Master M2 Cybersecurity — January, 2023
Course : Cryptographic Engineering — Duration 50 minutes

Problem

Your goal is to complete a model of a simple protocol in Tamarin (its Alice-Bob notation is given below).

```
I -> R :   I, g^h1(ex, x)
R -> I :   g^h1(ey, y), nR
I -> R :   senc(nR, k)
where:   * x - long-term private key of I
         * ex - ephemeral key generated by I (once per session)
         * y - long-term private key of R
         * ey - ephemeral key generated by R (once per session)
         * k = h2( g^(x* h1(ey, y)), g^(y* h1(ex, x)), g^(h1(ex, x)*h1(ey, y)) , I, R )
```

In this protocol (inspired from the Naxos protocol), each party A has a long-term private key xA and a corresponding public key pkA = *'g'^xA*, where *'g'* is a generator of the Diffie-Hellman group. Because *'g'* can be public, we model it as a public constant. Two different hash functions *h1* and *h2* are used.

To start a session, the initiator I first creates a fresh nonce *exI*, also known as I's ephemeral (private) key. He then concatenates *exI* with I's long-term private key xI, hashes the result using the hash function *h1*, and sends *'g'^h1(exI ,xI)* to the responder together with his identity. The responder R stores the received value in a variable X, computes a similar value based on his own nonce exR and long-term private key xR, concatenates it with a fresh nonce nR and sends the result to the initiator, who stores the received message in the variables Y and nR. Finally, both parties compute a session key (kI and kR, respectively) whose computation includes their own long-term private keys, such that only the intended partner can compute the same key. The last step is an acknowledgement from the initiator to the responder.

The incomplete modelization you should consider is given in Figure 2. Exercises:

1. Can you formulate an **"executable"** lemma in Tamarin that says that both participants (an initiator and a responder) can complete a session and compute the shared session key (even if the adversary does not corrupt anyone)?

   Hint: You should use an"exists-trace" assertion that uses the predicates Finish_I(...), Finish_R(...) and LtkRev(...) .

2. Can you formulate a **"secrecy_claim"** lemma in Tamarin that models the secrecy property of the session key (with respect to a session where both participants are not corrupted) ?

   Hint: The assertion should use the predicates Accept(...) and LtkRev(...).

3. We recall the definition of (perfect) forward secrecy: compromise of long-term keys of a set of principals does not compromise the session keys established in previous protocol runs involving those principals. Can you formulate a **"PFS_secrecy_claim"** lemma that models the PFS secrecy property of the session key?

   Hint: You should complete the previous lemma by allowing the adversary to corrupt the involved participants after the completion of the target session .

4. Can you complete the rules of the protocol and formulate a **"PFS_secrecy_claim_R"** lemma that models the PFS secrecy property of the session key from the point of view of the responder?

   Hint: You should add an action fact Accept_R(...) on one of the rules corresponding to the side of the responder.

5. Complete the rules of the protocol such that the lemma **"injective_agree"** should correspond to the Injective agreement property from the perspective of the responder.

   Hint: You should add action facts I_am_convinced(...) and I_think(...) on the right rules.

```
theory Naxos
begin
builtins: diffie-hellman, symmetric-encryption
functions: h1/1, h2/1

/* Generate long-term keypair */
rule generate_ltk:
   let pkA = 'g'^~xA   in
   [Fr(~xA)]  --[ OnlyOnce($A) ]->
   [ !Ltk( $A, ~xA ), !Pk( $A, pkA ), Out( pkA ) ]

/* Initiator */
rule Init_1:
  let hexI = h1(<~exI, ~xI >)
  hkI = 'g'^hexI
in
   [Fr( ~exI ), !Ltk( $I, ~xI )] -->
   [ Init_1( ~exI, $I, $R, ~xI, hkI ), Out( <$I, hkI> ) ]

rule Init_2:
  let hexI = h1(< ~exI, ~xI >)
  kI  = h2(< Y^~xI, pkR^hexI, Y^hexI, $I, $R >)
  c = senc(nR, kI)
in
   [ Init_1( ~exI, $I, $R, ~xI , hkI), !Pk( $R, pkR ), In( <Y, nR>  ) ]
   --[ Accept( $I, $R, kI), Finish_I( $I, $R, kI) ]->
   [ !Sessk( ~exI, kI), Out(c) ]

/* Responder */
rule Resp_1:
  let hexR = h1(< ~exR, ~xR >) hkr = 'g'^hexR
  kR  = h2(< pkI^hexR, X^~xR, X^hexR, I, $R >)  in
   [   Fr(~nR), Fr( ~exR ), !Ltk($R, ~xR), !Pk(I, pkI), In( <I, X> ) ]
   --[  Accept($R, I, kR ) ]->
   [   Resp_1( ~exR, I, $R, ~xR, hkr, ~nR, kR, X), Out( <hkr, ~nR> ) ]

rule Resp_2:
  let   c = senc(nR, kR) in
   [Resp_1(~exR, I, $R, ~lkR, hkr, nR, kR, X), In(c)]
   --[Finish_R($R, I, kR )]->
   [!Sessk( ~exR, kR)]

/* Key Reveals */
rule Ltk_reveal:
   [ !Ltk($A, lkA) ] --[ LtkRev($A) ]-> [ Out(lkA) ]

restriction OnlyOnce:
"   All x #i #j. OnlyOnce(x) @i & OnlyOnce(x) @j ==> #i = #j "
```

Figure 1: First Protocol

```
lemma executable:
   ...
```

```
 /*  If there exists a  session whose key k is known to the Adversary,
    then the long term key of one of the two participants involved in that session was revealed */
lemma secrecy_claim:
     ...
```

```
 /* If there exists a  session whose key k is known to the Adversary, then the long term key of one of
 the two participants involved in that session was revealed before that session  */
lemma PFS_secrecy_claim:
   ...
```

```
/* If there exists a session where the receiver has accepted a key k and k is known to the Adversary,
then  the long term key of one of the two participants involved in that session
was revealed before that session */
lemma PFS_secrecy_claim_R:
 ...
```