Security Handshake Protocols

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

\textit{security \& performance}

The \textit{performance} parameters are:

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

Login Protocols

\textbf{Shared Secret}

- \textbf{Protocol 1:}

\begin{center}
\texttt{----------------------------------}\n\texttt{Alice} \hspace{2cm} \texttt{Bob} \\
\texttt{I'm Alice} \hspace{2cm} \texttt{a Challenge} \hspace{2cm} \texttt{R} \\
\texttt{f(K, R)} \hspace{2cm} \texttt{----------------------------------} \\
\end{center}

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

\textbf{Pitfalls:}

- An eavesdropper could mount an \textit{off-line passwd-guessing attack} knowing both \textit{R} and \textit{f(K, R)}. 
Protocol 2:

\[
\begin{align*}
\text{Alice} & \quad \rightarrow \quad \text{Bob} \\
\text{I'm Alice} & \\
\text{<} & \quad \text{K}\{R\} \\
\text{R} & \quad \rightarrow \quad \text{<}
\end{align*}
\]

\[\text{K}\{R\} \text{ is an encryption and not a hash function.}\]

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, may mount a dictionary attack by sending "I'm Alice" and obtaining \text{K}\{R\}.

- On the other hand, Alice can authenticate Bob by:

  Recognizing the 32 zero bits of R

- To foil the replaying of K\{R\} (Trudy impersonate Bob to Alice):

  Use a timestamp instead of 32 zero bits to fill the encryption block.

Protocol 3 & 4:

\[
\begin{align*}
\text{Alice} & \quad \rightarrow \quad \text{Bob} \\
\text{I'm Alice} & \\
\text{<} & \quad \text{K}\{R\} \\
\text{R} & \quad \rightarrow \quad \text{<}
\end{align*}
\]
**Protocol 3:**

I'm Alice, \( K\{\text{timestamp} \} \) \longrightarrow \\
\[ \text{-----------------------------} \]

**Protocol 4:**

I'm Alice, \( \text{timestamp}, \text{hash} \{K, \text{timestamp} \} \) \longrightarrow \\
\[ \text{-----------------------------} \]

These two protocols require both Alice and Bob to 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 back Bob's clock, he can impersonate Alice.

*Clock-setting could be serious security vulnerability*

**One-way Public Key**

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:**

```
<table>
<thead>
<tr>
<th>Alice</th>
<th>Bob</th>
</tr>
</thead>
<tbody>
<tr>
<td>I'm Alice &gt;</td>
<td>R</td>
</tr>
<tr>
<td>sign R: [R]_{Alice} &gt;</td>
<td></td>
</tr>
</tbody>
</table>
```

**Pitfalls:**

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

• **Protocol 6:**

```
<table>
<thead>
<tr>
<th>Alice</th>
<th>Bob</th>
</tr>
</thead>
<tbody>
<tr>
<td>I'm Alice &gt;</td>
<td></td>
</tr>
<tr>
<td>X &lt; {R}_{Alice}</td>
<td></td>
</tr>
<tr>
<td>sign X: R = [X]_{Alice} &gt;</td>
<td></td>
</tr>
</tbody>
</table>
```

**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 pitfalls is to make sure that:

* R has a known type/structure.
**Mutual Authentication**

**Shared Secret**

- **Protocol 7:**

  \[
  \{\quad \text{Alice} \quad \rightarrow \quad \text{Bob} \quad \}
  \]

  I'm Alice  \[\rightarrow\]  
  \[f(K, Rb)\]  \[\rightarrow\] 

  Ra  \[\rightarrow\]  
  \[f(K, Ra)\]  \[\rightarrow\] 

- **Protocol 8:**

  Reduce number of messages in Protocol 7 by putting *more than one* item of information into each message:

  \[
  \{\quad \text{Alice} \quad \rightarrow \quad \text{Bob} \quad \}
  \]

  I'm Alice, Ra  \[\rightarrow\]  
  \[f(K, Ra), Rb\]  \[\rightarrow\] 

  \[f(K, Rb)\]  \[\rightarrow\]
• **Pitfall 1: Reflection Attack**

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

**Session 1:**

```
Trudy
```

```
Bob
```

```
I'm Alice, Ra    >
< f(K, Ra), Rb
```

*suspend session 1.....*

**Session 2:**

```
Trudy
```

```
Bob
```

```
I'm Alice, Rb    >
< f(K, Rb), Rb'
```

*abort session 2.....*
\textit{continue session 1.....}

\[ f(K, Rb) \]

\[ \text{Possible fix:} \]
\textit{Add your name to the encrypted quantity:}

\[ \text{=================================} \]

\textbf{Alice} \hspace{2cm} \textbf{Bob}

I'm Alice, \( Ra \) \hspace{2cm} \rightarrow

\[ f(K, Bob|Ra), Rb \]

\[ f(K, Alice|Rb) \hspace{2cm} \rightarrow \]

\[ \text{=================================} \]

Why Protocol 7 does not suffer from the reflection attack?
It follows a good security principle:

\textit{The initiator should be the first to prove its identity.}

\[ \text{Pitfall 2: Password guessing} \]

Trudy may mount an off-line password guessing attack:

\[ \text{======================================} \]

\textbf{Trudy} \hspace{2cm} \textbf{Bob}

I'm Alice, \( Ra \) \hspace{2cm} \rightarrow

\[ f(K, Ra), Rb \]

\[ \text{suspend session and use: } Ra, \text{ and } f(K,Ra) \text{ to guess } K. \]
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:

```
{===============================================}

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**:

```
{===============================================}

Alice                                                        Bob
I'm Alice, f(K, timestamp)                                   
<-------------------------------------------------------------f(K, timestamp++)
```

Two-way Public Key

**Protocol 11:**

Alice  Bob

I'm Alice, Ra -->

< --- [Ra]Bob , Rb

[Rb]Alice , Rb -->

**Protocol 12:**

Alice  Bob

I'm Alice, {Ra}Bob -->

< --- Ra, {Rb}Alice

Rb -->
KDC operation (in Principle):

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

Alice    KDC    Bob

Alice wants Bob ————>  
<---------------  $Ka \{use \ text{Kab \ for \ Bob}\}$

$Kb \{use \ text{Kab \ for \ Alice}\}$ ————>

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

KDC operation (in Practice):

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

Alice    KDC    Bob

Alice wants Bob ————>  
<---------------  $Ka \{use \ text{Kab \ for \ Bob}\}$,
\textit{ticket to Bob} = $Kb \{use \ text{Kab \ for \ Alice}\}$

"I'm Alice",  \textit{ticket to Bob} ————>

........  \textit{mutual authentication using Kab}  ........

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

Classical Examples of KDC authentication
The Basic Needham-Shroeder Protocol

\[
\begin{array}{ccc}
\text{Alice} & \text{KDC} & \text{Bob} \\
N1, \text{ Alice wants Bob } \rightarrow \\
<---------- Ka \{N1,"Bob", Kab, ticket to Bob}, \\
\text{ where ticket to Bob } = \text{ Kb } \{Kab, "Alice"\} \\
n\text{ticket to Bob, Kab}\{N2\} \rightarrow \\
<------------------------ Kab\{N2--, N3\} \\
Kab \{N3--\} \rightarrow \\
\end{array}
\]

- \( N \) 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-Shroeder protocol, but is much simpler since it is based on timestamp and the ticket includes expiration date.
$N_1$, Alice wants Bob $\rightarrow$

$\leftarrow \quad Ka\{N_1,"Bob", Kab, \text{ticket to Bob}\}$,
where ticket to Bob $= Kb\{Kab,"Alice", \text{expiration time}\}$

ticket to Bob, Kab{timestamp} $\rightarrow$

$\leftarrow \quad Kab\{timestamp++\}$