Recap

• Last time:
  – Authentication protocol with public keys
  – Digital Signatures

• Today:
  – Key distribution
  – Needham-Schroeder
  – Kerberos
Needham-Schroeder-Lowe Protocol

\[ K_B\{n_A, A\} \]

\[ K_A\{n_A, n_B, B\} \]

\[ K_B\{n_B\} \]
Key Establishment

• Symmetric keys.
  – Point-to-Point.
  – Needham-Schroeder.
  – Kerberos.
Point-to-Point

• Should also use timestamps & nonces.
• Session key should include a validity duration.

Session Key

$K_{AB}\{K_S,t,B\}$
Key Distribution Centers

Give me a key to talk with Bart

Here is the key

Tom gave us this session key
Distribution Center Setup

• A wishes to communicate with B.
• T (trusted 3rd party) provides session keys.
• T has a key $K_{AT}$ in common with A and a key $K_{BT}$ in common with B.
• A authenticates T using a nonce $n_A$ and obtains a session key from T.
• A authenticates to B and transports the session key securely.
Needham-Schroeder Key Distribution Protocol

1. A → T : A, B, n_A
2. T → A : K_{AT}\{K_S, n_A, B, K_{BT}\{K_S, A\} \}
   A decrypts with K_{AT} and checks n_A and B. Holds K_S for future correspondence with B.
3. A → B : K_{BT}\{K_S, A\}
   B decrypts with K_{BT}.
4. B → A : K_S\{n_B\}
   A decrypts with K_S.
5. A → B : K_S\{n_B - 1\}
   B checks n_B-1.
Attack Scenario 1

1. A $\rightarrow$ T : A, B, $n_A$

2. T $\rightarrow$ C (A) : $K_{AT}\{k, n_A, B, K_{BT}\{K_S, A}\}$

C is unable to decrypt the message to A; passing it along unchanged does no harm. Any change will be detected by A.
Attack Scenario 2

1. $A \rightarrow C \ (T)$ : $A, B, n_A$
2. $C \ (A) \rightarrow T$ : $A, C, n_A$
3. $T \rightarrow A$ : $K_{AT}\{K_S, n_A, C, K_{CT}\{K_S, A}\}$

Rejected by $A$ because the message contains $C$ rather than $B$. 
Attack Scenario 3

1. $A \rightarrow C (T) : A, B, n_A$
2. $C \rightarrow T : C, B, n_A$
3. $T \rightarrow C : K_{CT}\{K_S, n_A, B, K_{BT}\{K_S, C}\}$
4. $C (T) \rightarrow A : K_{CT}\{K_S, n_A, B, K_{BT}\{K_S, C}\}$

A is unable to decrypt the message.
Attack Scenario 4

1. C → T : C, B, n_A
2. T → C : K_{CT}{K_S, n_A, B, K_{BT}{K_S, C}}
3. C (A) → B : K_{BT}{K_S, C}

B will see that the purported origin (A) does not match the identity indicated by the distribution center.
Valid Attack

- The attacker records the messages on the network (in particular, the messages sent in step 3)
- Consider an attacker that manages to get an old session key $K_S$.
- That attacker can then masquerade as Alice:
  - Replay starting from step 3 of the protocol, but using the message corresponding to $K_S$.

- Could be prevented with time stamps.
Kerberos

• Reading: "Kerberos: An Authentication Service for Open Network Systems" (by Steiner, Neuman, Schiller 1988)
  – Available on course web pages (along with link to Kerberos FAQ)

• Key exchange protocol developed at MIT in the late 1980’s
• Central server provides “tickets”
• Tickets – (a form of capabilities):
  – Unforgeable
  – Nonreplayable
  – Authenticated
  – Represent authority
• Designed to work with NFS (network file system)
• Also saves on authenticating for each service
  – e.g. with rlogin or rsh.
Kerberos

User

Service Request

Service ticket

Ticket-granting server

Unique keys

Kerberos Server

File Server

Other Server

Authentication

TGT

Auth. key

Service request
Kerberos Login

- **U** = User’s machine
- **S** = Kerberos Server
  - Has a database of user passwords: userID → pwd
- **G** = Ticket granting server

- U → S : userID, G, n_U
- S → U : k pwd{u, K UG}, K SG{T(U,G)}
- S → G : K SG{K UG, userID}

- T(X,Y) = X, Y, addr(X), L, K XY

**IP address of X**  **Ticket lifetime**

**Kerberos ticket granting ticket**

**Session key**
Kerberos Service Request

- $U \rightarrow G : K_{UG}\{userID, t\}, K_{SG}\{T\}, req(F), n'_U$

- $G \rightarrow U : K_{UG}\{K_{UF}, n'_U\}, K_{F}\{T(U,F)\}$

- $U \rightarrow F : K_{UF}\{userID, t\}, K_{F}\{T(U,F)\}$
Kerberos Benefits

- Distributed access control
  - No passwords communicated over the network
- Cryptographic protection against spoofing
  - All accesses mediated by G (ticket granting server)
- Limited period of validity
  - Servers check timestamps against ticket validity
  - Limits window of vulnerability
- Timestamps prevent replay attacks
  - Servers check timestamps against their own clocks to ensure “fresh” requests
- Mutual authentication
  - User sends nonce challenges
Kerberos Drawbacks

- Requires available ticket granting server
  - Could become a bottleneck
  - Must be reliable
- All servers must trust G, G must trust servers
  - They share unique keys
- Kerberos requires synchronized clocks
  - Replay can occur during validity period
  - Not easy to synchronize clocks
- User’s machine could save & replay passwords
  - Password is a weak spot
- Kerberos does not scale well
  - Hard to replicate authentication server and ticket granting server
  - Duplicating keys is bad, extra keys = more management