Authentication Handshakes

- Secure communication almost always includes an initial authentication handshake.
  - Authenticate each other
  - Establish session keys
  - This process is not trivial; flaws in this process undermine secure communication

Authentication with Shared Secret

- Alice: I'm Alice
- Bob: A challenge $R$
- $K_{Alice-Bob}$

- Weaknesses
  - Authentication is not mutual; Trudy can convince Alice that she is Bob
  - Trudy can hijack the conversation after the initial exchange
  - If the shared key is derived from a password, Trudy can mount an off-line password guessing attack
  - Trudy may compromise Bob's database and later impersonate Alice
Authentication with Shared Secret (Cont’d)

- A variation
  - Requires reversible cryptography
  - Other variations are possible
- Weaknesses
  - All the previous weaknesses remain
  - Trudy doesn’t have to see R to mount off-line password guessing if R has certain patterns (e.g., concatenated with a timestamp)
    - Trudy sends a message to Bob, pretending to be Alice

Authentication with Public Key

- Bob’s database is less risky
- Weaknesses
  - Authentication is not mutual; Trudy can convince Alice that she is Bob
  - Trudy can hijack the conversation after the initial exchange
  - Trudy can trick Alice into signing something
    - Use different private key for authentication

Authentication with Public Key (Cont’d)

A variation
Mutual Authentication

<table>
<thead>
<tr>
<th>Alice</th>
<th>Bob</th>
</tr>
</thead>
<tbody>
<tr>
<td>I'm Alice, $R_1$</td>
<td></td>
</tr>
<tr>
<td>$f(K_{Alice-Bob}, R_1)$</td>
<td>$R_1$</td>
</tr>
<tr>
<td>$f(K_{Alice-Bob}, R_2)$</td>
<td></td>
</tr>
</tbody>
</table>

Optimize

<table>
<thead>
<tr>
<th>Alice</th>
<th>Bob</th>
</tr>
</thead>
<tbody>
<tr>
<td>I'm Alice, $R_1$</td>
<td></td>
</tr>
<tr>
<td>$R_2, f(K_{Alice-Bob}, R_2)$</td>
<td></td>
</tr>
<tr>
<td>$f(K_{Alice-Bob}, R_1)$</td>
<td></td>
</tr>
</tbody>
</table>

Mutual Authentication (Cont’d)

• Reflection attack

<table>
<thead>
<tr>
<th>Trudy</th>
<th>Bob</th>
</tr>
</thead>
<tbody>
<tr>
<td>I'm Alice, $R_1$</td>
<td></td>
</tr>
<tr>
<td>$R_2, f(K_{Alice-Bob}, R_3)$</td>
<td>$f(K_{Alice-Bob}, R_1)$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Trudy</th>
<th>Bob</th>
</tr>
</thead>
<tbody>
<tr>
<td>I'm Alice, $R_1$</td>
<td></td>
</tr>
<tr>
<td>$R_2, f(K_{Alice-Bob}, R_3)$</td>
<td>$f(K_{Alice-Bob}, R_1)$</td>
</tr>
</tbody>
</table>

Reflection Attacks (Con’td)

• Lesson: Don’t have Alice and Bob do exactly the same thing
  – Different keys
    • Totally different keys
    • $K_{Alice-Bob} = K_{Bob-Alice} + 1$
  – Different Challenges
  – The initiator should be the first to prove its identity
    • Assumption: initiator is more likely to be the bad guy
Mutual Authentication (Cont’d)

• Password guessing

Alice

I’m Alice, R₂

R₁, f(K_{Alice-Bob}, R₂)

f(K_{Alice-Bob}, R₁)

Bob

Countermeasure

Mutual Authentication (Cont’d)

• Public keys
  – Authentication of public keys is a critical issue

Alice

I’m Alice, {R₁}_{Bob}

Bob

{R₂}_{Alice}

{R₁}_{Alice}

R₁

Mutual Authentication (Cont’d)

• Mutual authentication with timestamps
  – Require synchronized clocks
  – Alice and Bob have to encrypt different timestamps

Alice

I’m Alice, f(K_{Alice-Bob}, timestamp)

Bob

f(K_{Alice-Bob}, timestamp+1)
Integrity/Encryption for Data

- Communication after mutual authentication should be cryptographically protected as well
  - Require a session key established during mutual authentication

Establishment of Session Keys

- Secret key based authentication
  - Assume the following authentication happened.
  - Can we use $K_{Alice-Bob}(R)$ as the session key?
  - Can we use $K_{Alice-Bob}(R + 1)$ as the session key?
  - In general, modify $K_{Alice-Bob}$ and encrypt $R$. Use the result as the session key.

<table>
<thead>
<tr>
<th>Alice</th>
<th>Bob</th>
</tr>
</thead>
<tbody>
<tr>
<td>I’m Alice</td>
<td>$R$</td>
</tr>
<tr>
<td>$K_{Alice-Bob}(R)$</td>
<td></td>
</tr>
</tbody>
</table>

Establishment of Session Keys (Cont’d)

- Two-way public key based authentication
  - Alice chooses a random number $R$, encrypts it with Bob’s public key
    - Trudy may hijack the conversation
  - Alice encrypts and signs $R$
    - Trudy may save all the traffic, and decrypt all the encrypted traffic when she is able to compromise Bob
    - Less severe threat
Two-Way Public Key Based Authentication (Cont’d)

- A better approach
  - Alice chooses and encrypts $R_1$ with Bob’s public key
  - Bob chooses and encrypts $R_2$ with Alice’s public key
  - Session key is $R_1 \oplus R_2$
  - Trudy will have to compromise both Alice and Bob

- An even better approach
  - Alice and Bob establish the session key with Diffie-Hellman key exchange
  - Alice and Bob signs the quantity they send
  - Trudy can’t learn anything about the session key even if she compromises both Alice and Bob

Establishment of Session Keys (Cont’d)

- One-way public key based authentication
  - It’s only necessary to authenticate the server
    - Example: SSL
  - Encrypt $R$ with Bob’s public key
  - Diffie-Hellman key exchange
  - Bob signs the D-H public key

Mediated Authentication (With KDC)

KDC operation (in principle)

- Some concerns
  - Trudy may claim to be Alice and talk to KDC
  - Trudy cannot get anything useful
  - Messages encrypted by Alice may get to Bob before KDC’s message
  - It may be difficult for KDC to connect to Bob
Mediated Authentication (With KDC)

KDC operation (in practice)

Alice \rightarrow KDC \rightarrow Bob

- Alice wants Bob
- Generate $K_{AB}$
- $K_{AB}(K_{Alice}), K_{AB}(K_{Bob})$
- $K_{Alice}(K_{Bob})$

- Must be followed by a mutual authentication exchange
  - To confirm that Alice and Bob have the same key

Needham-Schroeder Protocol

- Classic protocol for authentication with KDC
  - Many others have been modeled after it (e.g., Kerberos)
- Nonce: A number that is used only once
  - Deal with replay attacks

Alice \rightarrow KDC \rightarrow Bob

- Generate $K_{AB}$
- $N_1$, Alice wants Bob
- ticket to Bob, $K_{AB}\{N_2\}$
- $K_{Alice}\{N_1, 'Bob', K_{AB}, ticket to Bob\}$
- where ticket to Bob = $K_{Alice}(K_{AB}, Alice)$
- $K_{Bob}\{N_3, N_1\}$
- $K_{AB}\{N_3, N_2\}$

Needham-Schroeder Protocol (Cont’d)

- A vulnerability
  - When Trudy gets a previous key used by Alice, Trudy may reuse a previous ticket issued to Bob for Alice
  - Essential reason
    - The ticket to Bob stays valid even if Alice changes her key
## Expanded Needham-Schroeder Protocol

- Alice wants to talk to Bob, $K_{AB}(N_2)$
- The additional two messages assure Bob that the initiator has talked to KDC since Bob generates $N_B$

### Messages:

1. $K_{AB}(N_2)$
2. $K_{AB}(N_3)$
3. $K_{AB}(N_3)$

### Process:

- Alice wants to talk to Bob.
- KDC generates $K_{AB}$.
- Bob extracts $N_B$.
- KDC sends $K_{AB}(N_2)$, ticket to Bob, $K_{AB}(N_3)$.
- KDC sends $K_{AB}(N_3)$, $K_{AB}(N_3)$, ticket to Bob, $K_{AB}(N_3)$, $K_{AB}$.

## Otway-Rees Protocol

- Only has five messages
- KDC checks if $N_C$ matches in both cipher-texts
  - Make sure that Bob is really Bob

### Messages:

1. $K_{AB}(N_2)$
2. $K_{AB}(N_3)$
3. $K_{AB}(N_3)$
4. $K_{AB}(N_3)$
5. $K_{AB}$