Cryptography review: public key cryptography

- If $PK(Alice)$ is Alice's public key, then Bob can send Alice a secret message by encrypting it with Alice's public key: $\{m\}_{PK(Alice)}$; Alice can decrypt this message with her secret key $SK(Alice)$.
- Alice can send Bob an authenticated message by encrypting it with her secret key: $\{m\}_{SK(Alice)}$; Bob can decrypt this message with Alice's public key, and so verify that Alice sent it.

Example

Suppose agents a and b want to establish a cryptographic session key, with the help of a trusted server s.

We could arrange for s to generate the key and have it distributed to both agents as follows:

Msg 1. $a \rightarrow s : a, s, b$

Msg 2. $s \rightarrow a : s, a, b, k_{ab}$

Msg 3. $a \rightarrow b : a, b, k_{ab}$

What is wrong with this? Session key is unencrypted and therefore remains vulnerable to eavesdropping.

Second attempt

Let's suppose that a and b share long-term keys $shared(a, s)$ and $shared(b, s)$ with s. Then the key delivery messages could be encrypted with those keys:

Msg 1. $a \rightarrow s : a, s, b$

Msg 2. $s \rightarrow a : s, a, b, \{k_{ab}\}_{shared(a,s)}, \{k_{ab}\}_{shared(b,s)}$

Msg 3. $a \rightarrow b : a, b, \{k_{ab}\}_{shared(b,s)}$

This keeps the session key secret from eavesdroppers.

The fact that the key delivery message is encrypted with $shared(a, s)$ tells a that it was created by s.

Achieving authentication with shared keys

If:

- an agent a shares a key k with another agent s (and each knows that they share it),
- a receives a message encrypted with k, and
- a did not send the message himself,

then a can deduce that s created the message, and that s intended the message for a.