PBK: Purpose Built Keys Framework

original version

Protocol Purpose

Sender invariance (authentication assuming first message is not tampered with)

Definition Reference

http://www.ietf.org/internet-drafts/draft-bradner-pbk-frame-06.txt

Model Authors

- Daniel Plasto for Siemens CT IC 3, 2004
- Sebastian Mödersheim, ETH Zürich

Alice&Bob style

```
A -> B: A, PK_A, hash(PK_A)
A -> B: {Msg}_inv(PK_A), hash(PK_A)
B -> A: Nonce
A -> B: {Nonce}_inv(PK_A)
```

Problems considered: 1

Attacks Found

The initiator shall sign a random challenge received from the responder. This can easily be exploited to make agents sign whatever the intruder wishes:

```
i      -> (a,3) : start
(a,3)  -> i     : {Msg(1)}inv(pk_a),f(pk_a)
i      -> (a,12): start
(a,12) -> i     : {Msg(2)}inv(pk_a),f(pk_a)
```

```
i      -> (a,3) : x71
(a,3)  -> i     : {x71}inv(pk_a)
i      -> (b,3) : {x71}inv(pk_a),f(pk_a)
(b,3)  -> i     : Nonce(4)
i      -> (a,12): Nonce(4)
(a,12) -> i     : {Nonce(4)}inv(pk_a)
i      -> (b,3) : {Nonce(4)}inv(pk_a)
```

Further Notes

The protocol is so far only roughly described in natural language, and this file represents a verbatim translation to HLPSL as an "early prototype" and the AVISPA tool can identify a potential source for attacks which protocol designers should be aware of when implementing a protocol (see paragraph "Attacks"). A fixed version (with tagging the challenge before signing it) is also provided in this library.

The assumption is that the intruder cannot modify (or intercept) the first message is modelled by a compression-technique. Also, the authentication must be specified in a slightly different way, as A does not say for whom it signs the message (and anybody can act as responder).

HLPSL Specification

```
role alice (A,B        : agent,
            SND,RCV    : channel(dy),
            Hash       : function,
            PK_A       : public_key)
played_by A
def=

  local
    State      : nat,
    Msg        : text,
    Nonce      : text

  init State := 0
```

```
  transition

 1. State  = 0 /\ RCV(start) =|>
    State':= 2 /\ Msg' := new()
               /\ SND({Msg'}_inv(PK_A).Hash(PK_A))
               /\ witness(A,A,msg,Msg')

 3. State  = 2 /\ RCV(Nonce') =|>
    State':= 4 /\ SND({Nonce'}_inv(PK_A))

end role
```

```
role bob (B,A        : agent,
          SND,RCV    : channel(dy),
          Hash       : function,
          PK_A       : public_key)
played_by B
def=

  local
    State      : nat,
    Nonce      : text,
    Msg        : text

  init State := 1

  transition

 1. State  = 1 /\ RCV({Msg'}_inv(PK_A).Hash(PK_A)) =|>
    State':= 5 /\ Nonce' := new()
               /\ SND(Nonce')

 3. State  = 5 /\ RCV({Nonce}_inv(PK_A)) =|>
    State':= 7 /\ request(A,A,msg,Msg)

end role
```

```
role session(A,B  : agent,
             Hash : function,
             PK_A : public_key)
def=

  local SNDA,RCVA,SNDB,RCVB  : channel (dy)

  composition

    alice(A,B,SNDA,RCVA,Hash,PK_A)
 /\ bob(B,A,SNDB,RCVB,Hash,PK_A)

end role
```

```
role environment()
def=

  const
    a,b            : agent,
    f              : function,
    msg            : protocol_id,
    pk_a,pk_b,pk_i : public_key

  intruder_knowledge = {a,b,f,pk_a,pk_b,pk_i,inv(pk_i)}

  composition

     session(a,b,f,pk_a)
  /\ session(b,a,f,pk_b)
  /\ session(i,b,f,pk_i)
  /\ session(a,i,f,pk_a)

end role
```

```
goal
```

```
  %Alice authenticates Alice on msg
  authentication_on msg

end goal
```

```
environment()
```

References