Security Handshake Protocols

The following is a series of security handshake protocols. They are presented and evaluated according to security and performance. The performance parameters are:

- Number of messages,
- Processing power required, and
- Compactness of messages.

Login Protocols

Shared Secret

- **Protocol 1:**

  {==================================}
  Alice                                                                Bob
  I'm Alice  ------------------------>  
  <----------------------------- a Challenge R
  f(K, R)  -------------------------------->  
  {==================================}

  \( f(K, R) \): \( K \) is a shared secret between Alice and Bob. 
  \( f \) can be either encryption of hash function.

  **Pitfalls:**

  - An eavesdropper could mount an off-line passwd-guessing attack knowing both \( R \) and \( f(K,R) \).

- **Protocol 2:**

  {==================================}
  Alice                                                                Bob
  I'm Alice  ------------------------>  
  <-----------------------------K(R)
  R  -------------------------------->  
  {==================================}

  \( K(R) \) is encryption and not a hash function.

  **Pitfalls:**
If $R$ is a **recognizable quantity** (e.g., a 32-bit random number padded with 32 zero bits to fill out an encryption block), then Trudy, **without eavesdropping**, mount a dictionary attack by sending "I'm Alice" and obtaining $K[R]$.

On the other hand, Alice can authenticate Bob by recognizing $R$ (the 32 zeros). To foil the **replaying** of $K[R]$ (Trudy impersonate Bob to Alice), use a **timestamp** instead of zeros to fill the encryption block.

**Protocol 3 & 4:**

```
{ ======================================================
| Alice                                               | Bob |
| Protocol 3:                                        |
| I'm Alice, $K\{timestamp\} \longrightarrow$         |
| Protocol 4:                                        |
| I'm Alice, timestamp, hash $\{K, timestamp\} \longrightarrow$ |
| ======================================================
```

Requires that both Alice and Bob have reasonably **synchronized clocks**.
Beside saving two messages, Bob does not need to keep any **volatile state** (e.g., $R$).

**Pitfalls:**

- If Alice using the same $K$ on **multiple servers**, Trudy can send the same message to another server as Alice!
- Even if Alice is using $K$ for only one server, if Trudy can **reset** Bob's clock back, he can also impersonate Alice to the server.

**Clock-setting could be a serious security vulnerability**

---

**One-way Public Key**

Note that in the above four protocols, Trudy can impersonate Alice if she can **read Bob's database**.
This can be avoided by using public key technology.
• **Protocol 5:**

    {================================
      Alice                                                         Bob
      I'm Alice                         >
      <                                R
      sign R:  [R]Alice              >
    }  

    **Pitfalls:**

    • Trudy can trick Alice into signing something she does not know!

• **Protocol 6:**

    {================================
      Alice                                                        Bob
      I'm Alice                         >
      X <                                   {R}Alice  
      sign X:  R = [X]Alice        >
      }  

    **Pitfalls:**

    • Trudy can trick Alice into decrypting a message sent to Alice by someone else that he likes to read.

**The solution for the above two problems is to ensure that R has a known type/structure.**
Mutual Authentication

Shared Secret

- **Protocol 7:**

```plaintext
{=========================================================
Alice                                                   Bob
I’m Alice                                                 >
<--------------------------------------------------------Rb
f(K, Rb)                                                 >
Ra                                                      >
<--------------------------------------------------------f(K, Ra)
=========================================================
```

- **Protocol 8:**

Reduce number of messages in Protocol by putting more than one item of information into each message:

```plaintext
{=========================================================
Alice                                                   Bob
I’m Alice, Ra                                           >
<--------------------------------------------------------Rb, f(K, Ra)
f(K, Rb)                                                 >
=========================================================
```
**Pitfall 1: Reflection Attack**

Trudy can impersonate Alice to Bob by opening a second connection to Bob (or to another server that share the same secret with Alice):

**Session 1:**

```
Trudy                                                     Bob
I'm Alice, Ra   -------------------------------------------->
<------------------------Rb, f(K, Ra)
suspend session 1.....
```

**Session 2:**

```
Trudy                                                     Bob
I'm Alice, Rb   -------------------------------------------->
<------------------------Rb', f(K, Rb)
abort session 2.....
continue session 1.....
f(K, Rb)   -------------------------------------------->
```

**Possible fix:**

*Add your name to the encrypted quantity:*

```
Alice                                                     Bob
I'm Alice, Ra   -------------------------------------------->
<------------------------Rb, f(K, Bob|Ra)
f(K, Alice|Rb)   -------------------------------------------->
```
Why Protocol 7 does not suffer from the reflection attack? It follows a good security principle:

*The initiator should be the first to prove its identity.*

**Pitfall 2: Password guessing**
Trudy mount an off-line password guessing attack:

```plaintext
{========================================
Trudy                                                                 Bob
I'm Alice, Ra ..................................>  
<....................................................................Rb, f(K, Ra) 
............
suspend session and use: Ra, and f(K,Ra) to guess K.
========================================}  
```

Protocol 7 does not suffer from such attack (though Trudy can impersonate Bob to mount such attack, but it is much more difficult to impersonate Bob than to impersonate Alice).

- **Protocol 9:**

Protocol 7 is very good, since it does not suffer from Reflection and Password attacks. We can improve it by reducing the number of messages to four instead of five as follows:

```plaintext
{========================================
Alice                                                        Bob
I'm Alice ..................................>  
<..............................................Rb  
 f(K, Rb), Ra ..................................>  
<....................................................................f(K, Ra)
========================================}  
```
- **Protocol 10:**

We can use **time stamps** to reduce the number of messages to two:

```
<table>
<thead>
<tr>
<th>Alice</th>
<th>Bob</th>
</tr>
</thead>
<tbody>
<tr>
<td>I'm Alice, ( f(K, \text{timestamp}) )</td>
<td>( f(K, \text{timestamp}++) )</td>
</tr>
</tbody>
</table>
```

**Two-way Public Key**

- **Protocol 11:**

```
<table>
<thead>
<tr>
<th>Alice</th>
<th>Bob</th>
</tr>
</thead>
<tbody>
<tr>
<td>I'm Alice, Ra</td>
<td>( R_b, \text{signed } R_a )</td>
</tr>
<tr>
<td>( \text{signed } R_b )</td>
<td></td>
</tr>
</tbody>
</table>
```

- **Protocol 12:**

```
<table>
<thead>
<tr>
<th>Alice</th>
<th>Bob</th>
</tr>
</thead>
<tbody>
<tr>
<td>I'm Alice, ( {Ra}_{Bob} )</td>
<td>X</td>
</tr>
<tr>
<td>X</td>
<td>( \text{Ra, } {Rb}_{Alice} )</td>
</tr>
<tr>
<td>( R_b )</td>
<td></td>
</tr>
</tbody>
</table>
```
Mediated Authentication

KDC operation (in Principle):

\[
\begin{align*}
\text{Alice} & \quad \text{KDC} \quad \text{Bob} \\
\text{Alice wants Bob} & \quad \rightarrow \\
\left\langle \text{Ka \{use Kab for Bob\}}, \quad \text{ Kb \{use Kab for Alice\} } \rightarrow \\
\text{ mutual authentication using Kab } \quad \cdots \cdots \\
\end{align*}
\]

KDC operation (in Practice):

\[
\begin{align*}
\text{Alice} & \quad \text{KDC} \quad \text{Bob} \\
\text{Alice wants Bob} & \quad \rightarrow \\
\left\langle \text{Ka \{use Kab for Bob\}, ticket to Bob = Kb \{use Kab for Alice\} } \rightarrow \\
\text{ mutual authentication using Kab } \quad \cdots \cdots \\
\end{align*}
\]
Classical Examples of KDC authentication

The Basic Needham-Shroeder Protocol

N1, Alice wants Bob -->
<----------------- Ka {N1,"Bob", Kab, ticket to Bob},
where ticket to Bob = Kb {Kab, "Alice"}

ticket to Bob, Kab{N2} --------------------------------->

<------------------------------- Kab{N2--, N3}

Kab{N3--} --------------------------------->

- N is a "nonce", a number that is used only once (e.g., a sequence number, random number, timestamp).
- N1: to prevent Trudy from impersonating KDC and replaying old replies to Alice.
- N2 and N3 are challenges for mutual authentication.
The **Kerberos Authentication Protocol**:

It is based on Needham-Shroered protocol, but is much simpler since it is based on timestamp and the ticket includes expiration date.

{=====================================}

Alice                             KDC                             Bob

\[ N1, \text{Alice wants Bob} \rightarrow \]
\[ \leftarrow \text{Ka}\{N1, "Bob", Kab, ticket to Bob}\],
\[ \text{where ticket to Bob} = \text{Kb}\{\text{Kab, "Alice", expiration time}\} \]

\[ \text{ticket to Bob, Kab}\{\text{timestamp}\} \rightarrow \]
\[ \leftarrow \text{Kab}\{\text{timestamp++}\} \]

{=====================================}