What is Kerberos?

• Developed at M.I.T.
• A secret key based service for providing authentication in open networks
• Authentication mediated by a trusted 3rd party on the network:
  – Key Distribution Center (KDC)
Key Distribution Center

- Problem with Private Key Authentication
  - Need to establish key
  - for n people need $n^2$ keys
  - keys must be established via out-of-band communication
  - new entity requires $n$ new keys (draw fully connected graph)

- Solution: Key Distribution Center (KDC) (symmetric key)
  - trusted party used to assist in authentication
  - each party establishes a private key with the center

- have KDC trans-code a message with a session key
  - A sends to KDC $<A, K_A(B, K_s)>$
  - KDC sends to B $<K_b(A, K_s)>$
  - open to replay attack
    - T logs KDC to B message and all traffic using $K_s$
Mediated Authentication

• A trusted third party mediates the authentication process
• Called the Key Distribution Center (KDC)
• Each user and service shares a secret key with the KDC
• KDC generates a session key, and securely distributes it to communicating parties
• Communicating parties prove to each other that they know the session key

Mediated Authentication

• Nomenclature:
  – $K_a =$ Master key for “alice”, shared by alice and the KDC
  – $K_{ab} =$ Session key shared by “alice” and “bob”
  – $T_b =$ Ticket to use “bob”
  – $K\{data\} =$ “data” encrypted with key “K”
Mediated Authentication

Alice

I'm Alice, I want to speak to Bob

K_{ab}, Bob

KDC

generates key K_{ab}

K_{ab}, Alice

Bob

Now perform mutual authentication step to prove that both of you know K_{ab}
Mediated Authentication

Kerberos uses timestamps

- Timestamps as nonce’s are used in the mutual authentication phase of the protocol
- This reduces the number of total messages in the protocol
- But it means that Kerberos requires reasonably synchronized clocks amongst the users of the system
Kerberos (almost)

Kerberos (roughly)
Needham-Schroeder Protocol

Kerberos (detailed)

- Each user and service registers a secret key with the KDC
- Everyone trusts the KDC
  - “Put all your eggs in one basket, and then watch that basket very carefully” - Anonymous Mark Twain
- The user’s key is derived from a password, by applying a hash function
- The service key is a large random number, and stored on the server
Kerberos without TGS

- A simplified description of Kerberos without the concept of a TGS (Ticket Granting Service)
Combining 2 previous diags
Review: Kerberos Credentials

• **Ticket**
  – Allows user to use a service (actually authenticate to it)
  – Used to securely pass the identity of the user to which the ticket is issued between the KDC and the application server
  – $K_b\{\text{alice}, K_{ab}, \text{lifetime}\}$

• **Authenticator**
  – Proves that the user presenting the ticket is the user to which the ticket was issued
  – Proof that user knows the session key
  – Prevents ticket theft from being useful
  – Prevents replay attacks (timestamp encrypted with the session key): $K_{ab}\{\text{timestamp}\}$, in combination with a replay cache on the server

Ticket Granting Service (TGS)

• **Stateless servers……**
Kerberos with TGS

- **Ticket Granting Service (TGS):**
  - A Kerberos authenticated service, that allows user to obtain tickets for other services
  - Co-located at the KDC

- **Ticket Granting Ticket (TGT):**
  - Ticket used to access the TGS and obtain service tickets

- **Limited-lifetime session key: TGS sessionkey**
  - Shared by user and the TGS

- **TGT and TGS session-key cached on Alice’s workstation**

TGS Benefits

- **Single Sign-on (SSO) capability**

- **Limits exposure of user’s password**
  - Alice’s workstation can forget the password immediately after using it in the early stages of the protocol
  - Less data encrypted with the user’s secret key travels over the network, limiting attacker’s access to data that could be used in an offline dictionary attack
Color Code Scheme

- **Stuff** unencrypted
- **Stuff** encrypted w/ Alice's key
- **Stuff** encrypted w/ Bob's key
- **Stuff** encrypted w/ TGS key
- **Stuff** encrypted w/ TGS session key
- **Stuff** encrypted w/ Alice/Bob session key

Alice/Workstation  Bob/Application Server

1. alice, TGS
2. TGT = alice, $S_a$, life

User

alice, password

Workstation  Application Server (bob)
Limiting Password Exposure (memory)

• Early version:
  – Alice enters name/password
  – workstation sends request to KDC in clear
  – KDC responds encrypted w/ $K_A$
  – workstation uses $H(password)$ to decrypt
  – password in memory during round trip, for no good reason

• Later versionAlice types in name (not password)
  – request goes to KDC, encrypted response comes back
  – workstation prompts A for password, uses $H(password)$ to decrypt
  – password tossed, after being in memory much less time.

• Even later version
  – ask for password early, again, to encrypt request to KDC
  – (next slide)

Pre-authentication

• Kerberos 5 added pre-authentication
  – Client is required to prove it’s identity to the Kerberos AS in the first step
  – By supplying an encrypted timestamp (encrypted with users secret key)
  – This prevents an active attacker being able to easily obtain data from the KDC encrypted with any user’s key
    • Then able to mount an offline dictionary attack
Kerberos & Two-factor auth

- In addition to a secret password, user is required to present a physical item:
  - A small electronic device: h/w authentication token
  - Generates non-reusable numeric responses
- Called 2-factor authentication, because it requires 2 things:
  - Something the user knows (password)
  - Something the user has (hardware token)
Cross Realm Authentication

Hierarchy/Chain of Realms