1. [20 points]

Part a. [6 points]

Inv A_1 holds, where
$A_1 : \psi$(K)

Solution

True.

At the start, only A and B have K; it is not in α. The only expressions involving K seen by the attacker have the form enc(K,x), where x is received from the channel. So x can be a value generated by the attacker, but the attacker cannot set x to be a simple function of K or to dec(K,K). So enc(K,B.nA) does not expose K.

Part b. [8 points]

Inv A_2 holds, where
A_2 : ((i in hst.keys) and hst[i] = [B,p]) ⇒ ([A,p] in hst[0..i-1])

Solution

False.

Counter-example evolution.

1. Initial step
   After: [A,B,1,2] in chan and in α; hst = [].
2. Attacker changes msg 1 (i.e., message in step 1) to [A,B,1,3]. (Attacker knows that B's challenge will be 3.)
3. B.1: receive msg 2
   After: [B,A,1,3,enc(K,3)] in chan and in α; B at 2; B.nL = B.nR = 3; hst = [].
4. Attacker, using enc(K,3) field in msg 3, sets chan to [[A,B,2,enc(K,3)]].
5. B.2: receive msg 4
   After: hst = [[B,enc(−K,6)]]. A_1 not satisfied.

Part c. [6 points]

Attacker cannot learn K by dictionary attack, assuming that K is a weak key.

Solution

False

Consider the evolution consisting of the initial step and B.1 (and attacker only eavesdrops).
Then attacker gets:

- 2 //from msg [A,B,1,2] sent in initial step
- enc(K,2); refer to it as z. // from B's response msg [B,A,1,3,enc(K,2)]

Attacker can then do the following off-line dictionary attack:

```
for (cPw in Dictionary) {              // cPw: candidate password
   cK ← pwToKeyFunction(cPw);       // cK: candidate password key
   if (enc(cK,2) = z)
      cK equals K;
}
```

2. [10 points]

Can the attacker obtain data by dictionary attack, assuming K is a weak key?

Solution

Yes.

Here is an evolution ending with the attacker obtaining data.

1. Initial step
   After: let xA be A.nL's value; let tA = A.tL = g^{xA} mod p. [A,B,1,enc(K,tA)] in α;
2. Attacker:
   remove msg 1; generate random xB;
   let tB = g^{xA} mod p; send msg [B,A,1,tB]).
3. A.1: receive msg 2
   After: [A,B,2,enc(L,['HELLO',data])] in α, where L = tB^{xA} mod p = $g^{xA \cdot xB}$ mod p.

Attacker now has

- enc(K,tA); let this be z1. // step 2
- xB; tB = g^{xB} mod p // step 2
- enc(L,['HELLO',data]); let this be z2. // step 3

Attacker can then do the following off-line dictionary attack:

```
for (cPw in Dictionary) {          // cPw: candidate password
   cK ← pwToKeyFunction(cPw);      // cK: candidate password key
   cTA ← dec(cK,z1);               // cTA: candidate tA
   cKeyDH ← cTA^xA mod p;
   cMsg ← dec(cKeyDH, z2);
   if (cMsg.size = 2 and cMsg[0] = 'HELLO')
      data ← cMsg[1];
      // and K = cK and pw = cPw
}
```

3. [20 points]

Part a. [10 points]

Inv A_1 holds, where
A_1 : ψ(A.Key) // attacker does not learn A's current master key

Solution

True.

At the start, only A and Z have A.key; attacker does not. getPwdA does not give the attacker A.key, because it sets A.key to a new value (unknown to attacker) after reading it. Thus the only expressions involving A.key seen by the attacker are sent by Z; they have the form enc(A.key, x), where x involves a random component (kAB) and a value received from the channel (msg[3]). The attacker can set msg[3], but it cannot set x to a simple function of K or to dec(A.key, A.key). So the attacker cannot compute A.key.

Part b. [10 points]

Inv A_2 holds, where
A_2 : ((j in hst.keys) and j > 0 and hst[j] = [B,p]) $\Rightarrow$ hst[j−1] = [A,p]

Solution

False.

Counter-example evolution.

- Attacker eavesdrops an A-B connection, say with session key jAB and ticket jkt.
  After this: hst = [[A,jAB], [B,jAB]].
- Attacker calls getPwdA to get A.key's value, i.e., kAZ, and set A.key to new value.
  Attacker uses kAZ to get key jAB from messages in previous step.
- Attacker connects to B using ticket jkt and session key jAB.
  After this: hst = [[A,jAB], [B,jAB], [B,jAB]]. A_2 does not hold.

In more detail:

1. Initial step
   [A,Z,B,..] in channel.
2. Z.1: receive msg 1
   [Z,A,enc(kAZ,[..,B,jAB,jkt])] in channel and α; jkt = enc(kBZ,[jAB,A]).
3. A.1: receive msg 2; send msg [A,B,1,jkt,enc(jAB,.)].
4. B.1: receive msg 3; send msg [B,A,,enc(jAB,.)].
5. A.2: receive msg 4; send msg [A,B,enc(jAB,.)]; update hst to [[A,jAB]]; send msg [A,Z,B,..].
6. B.2: receive msg 5; update hst to [[A,jAB], [B,jAB]].
7. Attacker: call getPwdA to get TkAZ; decrypt field 2 of msg 2 using TkAZ to get jAB; send msg 3.
8. B.1: receive msg 7; send msg [B,A,,enc(kAB,[..,yB])], where yB = B.nL.
9. Attacker: receive msg 8; send msg [A,B,enc(jAB,yB−1)].
6. B.2: receive msg 9; update hst to [[A,jAB], [B,jAB], [B,jAB]].
   At this point, A_2 does not hold.
