Fixed version

Protocol Purpose

Establish an authenticated (Diffie-Hellman) shared-key between a mobile terminal (MT) and a visited gate-keeper (VGK), who do not know each other in advance, but who have a "mutual friend", an authentication facility (AuF) in the home domain of MT.

Definition Reference

http://www.itu.int/rec/recommendation.asp?type=folders&lang=e&parent=T-REC-H.530 (with "corrigendum")

Model Authors

Sebastian Mödersheim, ETH Zürich

Alice&Bob style

```
  Macros
  M1 = MT,VGK,NIL,CH1,exp(G,X)
  M2 = M1,F(ZZ,M1),VGK,exp(G,X) XOR exp(G,Y)
  M3 = VGK,MT,F(ZZ,VGK),F(ZZ,exp(G,X) XOR exp(G,Y)),
       exp(G,X) XOR exp(G,Y)  %%% this is the very term added
                              %%% to fix the protocol
  M4 = VGK,MT,CH1,CH2,exp(G,Y),F(ZZ,exp(G,X) XOR exp(G,Y)),F(ZZ,VGK)
  M5 = MT,VGK,CH2,CH3
  M6 = VGK,MT,CH3,CH4
--------------------------------------------------------------------
  1. MT  -> VGK : M1,F(ZZ,M1)
  2. VGK -> AuF : M2,F(ZZ_VA,M2)
  3. AuF -> VGK : M3,F(ZZ_VA,M3)
  4. VGK -> MT  : M4,F(exp(exp(G,X),Y),M4)
  5. MT  -> VGK : M5,F(exp(exp(G,X),Y),M5)
  6. VGK -> MT  : M6,F(exp(exp(G,X),Y),M6)
```

Problems considered: 3

Attacks Found

None

Further Notes

This is the fixed version.

HLPSL Specification

```
role mobileTerminal (
    MT,VGK,AuF : agent,
    SND,RCV    : channel(dy),
    F          : function,
    ZZ         : symmetric_key,
    NIL,G      : text)
played_by MT def=

  local
    State       : nat,
    X,CH1,CH3   : text,
    CH2,CH4     : text,
    GY,Key      : message

  const sec_m_Key : protocol_id

  init  State := 0

  transition

 1. State  = 0 /\ RCV(start) =|>
    State':= 1 /\ X' := new()
```

```
             /\ CH1' := new()
             /\ SND(MT.VGK.NIL.CH1'.exp(G,X').F(ZZ.MT.VGK.NIL.CH1'.exp(G,X')))

 2. State  = 1 /\ RCV(VGK.MT.CH1.CH2'.GY'.
                     F(ZZ.xor(exp(G,X),GY')).
                     F(ZZ.VGK).
                     F(exp(GY',X).VGK.MT.CH1.CH2'.GY'.
                       F(ZZ.xor(exp(G,X),GY')).
                       F(ZZ.VGK)))
            =|>
    State':= 2 /\ CH3' := new()
               /\ Key' := exp(GY',X)
               /\ SND(MT.VGK.CH2'.CH3'.F(Key'.MT.VGK.CH2'.CH3'))
               /\ witness(MT,VGK,key1,Key')

 3. State  = 2 /\ RCV(VGK.MT.CH3.CH4'.F(Key.VGK.MT.CH3.CH4')) =|>
    State':= 3 /\ request(MT,VGK,key,Key)
               /\ secret(Key,sec_m_Key,{VGK,AuF})

end role
```

```
role visitedGateKeeper (
    MT,VGK,AuF : agent,
    SND,RCV    : channel(dy),
    F          : function,
    ZZ_VA      : symmetric_key,
    NIL,G      : text)
played_by VGK def=

  local
    State          : nat,
    GX,Key         : message,
    FM1,FM2,FM3,M2 : message,
    Y,CH2,CH4      : text,
    CH1,CH3        : text

  const sec_v_Key : protocol_id

  init  State := 0
  transition

  1. State = 0  /\ RCV(MT.VGK.NIL.CH1'.GX'.FM1') =|>
     State':= 1 /\ Y' := new()
                /\ Key' := exp(GX',Y')
                /\ M2' := MT.VGK.NIL.CH1'.GX'.FM1'.VGK.xor(GX',exp(G,Y'))
                /\ SND(M2'.F(ZZ_VA.M2'))
                /\ witness(VGK,MT,key,Key')

  2. State = 1  /\ RCV(VGK.MT.FM2'.FM3'.
                        xor(GX,exp(G,Y)).
                        F(ZZ_VA.VGK.MT.FM2'.FM3'.xor(GX,exp(G,Y)))) =|>
     State':= 2 /\ CH2' := new()
                /\ SND(  VGK.MT.CH1.CH2'.exp(G,Y).FM3'.FM2'.
                         F(Key.VGK.MT.CH1.CH2'.exp(G,Y).FM3'.FM2'))

  3. State = 2  /\ RCV(MT.VGK.CH2.CH3'.F(Key.MT.VGK.CH2.CH3')) =|>
     State':= 3 /\ CH4' := new()
                /\ SND(VGK.MT.CH3'.CH4'.F(Key.VGK.MT.CH3'.CH4'))
                /\ request(VGK,MT,key1,Key)
                /\ secret(Key,sec_v_Key,{MT})

end role
```

```
role authenticationFacility(
    MT,VGK,AuF : agent,
    SND,RCV    : channel(dy),
    F          : function,
    ZZ,ZZ_VA   : symmetric_key,
    NIL,G      : text)
played_by AuF def=

  local
    State          : nat,
    GX,GY          : message,
    CH1            : text

  init
```

```
    State := 0

  transition

  1. State = 0 /\  RCV(       MT.VGK.NIL.CH1'.GX'.
                         F(ZZ.MT.VGK.NIL.CH1'.GX').
                              VGK.xor(GX',GY').
                      F(ZZ_VA.MT.VGK.NIL.CH1'.GX'.
                         F(ZZ.MT.VGK.NIL.CH1'.GX').
                              VGK.xor(GX',GY')))  =|>

     State':= 1 /\ SND(       VGK.MT.F(ZZ.VGK).F(ZZ.xor(GX',GY')).xor(GX',GY').
                      F(ZZ_VA.VGK.MT.F(ZZ.VGK).F(ZZ.xor(GX',GY')).xor(GX',GY')))

end role
```

```
role session(
    MT,VGK,AuF : agent,
    F          : function,
    ZZ,ZZ_VA   : symmetric_key,
    NIL,G      : text)
def=

  local SND,RCV : channel (dy)

  composition
    mobileTerminal(MT,VGK,AuF,SND,RCV,F,ZZ,NIL,G)
 /\ authenticationFacility(MT,VGK,AuF,SND,RCV,F,ZZ,ZZ_VA,NIL,G)
 /\ visitedGateKeeper(MT,VGK,AuF,SND,RCV,F,ZZ_VA,NIL,G)

end role
```

```
role environment()
def=

  const
    a,b,auf                    : agent,
    f                          : function,
    key, key1                  : protocol_id,
    zz_a_auf,zz_b_auf,zz_i_auf : symmetric_key,
    nil,g                      : text

  intruder_knowledge = {a,b,auf,f,zz_i_auf}

  composition
    session(a,b,auf,f,zz_a_auf,zz_b_auf,nil,g)
 /\ session(i,b,auf,f,zz_i_auf,zz_b_auf,nil,g)
 /\ session(a,i,auf,f,zz_a_auf,zz_i_auf,nil,g)

end role
```

```
goal

  %MobileTerminal    authenticates VisitedGateKeeper on key
  authentication_on key
  %VisitedGateKeeper authenticates MobileTerminal    on key1
  authentication_on key1

  %secrecy_of Key
  secrecy_of sec_m_Key,sec_v_Key

end goal
```

```
environment()
```

References