Problem 1. [50 points]

Part a. [10 points]

Does Inv $A_1$ hold, where

\[ A_1 : \psi(K) \]

// attacker does not learn K

Solution

Yes.

Initially $K$ is not in $\alpha$.

The only expressions involving $K$ that the attacker can read are [B,A,1,B.nB,enc(K,B.nA)] messages (sent by B in function serveClient). Here B.nA is obtained from an [A,B,....] message in the channel, and so it can be a value generated by the attacker. But the attacker cannot set nA to be a simple function of $K$ or to $\text{dec}(K,K)$. So $\text{enc}(K,B.nA)$ does not expose $K$.

Part b. [10 points]

Does Inv $A_2$ hold, where

\[ A_2 : ([A,p] \text{ in } \text{hst}) \Rightarrow \psi(p) \]

// attacker does not learn any session key of A

Solution

Yes.

Let [A,p] be an entry in hst. Then p equals $\text{enc}(-K,xA+yB)$ where [xA,yB] equals [A.nA,A.nB] when the entry was added. Neither A nor B send out any encryptions using $-K$. The attacker may know xA and xB but it does not know K. Hence it does not know p.

Part c. [10 points]

Does Inv $A_3$ hold, where

\[ A_3 : ((i,j \text{ in } \text{hst.keys}) \text{ and } i \neq j \text{ and } \text{hst[i][0]} = \text{hst[j][0]} = A) \Rightarrow p \neq q \]

// A uses a session key only once

Solution

Yes.

Let $i$ and $j$ satisfy the lhs (left hand side) of $A_3$.

Then $\text{hst[i][1]}$ equals $\text{enc}(-K,xA+yB)$, where [xA,yB] equals [A.nA,A.nB] when the entry was added.

And $\text{hst[j][1]}$ equals $\text{enc}(-K,yA+yB)$, where [yA,yB] equals [A.nA,A.nB] when the entry was added.

Because $i$ differs from $j$ and because $A.1$ assigns a new random value to A.nA at each execution, $xA+yB$ differs from $yA+yB$ unless the attacker can choose $xB$ or $yB$ so that $xA+yB$ equals $yA+yB$. But A gets $xB$ and $yB$ from [B,A,...] messages, which the attacker cannot generate or modify. So $xB$ and $yB$ are different random values generated by B. So $xA+yB$ differs from $yA+yB$. 
Part d. [10 points]

Does $Inv_{A_4}$ hold, where

\[ A_4 : (i > 0 \text{ and } \text{hst}[i] = [B,p]) \Rightarrow \text{hst}[i-1] = [A,p] \]

// attacker cannot connect to the server as A

**Solution**

No.

The reflection attack works here. Here is an evolution ending in a state where $A_4$ does not hold. (Below, msg $j$ means message sent in step $j$.)

1. Initial: $[A,B,1,x_A,0]$ in channel, where $x_A$ equals $A.n_A$.
2. B.1 receives msg 1, starts thread $B.t[x_A]$, which sends response message.
3. Attacker receives msg 2. Attacker sends $[A,B,1,y_A,0]$ for some $y_A$ (e.g., $y_A=7$).
4. B.1 receives msg 3, starts thread $B.t[y_A]$, which sends response message $[B,A,1,y_B,\text{enc}(K,y_A)]$.
6. B.1 receives msg 5, starts thread $B.t[y_B]$, which sends response message $[B,A,1,\ldots,\text{enc}(K,y_B)]$.
8. Thread $B.t[y_A]$ at B.2 receives msg 7, adds $[B,\text{enc}(K,y_A+y_B)]$ to hst.
   At this point, this is the only entry in hst, so $A_4$ does not hold.

Part e. [10 points]

Can the attacker learn $K$ by dictionary attack, assuming that $K$ is a weak key.

**Solution**

Yes.

Consider steps 1–4 in the evolution of part d.

From step 3, the attacker has $y_A$ (it generates it).

From step 4, the attacker gets $\text{enc}(K,y_A)$ (from message $[B,A,1,y_B,\text{enc}(K,y_A)]$).

So the attacker can do the following dictionary attack:

```plaintext
for (cPw in Dictionary) {
    // cPw: candidate password
    generate cK from cPw; // cK: candidate key
    if (enc(cK,y_A) = enc(K,y_A))
        [cPw,cK] is user's [password, key]
}
```
Problem 2. [50 points]

Part a. [10 points]

Does \( \text{Inv } A_1 \) hold, where

\[
A_1 : \psi(K)
\]

// attacker does not learn \( K \)

Solution

Yes. The argument below is the same as in problem 1a, with \( K \) replaced by \( K+1 \).

Initially \( K \) is not in \( \alpha \).

The only expressions involving \( K \) that the attacker can read are \([B,A,1,B.nB,\text{enc}(K+1,B.nA)]\) messages (sent by \( B \) in function \( \text{serveClient} \)). Here \( B.nA \) is obtained from an \([A,B,...]\) message in the channel, and so it can be a value generated by the attacker. But the attacker cannot set \( nA \) to be a simple function of \( K+1 \) or to \( \text{dec}(K+1,K+1) \). So \( \text{enc}(K,B.nA) \) does not expose \( K+1 \), so it does not expose \( K \).

Part b. [10 points]

Does \( \text{Inv } A_2 \) hold, where

\[
A_2 : ([A,p] \text{ in } hst) \Rightarrow \psi(p)
\]

// attacker does not learn any session key of \( A \)

Solution

Yes. The argument below is the same as in problem 1b.

Let \([A,p]\) be an entry in \( hst \). Then \( p \) equals \( \text{enc}(-K,xA+xB) \) where \([xA,xB]\) equals \([A.nA,A.nB]\) when the entry was added. Neither \( A \) nor \( B \) send out any encryptions using \(-K\). The attacker may know \( xA \) and \( xB \) but it does not know \( K \). Hence it does not know \( p \).

Part c. [10 points]

Does \( \text{Inv } A_3 \) hold, where

\[
A_3 : ((i,j \text{ in } hst.keys) \text{ and } i \neq j \text{ and } hst[i][0] = hst[j][0] = A) \Rightarrow p \neq q
\]

// \( A \) uses a session key only once

Solution

Yes. The argument below is the same as in problem 1c.

Let \( i \) and \( j \) satisfy the lhs (left hand side) of \( A_3 \).

Then \( hst[i][1] \) equals \( \text{enc}(-K,xA+xB) \), where \([xA,xB]\) equals \([A.nA,A.nB]\) when the entry was added.

And \( hst[j][1] \) equals \( \text{enc}(-K,yA+yB) \), where \([yA,yB]\) equals \([A.nA,A.nB]\) when the entry was added.

Because \( i \) differs from \( j \) and because \( A.1 \) assigns a new random value to \( A.nA \) at each execution, \( xA+xB \) differs from \( yA+yB \) unless the attacker can choose \( xB \) or \( yB \) so that \( xA+xB \) equals \( yA+yB \). But \( A \) gets \( xB \) and \( yB \) from \([B,A,...]\) messages, which the attacker cannot generate or modify. So \( xB \) and \( yB \) are different random values generated by \( B \). So \( xA+xB \) differs from \( yA+yB \).
Part d. [10 points]

Does $Inv\ A_4$ hold, where

$$A_4: (i > 0 \text{ and } \text{hst}[i] = [B,p]) \Rightarrow \text{hst}[i-1] = [A,p]$$

// attacker cannot connect to the server as $A$

**Solution**

Yes. The reflection attack does not work here.

Let $[B,\text{enc}(−K, xA+xB)]$ be added to hst at time $t_0$, where $xA, xB$ equals $B.nA, B.nB$. We need to show that $[A,\text{enc}(−K, xA+xB)]$ is the last entry in hst just before $t_0$.

At $t_0$, thread $B.t[xA]$ is at 2 and receives $[A,B,2,xA,\text{enc}(K−1,xB)]$ (otherwise it would not have added the above entry to hst).

Let thread $B.t[xA]$ have set its $nB$ (i.e., $B.t[xA].nB$) to $xB$ at some time $t_1 (< t_0)$, upon receiving $[A,B,1,xA,0]$.

Because no thread in $B$ sends an encryption using $K−1$ and because the attacker does not have $K$, the $\text{enc}(K−1,xB)$ field in message $[A,B,2,xA,\text{enc}(K−1,xB)]$ was generated by $A$ at some time $t_2$ between $t_1$ and $t_0$. Because the attacker cannot alter or read this message, the entire message $[A,B,2,xA,\text{enc}(K−1,xB)]$ was generated by $A$ at time $t_2$.

So at $t_2$, $A$ receives $[B,A,1,xB,\text{enc}(K+1,yA)]$, where $yA$ equals $A.nA$, and added $[A,\text{enc}(−K, yA+xB)]$ to hst. This message was sent by $B$ (because the attacker cannot send a $[B,A,...]$ message). Because field 3 of this message is $xB$, this message was sent by thread $B.t[xA]$, i.e., it’s the message sent at time $t_1$. So $yA$ equals $xA$. So the entry that $A$ adds to hst at time $t_2$ is $[A,\text{enc}(−K, xA+xB)]$. Between $t_2$ and $t_0$, there is no change to hst. We are done.

Part e. [10 points]

Can the attacker learn $K$ by dictionary attack, assuming that $K$ is a weak key.

**Solution**

Yes. The argument below is the same as in problem 1e.

Consider the following evolution.

1. Initial: $[A,B,1,xA,0]$ in channel, where $xA$ equals $A.nA$.
2. $B.1$ receives msg 1, starts thread $B.t[xA]$, which sends response message.
3. Attacker receives msg 2. Attacker sends $[A,B,1,yA,0]$ for some $yA$ (e.g., $yA=7$).
4. $B.1$ receives msg 3, starts thread $B.t[yA]$, which sends response message $[B,A,1,yB,\text{enc}(K+1,yA)]$.

From step 3, the attacker has $yA$ (it generates it).

From step 4, the attacker gets $\text{enc}(K+1,yA)$.

So the attacker can do the following dictionary attack:

```plaintext
for (cPw in Dictionary) {
    // cPw: candidate password
    generate cK from cPw; // cK: candidate key
    if (enc(cK+1,yA) = enc(K+1,yA))
        [cPw,cK] is user’s [password, key]
}
```