## Another attack

The protocol still suffers from a lack of explicitness.

The intruder can imitate the responder as follows:

Msg 1. $A \rightarrow I_S : A, B, \{K_a\}_{shared(A,S)}$

Msg 1'. $I_A \rightarrow S : A, I, \{K_a\}_{shared(A,S)}$

Msg 2. $S \rightarrow I : A$

Msg 3. $I \rightarrow S : \{K_{ab}\}_{shared(I,S)}$

Msg 4. $S \rightarrow A : \{K_{ab}\}_{K_a}$ .

## Another attack

The intruder can imitate the initiator as follows:

Msg 1. $I \rightarrow S : I, B, \{K_i\}_{shared(I,S)}$

Msg 2. $S \rightarrow I_B : I$

Msg 2'. $I_S \rightarrow B : A$

Msg 3. $B \rightarrow S : \{K_{ab}\}_{shared(B,S)}$

Msg 4. $S \rightarrow I : \{K_{ab}\}_{K_i}$ .



## Fixing the protocol

The obvious way to prevent these attacks is to include appropriate identities inside the encryptions:

Msg 1. $a \rightarrow s : a, \{b, k_a\}_{shared(a,s)}$

Msg 2. $s \rightarrow b : a$

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

Msg 4. $s \rightarrow a : \{k_{ab}\}_{k_a}$ .

Note the different forms of messages 1 and 3, in order to prevent one message being replayed in the place of the other.

## Freshness

The protocol still suffers from a lack of freshness: either a message 1 or a message 3 could be replayed, after $k_a$ or $k_{ab}$ has been compromised. This can be fixed using suitable nonce challenges:

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

Msg 2. $s \rightarrow a : n$

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

Msg 4. $s \rightarrow b : a, n'$

Msg 5. $b \rightarrow s : \{n', k_{ab}, a\}_{shared(b,s)}$

Msg 6. $s \rightarrow a : \{k_{ab}\}_{k_a}$ .