

Security Handshake Protocols

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

security & performance

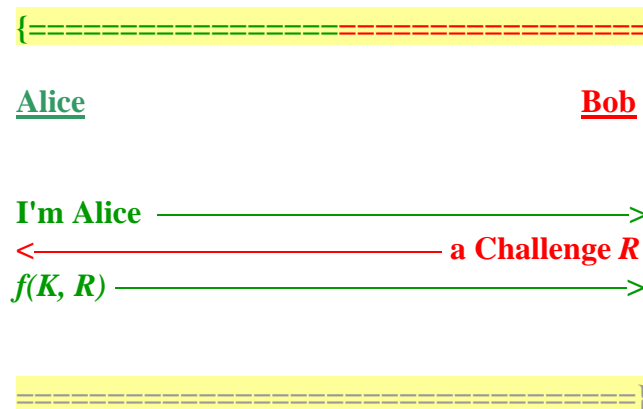
The *performance* parameters are:

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

Login Protocols

Shared Secret

- Protocol 1:

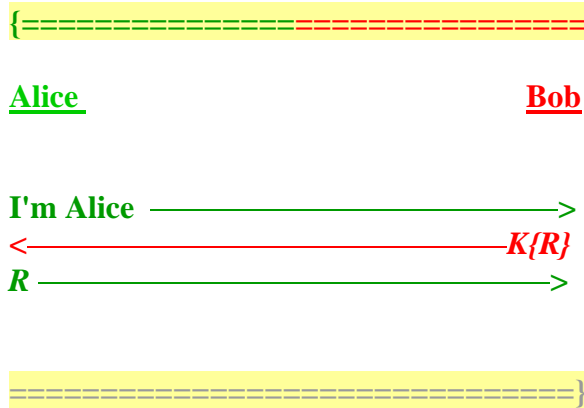


$f(K, R)$: K is a *shared secret* between Alice and Bob.
 f can be either *encryption* or *hash function*.

Pitfalls:

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

- **Protocol 2:**



$K\{R\}$ 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 $K\{R\}$.

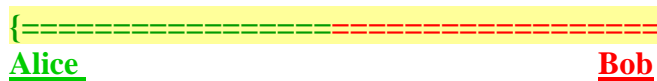
- On the other hand, *Alice can authenticate Bob* by:

Recognizing the *32 zero bits* of R

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

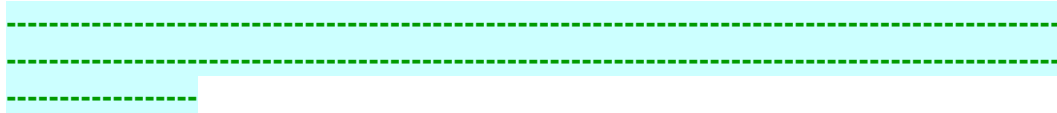
Use a *timestamp* instead of *32 zero bits* to fill the encryption block.

- **Protocol 3 & 4:**



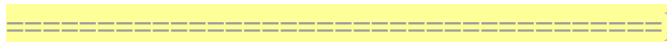
Protocol 3:

I'm Alice, $K\{timestamp\}$ \longrightarrow



Protocol 4:

I'm Alice, $timestamp, hash\{K, timestamp\}$ \longrightarrow



These two protocols requires 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

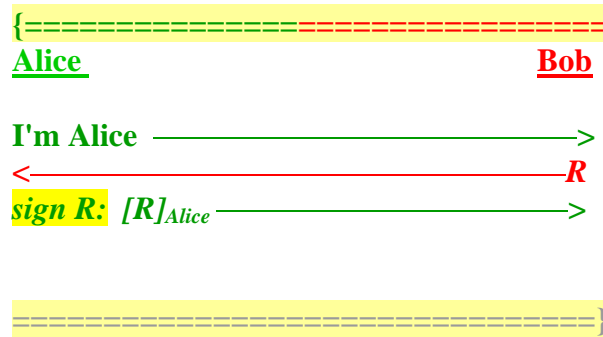
One-way Public Key

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.

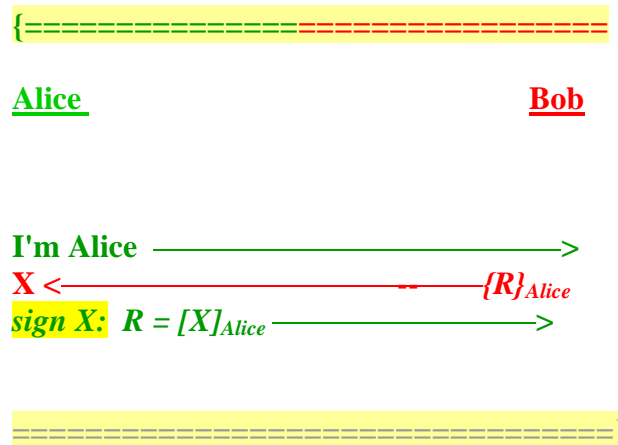
- **Protocol 5:**



Pitfalls:

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

- **Protocol 6:**



Pitfalls:

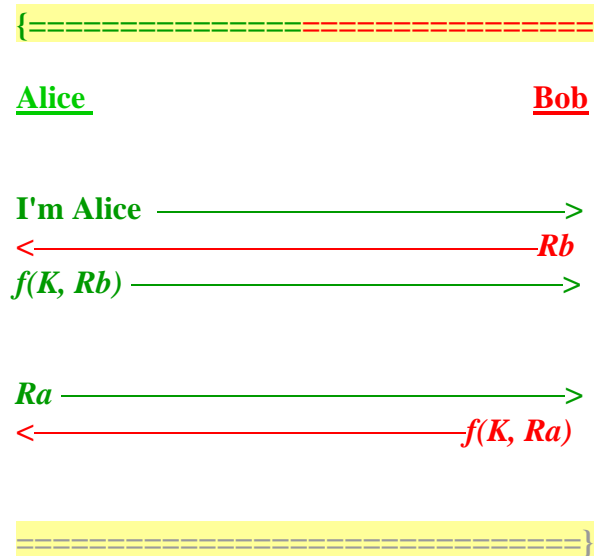
- Trudy can trick Alice into **decrypting** a message sent to Alice by someone else that he likes to read.

The solution for the above two pitfalls is to make sure that:
R has a **known type/structure**.

Mutual Authentication

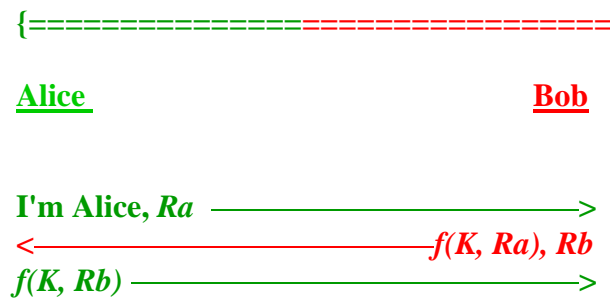
Shared Secret

- Protocol 7:



- Protocol 8:

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



continue session 1.....

$f(K, Rb)$ —————>

=====}

Possible fix:

Add your name to the encrypted quantity:

{=====}

Alice

Bob

I'm Alice, Ra —————>

<----- $f(K, \text{Bob}/Ra), Rb$

$f(K, \text{Alice}/Rb)$ —————>

=====}

**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 may mount an off-line password guessing attack:

{=====}

Trudy

Bob

I'm Alice, Ra —————>

<----- $f(K, Ra), Rb$

.....

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:

{=====}



=====}

- **Protocol 10:**

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

{=====}



=====}

Two-way Public Key

- Protocol 11:

{=====}

Alice

Bob

I'm Alice, R_a —————>
<----- $[R_a]_{Bob}, R_b$
 $[R_b]_{Alice}, R_b$ —————>

=====}

- Protocol 12:

{=====}

Alice

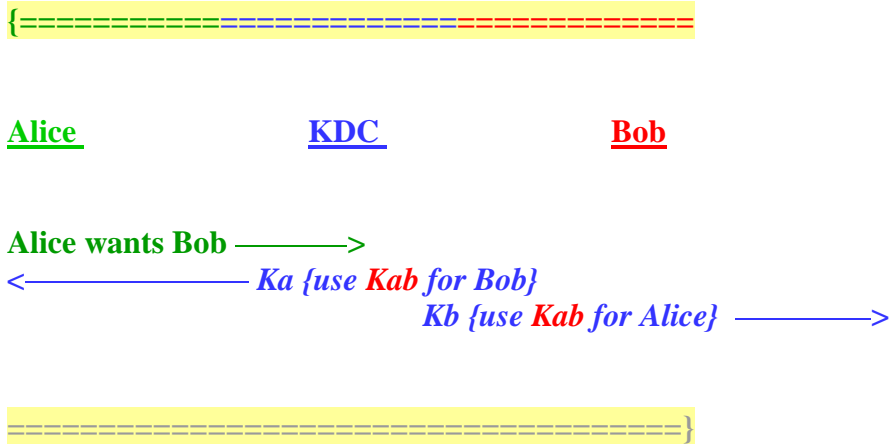
Bob

I'm Alice, $\{R_a\}_{Bob}$ —————>
<----- $R_a, \{R_b\}_{Alice}$
 R_b —————>

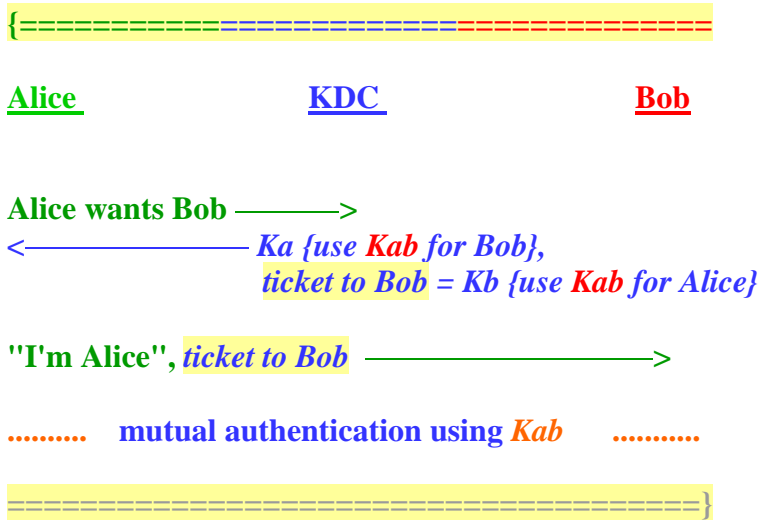
=====}

Mediated Authentication

KDC operation (in Principle):

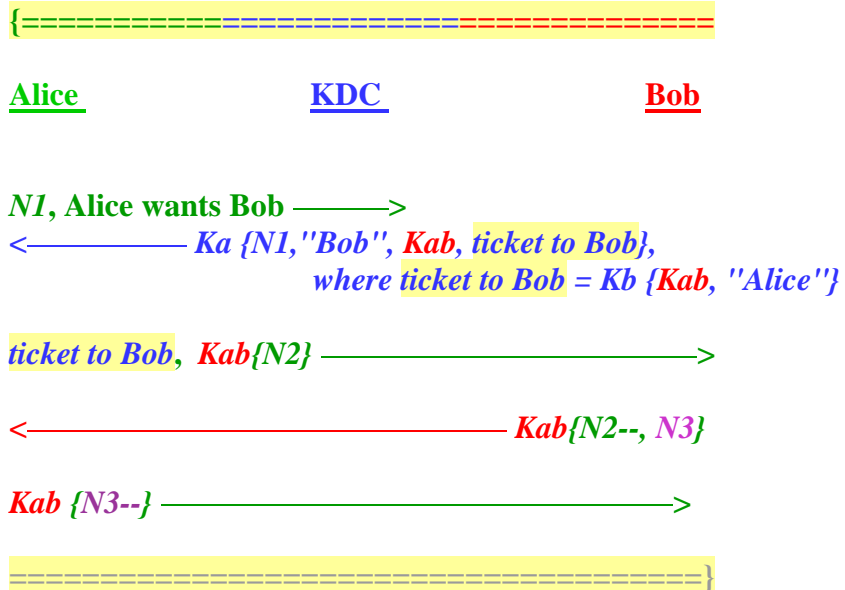


KDC operation (in Practice):



Classical Examples of KDC authentication

The Basic Needham-Shroeder Protocol



- 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.



NI, Alice wants Bob ———>

<———— *Ka{NI, "Bob", Kab, ticket to Bob},*
where *ticket to Bob = Kb {Kab, "Alice", expiration time}*

ticket to Bob, Kab{timestamp} ———>

<———— *Kab{timestamp++}*

=====}