Problem 1. [15 points]

Does assertion *Inv* B_3 hold for program `Protocol`, where

B_3 : `(exists(A.S)` $\Rightarrow$ ψ`(A.S))`

Solution

It holds.

We have already shown that *Inv* ψ(`K`) holds (in the Note). So *Inv* ψ(`K+1`) also holds. [5 points]

Neither `A` nor `B` send out anything encrypted by `K+1`. [5 points]

So the only way the attacker can compute `enc(K+1,A.nA+A.nB)` is if `A.nA+A.nB` is some silly thing like `dec(K+1,K+1)`. But this is not the case because `A.nA` is randomly computed and the attacker cannot influence it. [5 points]

Problem 2. [15 points].

Does assertion *Inv* B_4 hold for `Protocol`, where

B_4 : `forall(i in hst.keys:  [B,S] = hst[i]` $\Rightarrow$ `([A,S] in hst[0..i-1]))`

Solution

Let's try to prove it.

Suppose `[B,enc(K+1,xA+xB)]` enters `hst` at time t_0, where `B.nB` equals `xB` and `B.nA` equals `xA`. So at t_0, `B` receives `[A,B,2,enc(K,xB),.]`.

Suppose `B.nB` was set to `xB` at time t_1 ($< t_0$). At t_1, `B` receives `[A,B,1,xA]` and responds with `[B,A,1,xB,enc(K,xA)]`. During (t_1, t_0), `B` is idle (otherwise its `nB` would not be `xB` at t_0).

At some time t_2 where $t_1 < t_2 < t_0$ holds, `A` sends a message with `enc(K,xB)` (the attacker couldn't have sent it because it does not have `K`, as proved earlier). So at t_2, `A` sends a `[A,B,2,enc(K,xB),.]` message (because that is the only kind of message that `A` sends with a `enc(K,xB)` field).

`A` sends this message at t_2 only if it receives a `[B,A,1,xB,enc(K,yA)]` message, where `yA` equals `A.nA`. *Because the attacker cannot send message* `[B,A,...]`, messae `[B,A,1,xB,enc(K,yA)]` was sent by `B`. Because field 3 equals `xB`, `B` sent this message when it set `B.nB` to `xB`, i.e., at time t_2. Hence `yA` equals `B.nB` at that time, which equals `xA`. So `[A,,enc(K+1,xA+xB)]` is added to `hst` at t_2.

So *Inv* B_4 holds.

Different problem

If the attacker could also send and receive messages as `B`, or read/write the channel, then *Inv* B_4 does not hold, as shown below.

Solution attempt 1

Let's try to prove it.

Suppose `[B,enc(K+1,xA+xB)]` enters `hst` at time t_0, where `B.nB` equals `xB` and `B.nA` equals `xA`. So at t_0, `B` receives `[A,B,2,enc(K,xB),.]`.

Suppose `B.nB` was set to `xB` at time t_1 ($< t_0$). At t_1, `B` receives `[A,B,1,xA]` and responds with `[B,A,1,xB,enc(K,xA)]`. During (t_1, t_0), `B` is idle (otherwise its `nB` would not be `xB` at t_0).

At some time t_2 where $t_1 < t_2 < t_0$ holds, `A` sends `[A,B,2,enc(K,xB),.]`. (The attacker couldn't have sent it because it does not have `K` (proved earlier)). So at t_2, `A` receives `[B,A,1,xB, enc(K,A.nA)]` and adds `[A,enc(K+1, A.nA+xB)]` to `hst`.

So if `A.nA` equals `xA` (which is what `B.nA` equals), then *Inv* B_4 would hold. Could the attacker arrange it so that `A.nA` is not `xA`? Think about it.

Solution

Let's try to disprove it. Below, "msg I" means the message sent in step I.

1. Initial step.

   After this: `A.nA = yA; [A,B,1,yA]` in `chan`.
2. `B` receives msg 1 (i.e., msg sent in step 1).

   After this: `B` is at `2`; `B.nA = yA; B.nB = yB; [B,A,1,yB,enc(K,yA)]` in `chan`.
3. Attacker receives msg 2 and sends `[A,B,2,zA,.]` where `zA` is not `yA`.
4. `B` receives msg 3 and goes back to 1 without updating `hst` (because `zA` does not equal `enc(K,yB)`).
5. Attacker sends `[A,B,1,zA,.]`.
6. `B` receives msg 5.

   After this: `B` is at `2`; `B.nA = zA; B.nB = zB; [B,A,1,zB,enc(K,zA)]` in `chan`.
7. Attacker receives msg 6, changes the last field to `enc(K,yA)` (which it had read in step 3), and sends `[B,A,1,zB,enc(K,yA)]`.
8. `A` receives msg 7 and updates `hst` (because it gets the response it expects).

   After this: `hst = [[A,enc(K+1, yA+zB)]]; [A,B,2, enc(K,zB),.]` in `chan`.
9. `B` receives msg 8 and adds `[B, enc(K+1, zA+zB)]` to `hst`.

   After this: `hst = [[A,enc(K+1, yA+zB)];  [B, enc(K+1, zA+zB)]]`.
   A_4 does not hold.

So *Inv* A_4 does not hold.

Can you come up with a simpler counter-example evolution?