SPKM-LIPKEY

Known initiator

Protocol Purpose

Provide a secure channel between a client and server, authenticating the client with a password, and a server with a public key certificate.

Definition Reference

RFC 2847, http://www.faqs.org/rfcs/rfc2847.html

Model Authors

- Boichut Yohan, LIFC-INRIA Besancon, May 2004
- Sebastian Mödersheim, ETH Zürich, January 2005

Alice&Bob style

```
1. A -> S:  A.S.Na.exp(G,X).{A.S.Na.exp(G,X)}_inv(Ka)
2. S -> A:  A.S.Na.Nb.exp(G,Y).{A.S.Na.Nb.exp(G,Y)}_inv(Ks)
3. A -> S: {login.pwd}_K where K= exp(exp(G,Y),X) = exp(exp(G,X),Y)
```

Model Limitations

In reality, the messages 1 and 2 contain respectively the two following items lists.

- the initiator and target names,
- a fresh random number,
- a list of available confidentiality algorithms,
- a list of available integrity algorithms,
- a list of available key establishment algorithms,

- a context key (or half key) corresponding to the first key establishment algorithm given in the previous list,
- GSS context options/choices (such as unilateral or mutual authentication, use of sequencing and replay detection, and so on).

and

- the initiator and target names,
- the random number sent by the initiator,
- a fresh random number,
- the subset of offered confidentiality algorithms which are supported by the target,
- the subset of offered integrity algorithms which are supported by the target,
- an alternative key establishment algorithm (chosen from the offered list) if the first one offered is unsuitable,
- the half key corresponding to the initiator's key establishment algorithm (if necessary), or a context key (or half key) corresponding to the key establishment algorithm above,
- GSS context options/choices (such as unilateral or mutual authentication, use of sequencing and replay detection, and so on).

The sets of algorithms agreed are not used by LIPKEY, indeed LIPKEY only uses SPKM for key establishment. Thus they are not modelled. Furthermore, the key establishment modelled is à la Diffie-Hellman and GSS context options are not modelled.

Problems considered: 2

Attacks Found

None

Further Notes

Here, we consider the case of mutual authentication, but there exists a unilateral (one-way) authentication version.

HLPSL Specification

```
role initiator (
        A: agent,
        S: agent,
        G: nat,
        H: function,
        Ka: public_key,
        Ks: public_key,
        Login_A_S: message,
        Pwd_A_S:   message,

        SND, RCV: channel (dy))
played_by A
def=

  local
        State        : nat,
        Na,Nb        : text,
        Rnumber1     : text,
        X            : message,
        Keycompleted : message,
        W            : nat,
        K            : text.text

  const sec_i_Log, sec_i_Pwd : protocol_id

  init  State := 0

  transition
```

```
  1.  State  = 0 /\ RCV(start) =|>
      State':= 1 /\ Na' := new()
                 /\ Rnumber1' := new()
                 /\ SND(A.S.Na'.exp(G,Rnumber1').
                        {A.S.Na'.exp(G,Rnumber1')}_inv(Ka))

  2.  State  = 1 /\ RCV(A.S.Na.Nb'.X'.{A.S.Na.Nb'.X'}_inv(Ks)) =|>
      State':= 2 /\ Keycompleted' := exp(X',Rnumber1)
                 /\ SND({Login_A_S.Pwd_A_S}_Keycompleted' )
                 /\ secret(Login_A_S,sec_i_Log,{S})
                 /\ secret(Pwd_A_S,  sec_i_Pwd,{S})
                 /\ K' := Login_A_S.Pwd_A_S
                 /\ request(A,S,ktrgtint,Keycompleted')
                 /\ witness(A,S,k,Keycompleted')

end role
```

```
role target(
        A,S             : agent,
        G             : nat,
        H             : function,
        Ka,Ks            : public_key,
        Login, Pwd    : function,
        SND, RCV      : channel (dy))
played_by S def=

  local State        : nat,
        Na,Nb        : text,
        Rnumber2     : text,
        Y            : message,
        Keycompleted : message,
        W            : nat,
        K            : text.text

  const sec_t_Log, sec_t_Pwd : protocol_id

  init  State := 0
```

```
  transition

  1. State  = 0 /\ RCV(A.S.Na'.Y'.{A.S.Na'.Y'}_inv(Ka)) =|>
     State':= 1 /\ Nb' := new()
                /\ Rnumber2' := new()
                /\ SND(A.S.Na'.Nb'.exp(G,Rnumber2').
                      {A.S.Na'.Nb'.exp(G,Rnumber2')}_inv(Ks))
                /\ Keycompleted'=exp(Y',Rnumber2')
                /\ secret(Login(A,S),sec_t_Log,{A})
                /\ secret(Pwd(A,S),  sec_t_Pwd,{A})
                /\ witness(S,A,ktrgtint,Keycompleted')

  2. State  = 1 /\ RCV({Login(A,S).Pwd(A,S)}_Keycompleted) =|>
     State':= 2 /\ K'=Login(A,S).Pwd(A,S)
                /\ request(S,A,k,Keycompleted)

end role
```

```
role session(
          A,S : agent,
          Login, Pwd: function,
          Ka: public_key,
          Ks: public_key,
          H: function,
          G: nat)
def=

  local   SndI, RcvI,
          SndT, RcvT : channel (dy)
  composition

     initiator(A,S,G,H,Ka,Ks,Login(A,S),Pwd(A,S),SndI,RcvI)
  /\ target(   A,S,G,H,Ka,Ks,Login,Pwd,SndT,RcvT)

end role
```

```
role environment()
```

```
def=

  const a,s,i,b: agent,
        ka, ki, kb, ks: public_key,
        login, pwd : function,
        h: function,
        g: nat,
        k,ktrgtint: protocol_id

  intruder_knowledge = {ki,i, inv(ki),a,b,s,h,g,ks,login(i,s),pwd(i,s),ka
                       }

  composition
            session(a,s,login,pwd,ka,ks,h,g)
        /\  session(b,s,login,pwd,kb,ks,h,g)
        /\  session(i,s,login,pwd,ki,ks,h,g)

end role
```

```
goal

  %Target authenticates Initiator on k
  authentication_on k
  %Initiator authenticates Target on ktrgtint
  authentication_on ktrgtint

  %secrecy_of Login, Pwd
  secrecy_of sec_i_Log, sec_i_Pwd,
             sec_t_Log, sec_t_Pwd

end goal
```

```
environment()
```

References