Announcements

• Midterm II
  – Thursday, March 22
  – In class, format similar to Midterm I

• Project 3 is available on the web pages
  – Handout for SDES needed for the project
  – Due: April 3rd
  – (Don't worry, it's shorter than the previous projects.)
Plan for Today

• Complete Digital Signatures
• Talk about Key Distribution Protocols
  – Needham Schroeder
  – Kerberos
Digital Signatures with Public Keys

• Assumes the algorithm is *commutative*:
  – \( D(E(M, K), k) = E(D(M, k), K) \)

• Let \( K_A \) be Alice’s public key
• Let \( k_A \) be her private key
• To sign msg, Alice sends \( D(msg, k_A) \)
• Bart can verify the message with Alice’s public key

• Works! RSA: \( (m^e)^d = m^{ed} = (m^d)^e \)
Digital Signatures with Public Keys

- No trusted 3rd party.
- Simpler algorithm.
- More expensive.
- No confidentiality.
Variations on Public Key Signatures

- Timestamps again (to prevent replay)
  - Signed certificate valid for only some time.

- Add an extra layer of encryption to guarantee confidentiality
  - Alice sends $A, K_B\{k_A\{msg\}\}$ to Bart

- Combined with hashes:
  - Send $(A, \text{msg}, k_A\{\text{MD5(msg)}\})$
Multiple Use of Keys

- Risky to use keys for multiple purposes.
- Using an RSA key for both authentication and signatures may allow a chosen-text attack.
- B attacker/verifier, $n_B = H(M)$ for some message $M$.

B, pretending to be A
Key Establishment

• Establishing a "session key"
  – A shared key used for encrypting communications for a short duration -- a session
  – Need to authenticate first

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

- Key establishment using only symmetric keys requires use of pre-distribution keys to get things going.

- Then protocol can be based on:
  - Point to point distribution, or
  - Key Distribution Center (KDC).
Point-to-Point

- Should also use timestamps & nonces.
- Session key should include a validity duration.
- Could also use public key cryptography to
  - Authenticate
  - Exchange symmetric shared 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 3\textsuperscript{rd} 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 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 → T : A, B, n_A

2. T → 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 → C (T) : A, B, n_A
2. C (A) → T : A, C, n_A
3. T → 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. $C \rightarrow T : C, B, n_A$

2. $T \rightarrow C : K_{CT}\{K_S, n_A, B, K_{BT}\{K_S, C}\}$

3. $C (A) \rightarrow B : K_{BT}\{K_S, C\}$

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

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

User → User Authentication → Kerberos Server → S → F → Other Server → F → File Server → F → Ticket-granting server → G → G → User

- Service Request
- Service ticket
- Unique keys $K_{FG}$, etc.
Kerberos Login

- **U** = User's machine
- **S** = Kerberos Server
  - Has a database of user "passwords": user\text{ID} \rightarrow k_{pwd}
- **G** = Ticket granting server

- **U** \rightarrow **S** : user\text{ID}, G, n_U
- **S** \rightarrow **U** : k_{pwd}(n_U, K_{UG}), K_{SG}(T(U,G))
- **S** \rightarrow **G** : K_{SG}(K_{UG}, user\text{ID})

- \text{T}(X,Y) = X, Y, L, K_{XY}

Kerberos ticket granting ticket
Session key
Ticket lifetime
Kerberos Service Request

• Requesting a service from server F

• \( U \rightarrow G : K_{UG}\{\text{userID}, \text{timestamp}\}, K_{SG}\{T(U,G)\}, \text{req}(F), n'_U \)

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

• \( U \rightarrow F : K_{UF}\{\text{userID}, \text{timestamp}\}, K_{FG}\{T(U,F)\} \)
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