C$ || Ticket$_{tgs}$ || Authenticator$_C$
  4) $TGS \rightarrow C$: $E_{K_{C,TGS}}[K_{C,V} || ID_C || AD_C || ID_{tgs} || TS_2 || Lifetime_2]$
    - Ticket$_{tgs} = E_{K_{TGS}}[K_{C,TGS} || ID_C || AD_C || ID_{tgs} || TS_4 || Lifetime_4]$
    - Authenticator$_C = E_{K_{C,TGS}}[ID_C || AD_C || TS_3]$

Kerberos Version 4 Protocol (Cont’d)

- Client/Server Authentication Exchange: to obtain service
  5) $C \rightarrow V$: Ticket$_V$ || Authenticator$_C$
  6) $V \rightarrow C$: $E_{K_{C,V}}[TS_5 + 1]$
    - Ticket$_V = E_{K_{C,V}}[K_{C,V} || ID_C || AD_C || ID_V || TS_4 || Lifetime_4]$
    - Authenticator$_C = E_{K_{C,V}}[ID_C || AD_C || TS_3]$

The Whole Picture
Kerberos Deployment

- The Kerberos server must have the user ID and hashed password of all participating users in its database.
- The Kerberos server must share a secret key with each server.
- Kerberos are “physically” secured
- Kerberos libraries are distributed on all nodes with users, applications, and other Kerberos-controlled resources

Replicated Kerberos

- Multiple replica of Kerberos - availability and performance
- Keeping Kerberos databases consistent
  - Single master Kerberos as the point of direct update to principals’ database entries
  - Updated database is downloaded from the master to all replica Kerberos
  - Periodic download or on-demand

Kerberos Realms and Multiple Kerberi

- Kerberos realm
  - A full-service Kerberos environment consists of a Kerberos server, a number of clients, and a number of application servers
- Inter-realm authentication
  - The Kerberos server in each interoperating realm shares a secret key with the server in the other realm. The two Kerberos servers are registered with each other.
Inter-realm Authentication

1. Request ticket for local TGS
2. Ticket for local TGS
3. Request ticket for remote TGS
4. Ticket for remote TGS
5. Request ticket for remote server
6. Ticket for remote server
7. Ticket for remote server
8. Remote server authenticator

Public Key Infrastructure (PKI)

What Is PKI

- Informally, the infrastructure supporting the use of public key cryptography.
- A PKI consists of
  - Certificate Authority (CA)
  - Certificates
  - A repository for retrieving certificates
  - A method of revoking certificates
  - A method of evaluating a chain of certificates from known public keys to the target name
Certification Authorities (CA)

- A CA is a trusted node that maintains the public keys for all nodes (Each node maintains its own private key)

![Diagram of Certification Authorities (CA)]

If a new node is inserted in the network, only that new node and the CA need to be configured with the public key for that node.

Certificates

- A CA is involved in authenticating users’ public keys by generating certificates
- A certificate is a signed message vouching that a particular name goes with a particular public key
- Example:
  1. [Alice’s public key is 876234]_{CA} & [Alice’s public key is 876234]_{Carol}
  2. [Carol’s public key is 676554]_{CA} & [Alice’s public key is 876234]_{Carol}

- Knowing the CA’s public key, users can verify the certificate and authenticate Alice’s public key.

Certificates

- Certificates can hold expiration date and time
  - Alice keeps the same certificate as long as she has the same public key and the certificate does not expire
  - Alice can append the certificate to her messages so that others know for sure her public key.
CA Advantages

1. The CA does not need to be online. [Why?]

2. If a CA crashes, then nodes that already have their certificates can still operate.

3. Certificates are not security sensitive (in terms of confidentiality).
   - Can a compromised CA decrypt a conversation between two parties?
   - Can a compromised CA fool Alice into accepting an incorrect public key for Bob, and then impersonate Bob to Alice?

CA Problems

- What if Alice is given a certificate with an expiration time and then is revoked (fired) from the system?
  - Alice can still use her certificate till the expiration time expires.
  - What kind of harm can this do?
  - Alice can still exchange messages with Bob using her unexpired certificate.

- Solution:
  - Maintain a Certificate Revocation List (CRL) at the CA. A Certificate is valid if (1) it has a valid CA signature, (2) has not expired, and (3) is not listed in the CA’s CRL list.

Terminology

- A CA signing a certificate for Alice’s public key
  - CA → issuer Alice → subject
- Alice wants to find the Bob’s public key
  - Bob → target
- Anyone with a public key is a principal
- Alice is verifying a certificate (or a chain of certificates)
  - Alice → verifier
- Trust anchor → A CA with a trusted public key
### PKI Models

1. **Monopoly model**
2. Monopoly + RA
3. Delegated CAs
4. Oligarchy model
5. Anarchy model
6. Name constraints
7. Top-down with name constraints
8. Bottom-up with name constraints

### Monopoly Model

- One CA universally trusted by everyone
- Everyone must get certificates from this CA
- The public key to this organization is the only PKI trust anchor and is embedded in all software and hardware

### Problems

1. There is NO universally trusted organization
2. Monopoly control. CA could charge any fees.
3. Once deployed, it is hard to switch to a different CA
4. Entire world’s security relies on this CA
5. Inconvenient.
PKI Models

1. Monopoly model
2. Monopoly + RA
3. Delegated CAs
4. Oligarchy model
5. Anarchy model
6. Name constraints
7. Top-down with name constraints
8. Bottom-up with name constraints

Monopoly + Registration Authorities (RA)

- RAs are affiliated with the single CA and are trusted by this CA.
- RAs check identities and provide the CA with relevant information (identity and public key information) to generate certificates.
- More convenient (more places to be certified).
- Still a monopoly. All the monopoly problems still hold.
Delegated CAs

- The trust anchor (known CA) issues certificates to other CAs (delegated CAs) vouching for their trustworthiness as CAs.
- Users can obtain their certificates from delegated CAs instead of the trust anchor CA.
- Example:
  - [Carol’s public key is 676554]_{Ted} & [Alice’s public key is 876234]_{Carol}
  - Ted: trust anchor CA & Carol: delegated CA

PKI Models

1. Monopoly model
2. Monopoly + RA
3. Delegated CAs
4. Oligarchy model
5. Anarchy model
6. Name constraints
7. Top-down with name constraints
8. Bottom-up with name constraints

Oligarchy Model

- A few trusted CAs and a certificate issued by any one of them is accepted
- Competition between CAs is good
- Problems: Not as secure as the monopoly case
  - Need to protect more CAs (instead of only one)
  - Might be easier to trick a naïve user by inserting a bogus trust anchor in the list of trusted CAs
  - It is hard to examine the set of trust anchors and determine whether some has modified the set
PKI Models

1. Monopoly model
2. Monopoly + RA
3. Delegated CAs
4. Oligarchy model
5. Anarchy model
6. Name constraints
7. Top-down with name constraints
8. Bottom-up with name constraints

Anarchy Model (Web of Trust)

- Fully distributed approach. No CA or list of CA provided to the users. Anyone can sign certificates for anyone else.
- Each user is responsible for configuring some trust anchors (provide his own certificates for them).
- A database maintains these certificates.
- Unworkable on a large scale (Why?).

PKI Models

1. Monopoly model
2. Monopoly + RA
3. Delegated CAs
4. Oligarchy model
5. Anarchy model
6. Name constraints
7. Top-down with name constraints
8. Bottom-up with name constraints
Name Constraints

- A CA is responsible for certifying users in his domain only
  - NCSU CA certifies NCSU students
- Provides complete autonomy
- CAs need to be able to identify each other. How?

PKI Models

1. Monopoly model
2. Monopoly + RA
3. Delegated CAs
4. Oligarchy model
5. Anarchy model
6. Name constraints
7. Top-down with name constraints
8. Bottom-up with name constraints

Top-Down with Name Constraints

- Everyone agrees on a root organization and the root CA delegates to other CA. (A centralized trust anchor (CA) + delegated CAs).
- To get a certificate, contact the root.
- You will be redirected to an appropriate delegated CA.
- Delegated CAs can only issue certificates for users in their domain.
PKI Models

1. Monopoly model
2. Monopoly + RA
3. Delegated CAs
4. Oligarchy model
5. Anarchy model
6. Name constraints
7. Top-down with name constraints
8. Bottom-up with name constraints

Bottom-Up with Name Constraints

- Assumes a hierarchical name space.
  - Similar to Internet domain names.
- Each organization maintains its own CA, and CAs link to others.
  - Similar to DNS tree hierarchy but also cross-links (cross certificates) are allowed (Forest hierarchy).
  - A parent certifies its children and children certify their parent.
- The hierarchy is traversed in a bottom-up fashion.
  - Follow up-links until you encounter an ancestor of the target, then follow at most one cross-link, and then follow down-links from there.

How can A/C/Y verify the certificate of B/Y/Z/C?
How can B/Y/Z/C verify the certificate of A/C/Y?
Advantages

1. Easy to navigate the hierarchy (similar to DNS).
2. No monopoly.
3. Replacing keys is reasonably easy.
4. Can be deployed in any organization without help from the rest of the world.
5. Authentication between users in the same organization does not need to go outside the organization.

Certificate Revocation

- Certificates for public keys (Campus IDs) might need to be revoked from the system
  - Someone is fired
  - Someone graduated
  - Someone’s certificate (card) is stolen

Certificate Revocation

- Certificates typically have an associated expiration time
  - Typically in the order of months (too long to wait if it needs to be revoked)

- Solutions:
  - Maintain a Certificate Revocation List (CRL)
  - A CRL is issued periodically by the CA and contains all the revoked certificates
  - Each transaction is checked against the CRL
CRLs

1. Why are CRLs issued periodically even if no certificates are revoked?

2. How frequent should CRLs be issued?

3. If a CRL is maintained, why associate an expiration time with certificates?

Delta CRL

- Certificates (1) may be huge, and (2) need to be issued periodically
- A Delta CRL includes lists changes from the last complete CRL
- Delta CRLs may be issued periodically (frequently) and full CRLs are issued less frequently

On-line Revocation Servers (OLRS)

- An OLRS is a system that can be queried over the network for the revocation status of individual certificates
- An OLRS maintains the full CRL list
- What if someone impersonates an OLRS?
  ----
  - Solution?
  ----
Good-lists vs. Bad-lists

- How about maintaining a list of valid certificates in the CRL instead of the revoked certificates?
- Is this more secure? Why?

- **Problems:**
  1. A good list is likely to be much larger than the bad list (worse performance)
  2. Organizations might not want to maintain its list of valid certificates public.

  **Solution:** The good-list can maintain only hashes of the valid certificates