Trusted Intermediaries

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

Outline

- Introduction
- Version 4: Basics
- Additional Capabilities
- Version 5 and Inter-Realm Authentication

Goals of Kerberos

1. User ↔ server mutual authentication
2. Users should only need to authenticate once to obtain services from multiple servers
3. Should scale to large numbers of users and servers
   - makes use of a Key Distribution Center so servers don’t need to store information about users

Introduction
Some Properties

• Kerberos uses only secret key (symmetric) encryption
  – originally, only DES, but now 3DES and AES as well
• A stateless protocol
  – KDCs do not need to remember what messages have previously been generated or exchanged
  – the state of the protocol negotiation is contained in the message contents

Example Scenario

• Alice wants to make use of services from X, contacts the KDC to authenticate, gets ticket to present to X
• Bob wants to make use of services from X and Y, contacts the KDC, gets tickets to present to X and Y

The KDC

• Infrastructure needed (KDC components)
  1. the database of user information (IDs, password hash, shared secret key, etc.)
  2. an authentication server (AS)
  3. a ticket-granting server (TGS)
• The KDC of course is critical and should be carefully guarded

Secrets Managed by the KDC

• A personal key used for encrypting/decrypting the database, and for enciphering / deciphering message contents it sends to itself!
• A master (semi-permanent) shared key for each user
• a master shared key for each server

Passwords and Tickets

1. Alice provides a password when she logs into her workstation
2. Alice’s workstation…
   – derives Alice’s master key from the password
   – asks the KDC for a temporary session key $K_A$
3. The KDC provides a ticket-granting ticket (TGT) for Alice to use; eliminates need for…
   – …repeated authentication
   – …further use of master key
Protocol Sketch (Common Case)

Alice
#1 Login
Password
Alice’s
Workstation
#2 Alice wants to authenticate
#3 Here’s Alice’s TGT
#4 Request
service from V
#5 Alice wants service from V
#6 Here is key + ticket to use
#7 Here is Alice’s ticket for
service + key to use
#8 Alice’s request for service is
granted, using key supplied

KDC
Server
V

Msg#1: Enter Password

#1 A→W: “Alice” | password
• Alice types in her user ID and password in unencrypted form into her workstation

Msg#2: Request for Authentication

#2 W→KDC: ID_A | TS_2 | ID_KDC
• Workstation sends a message to KDC with Alice’s ID (in unencrypted form)
• Many of these messages contain timestamps, for a) liveness, and b) anti-replay
• ID includes name and realm (see later)

Msg#3: Authentication Success

#3 KDC→W:
K_A,KDC(ID_A | TS_2 | Lifetime_TGT | K_A,KDC | ID_KDC | TGT)
• KDC sends Alice’s workstation a session key and a TGT
  – encrypted with the master key shared between Alice and the KDC
• K_A,KDC is derived from Alice’s password, used to decrypt session key K_A,KDC

Msg#3: … (cont’d)

K_KDC(ID_A | Addr_A | K_A,KDC | Lifetime_TGT | TS_TGT | ID_KDC)
• The TGT is what allows the KDC to be stateless
  – means simpler, more robust KDC design
  – allows replicated KCDCs (see later)
• The TGT contains
  – the session key to be used henceforth
  – the user ID (Alice)
  – the valid lifetime for the TGT

Msg#4: Alice Requests Service V

#4 A→W: ReqServ(V)
• Alice enters (to workstation) a request to access the service provided by V
Msg#5: Workstation Requests Service V

#5 W→KDC:

\[
\text{TGT | authenticator}_w | \text{TS}_o | \text{Lifetime}_o | \text{ID}_v
\]

- Workstation sends to the KDC…
  - the TGT previously granted (proves Alice’s identity)
  - the server she wishes to request service from
  - an authenticator for this message

Msg#5… (cont’d)

- The authenticator is an encrypted timestamp
  - why needed?
  - (reminder: timestamps requires user and KDC clocks to be loosely synchronized)

Msg#6: KDC Generates Ticket

#6 KDC→W:

\[
\mathcal{K}_{A-KDC}(\text{ID}_A | \text{TS}_o | \text{Lifetime}_o | \mathcal{K}_{A-V} | \text{ID}_v | \text{TKT}_v)
\]

- KDC decrypts the TGT and…
  - checks that lifetime has not expired
  - gets the shared key \(\mathcal{K}_{A-KDC}\)
- KDC sends back to workstation
  - identity of the server
  - a shared key \(\mathcal{K}_{A-V}\)
  - a ticket for Alice to present to V

Msg#6… (cont’d)

- The ticket contains
  - ID of the initiating user
  - shared key \(\mathcal{K}_{A-V}\)
  - lifetime of the ticket

Msg#7: Workstation Contacts Server

#7 W→V:

\[
\text{ID}_v | \text{TKT}_v | \text{authenticator}_v
\]

- Message contains
  - ticket (from the KDC)
  - authenticator
- If server V is replicated, ticket can be used with each server to receive service

Msg#7… (cont’d)

- Authenticator is valid for 5 minutes
  - loose synchronization required
  - replay attack possible for short period if server does not store previous authenticators
Done!

1. Alice has authenticated to KDC (which is trusted by server)
2. Server has authenticated to Alice
3. A session key has been negotiated, for encryption, message authentication, or both (but see previous discussions)

Key Updates

- Users will need to change their keys periodically, as do servers
- Implication: outstanding tickets (based on old keys) must be invalidated, and new ones issued
  - how find all those old tickets and recall them?
- Alternative: allow key versions
  - key version number to use is included in messages
  - KDCs and servers must allow overlap of old keys and new keys, allow time for use of old keys to age out

Adding Network Addresses to Tickets

- Add IP addresses (in addition to user IDs) to tickets
  - must match Source IP address in the packet containing the ticket, or message is rejected
  - just one more piece of information to make attacks harder (not foolproof, spoofing IP addresses is relatively easy)
- Problems
  - NATs will change IP addresses in packet headers but not in tickets
  - prevents delegating access rights (i.e., a ticket) to a user at another location
Specification of Messages

• See the text, or RFC, for full details

Kerberos v5 +
Interrealm Authentication

Some Differences with v4

1. v5 uses ASN.1 syntax to represent messages
   – a standardized syntax, not particularly easy to read
   – but, very flexible (optional fields, variable field lengths, extensible value sets, …)
2. v5 extends the set of encryption algorithms
3. v5 supports much longer ticket lifetimes
4. v5 allows “Pre-authentication” to thwart password attacks
5. v5 allows delegation of user access / rights

Delegation

• Giving someone else the right to access your services
  – how is that useful?
• Some not-so-good ways to implement
  – give someone else your password / key
  – give someone else your tickets (TKT’s)
• Kerberos v5 provides 3 better choices

Delegation… (cont’d)

• Choice #1: Alice asks the KDC to issue a TGT with Bob’s network address
  – she then passes this TGT and the corresponding session key to Bob
  – in effect, she tells the KDC she will be delegating this access right
• Choice #2: Alice asks the KDC to issue a TGT directly to Bob, with Bob’s address
  – even better, although now the KDC is required to contact Bob directly

Delegation… (cont’d)

• Choice #3: Alice gets a TGT, gives it to Bob
  – along with authorization data that will be passed to the application service, and must be interpreted by the application
**Transitive Delegation**

- Alice delegates to Bob who delegates to Carol who…
- TGTs (for arbitrary service) can be transitively delegated if marked as “forwardable”
- Tickets (providing access to a specific service) can be transitively delegated if marked as “proxiable”
- Servers are not obligated to honor such requests for transitive delegation

**Pre-Authentication**

- Reminder: Msg #3 is encrypted by the KDC with $K_{A,KDC}$
  - could be used by adversary to mount a password- or key-guessing attack
- Solution: before Msg #2, require Alice to send *pre-authentication data* to the KDC
  - i.e., a timestamp encrypted with the shared master key
  - this proves Alice knows the key

**Renewable Tickets**

- Tickets in v5 can be valid for a long time, but have to be *renewed* periodically, by contacting the KDC
- Each ticket contains
  - authorization time
  - start (valid) and end (expiration) times
  - renew-until (latest possible valid) time
- Newly-issued (renewed) tickets will have a new session key

**Cryptographic Algorithms in v5**

- **Message integrity only**
  - MD5 + encrypt result with DES using shared secret key
  - use DES residue
  - + others
- **Encryption + integrity**
  - basic = DES/CBC with a CRC
  - extended: 3DES + HMAC/SHA1
  - recently: AES/CBC + HMAC/SHA1
- **Note**: secret key only
“Sub-Session” Keys

- Alice may wish to use different keys for different conversations/connections with the same server – why?
- This is made possible by including in the authenticator of Msg #7 a subkey to use just for this connection

\[ x_{A} \cdot (x_{B} \cdot (x_{T}^{\text{msg7}})) \]
expanded to:

\[ x_{A} \cdot (x_{B} \cdot (x_{T}^{\text{msg7}} | \text{subkey})) \]

v5 Messages

- See text or RFC for lots of details, and specifications of message formats and contents…

Realms

- A realm is a group of resources sharing a single authority for authorization
  – frequently the same as a DNS domain, and referred to by the domain name (e.g., “ncsu.edu”)
- A realm consists of…
  1. KDC (TGS, AS, and database)
  2. users
  3. servers

Inter-Realm Authentication

- What if a user wants access to services located in a different realm?
- Simple solution: require Alice to be registered in each realm, has to undergo separate authentication in each
- More complex solution: the KDCs cooperate to perform inter-realm authentication
  – these KDCs must have previously-negotiated shared secret keys
  – receiving KDC can decide for itself whether to accept credentials issued by another KDC

Inter-Realm… (cont’d)

- A complex extension is the notion of inter-realm paths (> 2 KDCs cooperating)
- How find a path of cooperating KDCs to a target?
  – typical solution: hierarchy of KDCs (only one possible path)
- A ticket will contain the path of realms traversed by this ticket
  – the server receiving the ticket can decide if each of those realms is trustworthy, in order to accept or reject the ticket

Example
Summary

1. Kerberos is the most widely used authentication service
2. Modeled on the Needham-Schroeder protocol, but adds the TGT
3. v5 extends and fixes problems of v4; v4 no longer in active use
4. Inter-realm authentication scales to very large systems (e.g., the Internet)