authentication based on MACs, extended

Protocol Purpose

IKE is designed to perform mutual authentication and key exchange prior to setting up an IPsec connection. IKEv2 exists in several variants, the defining difference being the authentication method used.

This variant, which we call IKEv2-MACx, is based on exchanging the MAC of a pre-shared secret that both nodes possess. Analogous to the IKEv2-DSx variant, it also contains a slight extension in order to provide key confirmation.

Definition Reference

[Kau03]

Model Authors

- Sebastian Mödersheim, ETH Zürich, December 2003
- Paul Hankes Drielsma, ETH Zürich, December 2003

Alice&Bob style

IKEv2-MACx proceeds in three so-called exchanges. In the first, called IKE_SA_INIT, the users exchange nonces and perform a Diffie-Hellman exchange, establishing an initial security association called the IKE_SA. The second exchange, IKE_SA_AUTH, then authenticates the previous messages, exchanges the user identities, and establishes the first so-called "child security association" or CHILD_SA which will be used to secure the subsequent IPsec tunnel. A (respectively B) generates a nonce Na and a Diffie-Hellman half key KEa (respectively KEb). In addition, SAa1 contains A's cryptosuite offers and SAb1 B's preference for the establishment of the IKE_SA. Similarly SAa2 and SAb2 for the establishment of the CHILD_SA. The two parties share a secret in advance, the so-called PSK or pre-shared key. The authenticator message is built by taking a hash of the PSK and the previously exchanged messages. We extend these standard two exchanges with a third which we call EXTENSION. It consists of two messages, each containing a nonce (MA and MB, respectively) and a distinguished constant (0 and 1, respectively) encrypted with the IKE_SA key K.

```
IKE_SA_INIT
```

```
1. A -> B: SAa1, KEa, Na
2. B -> A: SAb1, KEb, Nb
IKE_SA_AUTH
3. A -> B: {A, AUTHa, SAa2}K
   where K = H(Na.Nb.SAa1.g^KEa^KEb) and
     AUTHa = F(PSK.SAa1.KEa.Na.Nb)
4. B -> A: {B, AUTHb, SAb2}K
   where
     AUTHb = F(PSK.SAa1.KEr.Na.Nb)
EXTENSION
5. A -> B: {MA, 0}K
6. B -> A: {MB, 1}K
```

Note that because we abstract away from the negotiation of cryptographic algorithms, we have SAa1 = SAb1 and SAa2 = SAb2.

Model Limitations

Issues abstracted from:

- The parties, Alice and Bob, should negotiate mutually acceptable cryptographic algorithms. This we abstract by modelling that Alice sends only a single offer for a crypto-suite, and Bob must accept this offer.
- There are goals of IKEv2 which we do not yet consider. For instance, identity hiding.
- We do not model the exchange of traffic selectors, which are specific to the IP network model and would be meaningless in our abstract communication model.

Problems considered: 3

Attacks Found

None.

HLPSL Specification

```
role alice(A,B: agent,
           G: text,
           F: function,
           PSK: symmetric_key,
           SND_B, RCV_B: channel (dy))
played_by A
def=

  local Ni, SA1, SA2, DHX: text,
        Nr: text,
        KEr: message, %% more specifically: exp(text,text)
        SK: message,
        State: nat,
        MA: text,
        MB: text,
        AUTH_B: message

  const sec_a_SK : protocol_id

  init  State := 0

  transition

  %% The IKE_SA_INIT exchange:
  1. State = 0  /\ RCV_B(start) =|>
     State':= 2 /\ SA1' := new()
                /\ DHX' := new()
                /\ Ni' := new()
                /\ SND_B( SA1'.exp(G,DHX').Ni' )

  %% Alice receives message 2 of IKE_SA_INIT, checks that Bob has
  %% indeed sent the same nonce in SAr1, and then sends the first
  %% message of IKE_AUTH.
  %% As authentication Data, she signs her first message and Bob's nonce.
  2. State = 2  /\ RCV_B(SA1.KEr'.Nr') =|>
     State' = 4 /\ SA2' := new()
                /\ SK' := F(Ni.Nr'.SA1.exp(KEr',DHX))
                /\ SND_B( {A.F(PSK.SA1.exp(G,DHX).Ni.Nr').SA2'}_SK' )
```

```
  3. State = 4  /\ RCV_B({B.F(PSK.SA1.KEr.Ni.Nr).SA2}_SK) =|>
     State':= 6 /\ MA' := new()
                /\ SND_B({MA'.zero}_SK)
                /\ AUTH_B' := F(PSK.SA1.KEr.Ni.Nr)
                /\ witness(A,B,sk1,SK)

  4. State = 6  /\ RCV_B({MB'.one}_SK) =|>
     State':= 8 /\ secret(SK,sec_a_SK,{A,B})
                /\ request(A,B,sk2,SK)

end role
```

```
role bob(B,A:agent,
         G: text,
         F: function,
         PSK: symmetric_key,
         SND_A, RCV_A: channel (dy))
played_by B
def=

  local Ni, SA1, SA2: text,
        Nr, DHY: text,
        SK, KEi: message,
        State: nat,
        MA: text,
        MB: text,
        AUTH_A: message

  const sec_b_SK : protocol_id

  init  State := 1

  transition

  1. State = 1  /\ RCV_A( SA1'.KEi'.Ni' ) =|>
     State':= 3 /\ DHY' := new()
                /\ Nr' := new()
                /\ SND_A(SA1'.exp(G,DHY').Nr')
```

```
                /\ SK' := F(Ni'.Nr'.SA1'.exp(KEi',DHY'))

  2. State = 3  /\ RCV_A( {A.F(PSK.SA1.KEi.Ni.Nr).SA2'}_SK ) =|>
     State':= 5 /\ SND_A( {B.F(PSK.SA1.exp(G,DHY).Ni.Nr).SA2'}_SK )
                /\ AUTH_A' := F(PSK.SA1.KEi.Ni.Nr)
                /\ witness(B,A,sk2,SK)

  3. State = 5  /\ RCV_A({MA'.zero}_SK) =|>
     State':= 7 /\ MB' := new()
                /\ SND_A({MB'.one}_SK)
                /\ secret(SK,sec_b_SK,{A,B})
                /\ request(B,A,sk1,SK)

end role
```

```
role session(A, B: agent,
             PSK: symmetric_key,
             G: text, F: function)
def=

  local SA, RA, SB, RB: channel (dy)

  composition

     alice(A,B,G,F,PSK,SA,RA)
  /\ bob(B,A,G,F,PSK,SB,RB)

end role
```

```
role environment()
def=

  const sk1, sk2      : protocol_id,
        a, b          : agent,
        kab, kai, kbi : symmetric_key,
        g :text, f          : function,
        zero, one     : text
```

```
  intruder_knowledge = {g,f,a,b,i,kai,kbi,zero,one
                       }

  composition

        session(a,b,kab,g,f)
     /\ session(a,i,kai,g,f)
     /\ session(i,b,kbi,g,f)

end role
```

```
goal

  %secrecy_of SK
  secrecy_of sec_a_SK, sec_b_SK

  %Alice authenticates Bob on sk
  authentication_on sk1
  %Bob authenticates Alice on sk
  authentication_on sk2

end goal
```

```
environment()
```

References

[Kau03] Charlie Kaufman. Internet Key Exchange (IKEv2) Protocol, October 2003. Work in Progress.