public key initialisation

Protocol Purpose

Mutual Authentication with Public Key initialisation (in case the Authentication Server and Client don't share a key)

Definition Reference

- http://www.ietf.org/internet-drafts/draft-ietf-cat-kerberos-pk-init-22.txt

Model Authors

- Vishal Sankhla, University of Southern California, August 2004
- Daniel Plasto for Siemens CT IC 3, 2004

Alice&Bob style

```
C -> A: U,G,N1,{Kca,T0,N1,hash(U,G,N1)}inv(Kca)

  In PKINIT, the first message contains additional
  information in the pre-authentication field:
  The public key of U, a timestamp, the nonce repeated,
  and a checksum of the message body.  This is all signed
  with the private key of U.

A -> C: U,Tcg,{G,Kcg,T1start,T1expire,N1}Ktemp,{{Ktemp}Kca}inv(Pka)

where Tcg := {U,C,G,Kcg,T1start,T1expire}Kag

  A replies as usual, except the reply is encrypted
  with a random key, and this key is included in the
  pre-authentication field and encrypted with the U's
  public key and signed with the A's private key.

C -> G: S,N2,Tcg,Acg
G -> C: U,Tcs,{S,Kcs,T2start,T2expire,N2}Kcg
```

```
where Acg := {C,T1}Kcg  (T1 is a timestamp)
      Tcs := {U,C,S,Kcs,T2start,T2expire}Kgs

C -> S: Tcs,Acs
S -> C: {T2'}Kcs

where Acs := {C,T2'}Kcs  (T2 is a timestamp)
```

The AS, TGS and S cache the timestamps they have received in order to prevent replays as specified in RFC 1510.

We assume that the Key Distribution Centre (KDC) is the certifying authority here.

Problems considered: 7

Attacks Found

None

HLPSL Specification

```
role authenticationServer(
             A,C,G    : agent,
             Kca      : public_key,
             Kag      : symmetric_key,
             SND, RCV : channel(dy),
             L        : text set,
             Pka      : public_key,
             Hash     : function)
played_by A
def=

  local State    : nat,
        N1       : text,
        U        : text,
        T0       : text,
```

```
        Kcg      : symmetric_key,
        T1start  : text,
        T1expire : text,
        Ktemp    : symmetric_key

  const sec_a_Kcg : protocol_id

  init  State := 11

  transition
    1. State  = 11  /\ RCV(U'.G.N1'.
                           {Kca.T0'.N1'.Hash(U'.G.N1')}_inv(Kca))
                    /\ not(in(T0',L)) =|>
       State' := 12 /\ Kcg' := new()
                    /\ T1start' := new()
                    /\ T1expire' := new()
                    /\ Ktemp' := new()
                    /\ SND(U'.
                           {U'.C.G.Kcg'.T1start'.T1expire'}_Kag.
                           {G.Kcg'.T1start'.T1expire'.N1'}_Ktemp'.
                           {{Ktemp'}_Kca}_inv(Pka))
                    /\ L' := cons(T0',L)
                    /\ witness(A,C,n1,N1')
                    /\ wrequest(A,C,t0,T0')
                    /\ secret(Kcg',sec_a_Kcg,{A,C,G})

end role
```

```
role ticketGrantingServer (
             G,S,C,A      : agent,
             Kag,Kgs      : symmetric_key,
             SND,RCV      : channel(dy),
             L            : text set)
played_by G
def=

  local State   : nat,
        N2      : text,
        U       : text,
```

```
        Kcg     : symmetric_key,
        Kcs     : symmetric_key,
        T1start,T1expire : text,
        T2start,T2expire : text,
        T1      : text

  const sec_t_Kcg, sec_t_Kcs : protocol_id

  init  State := 21

  transition
    1. State  = 21  /\ RCV( S.N2'.
                           {U'.C.G.Kcg'.T1start'.T1expire'}_Kag.
                           {C.T1'}_Kcg')
                           /\ not(in(T1',L)) =|>
       State' := 22 /\ Kcs' := new()
                    /\ T2start' := new()
                    /\ T2expire' := new()
                    /\ SND(  U'.
                           {U'.C.S.Kcs'.T2start'.T2expire'}_Kgs.
                           {S.Kcs'.T2start'.T2expire'.N2'}_Kcg'
                         )
                    /\ L' := cons(T1',L)
                    /\ wrequest(G,C,t1,T1')
                    /\ witness(G,C,n2,N2')
                    /\ secret(Kcg',sec_t_Kcg,{A,C,G})
                    /\ secret(Kcs',sec_t_Kcs,{G,C,S})

end role
```

```
role server( S,C,G    : agent,
             Kgs   : symmetric_key,
             SND, RCV : channel(dy),
             L       : text set)
played_by S
def=

  local State   : nat,
        U       : text,
```

```
        Kcs     : symmetric_key,
        T2expire: text,
        T2start : text,
        T2      : text

  const sec_s_Kcs : protocol_id

  init  State := 31

  transition
    1. State  = 31  /\ RCV({U'.C.S.Kcs'.T2start'.T2expire'}_Kgs.{C.T2'}_Kcs')
                    /\ not(in(T2',L)) =|>
       State' := 32 /\ SND({T2'}_Kcs')
                    /\ L' := cons(T2',L)
                    /\ witness(S,C,t2a,T2')
                    /\ request(S,C,t2b,T2')
                    /\ secret(Kcs',sec_s_Kcs,{G,C,S})

end role
```

```
role client( C,G,S,A      : agent,
             SND,RCV      : channel(dy),
             Kca,Pka      : public_key,
             U            : text,
             Hash         : function)
played_by C
def=

  local State    : nat,
        Kcs      : symmetric_key,
        T1expire : text,
        T2expire : text,
        T1start  : text,
        T2start  : text,
        Kcg      : symmetric_key,
        Tcg,Tcs  : {text.agent.agent.symmetric_key.text.text}_symmetric_key,
        T0,T1,T2 : text,
        Ktemp    : symmetric_key,
        N1, N2   : text
```

```
    const sec_c_Kcs,sec_c_Kcg : protocol_id

    init  State := 1

    transition
      1. State  = 1  /\ RCV(start) =|>
         State' := 2 /\ T0' := new()
                     /\ N1' := new()
                     /\ SND(U.G.N1'.{Kca.T0'.N1'.Hash(U.G.N1')}_inv(Kca))
                     /\ witness(C,A,t0,T0')

      2. State  = 2  /\ RCV(U.Tcg'.
                            {G.Kcg'.T1start'.T1expire'.N1}_Ktemp'.
                            {{Ktemp'}_Kca}_inv(Pka)) =|>
         State' := 3 /\ T1' := new()
                     /\ N2' := new()
                     /\ SND(S.N2'.Tcg'.{C.T1'}_Kcg')
                     /\ witness(C,G,t1,T1')
                     /\ request(C,A,n1,N1)
                     /\ secret(Kcg',sec_c_Kcg,{A,C,G})

      3. State  = 3  /\ RCV(U.Tcs'.{S.Kcs'.T2start'.T2expire'.N2}_Kcg) =|>
         State' := 4 /\ T2' := new()
                     /\ SND(Tcs'.{C.T2'}_Kcs')
                     /\ witness(C,S,t2b,T2')
                     /\ request(C,G,n2,N2)
                     /\ secret(Kcs',sec_c_Kcs,{G,C,S})

      4. State  = 4  /\ RCV({T2}_Kcs) =|>
         State' := 5 /\ request(C,S,t2a,T2)

  end role
```

```
role session(
        A,G,C,S                 : agent,
        Kag,Kgs                 : symmetric_key,
        LS                      : text set,
        Hash                    : function,
```

```
        U                          : text,
        Kca,Pka                    : public_key)
def=

  local
        SendC,ReceiveC             : channel (dy),
        SendS,ReceiveS             : channel (dy),
        SendG,ReceiveG             : channel (dy),
        SendA,ReceiveA             : channel (dy)

  composition
           client(C,G,S,A,SendC,ReceiveC,Kca,Pka,U,Hash)
      /\   server(S,C,G,Kgs,SendS,ReceiveS,LS)
      /\   ticketGrantingServer(G,S,C,A,Kag,Kgs,SendG,ReceiveG,LS)
      /\   authenticationServer(A,C,G,Kca,Kag,SendA,ReceiveA,LS,Pka,Hash)

end role
```

```
role environment()
def=

  local LS : text set

  const a,g,c,s              : agent,
        k_gi,
        k_ag,k_gs            : symmetric_key,
        kia,kca,pka          : public_key,
        hash_                : function,
        u1,u2                : text,
        t0,t1,t2a,t2b,n1,n2 : protocol_id

  init LS = {}

  intruder_knowledge = {a,g,c,s,pka,hash_,k_gi,u1,u2,
                        kia,inv(kia)}

  composition
        session(a,g,c,s,k_ag,k_gs,LS,hash_,u1,kca,pka)
 /\     session(a,g,i,s,k_ag,k_gs,LS,hash_,u2,kia,pka)
```

```
end role
```

```
goal

  %secrecy_of Kcg,Kcs
  secrecy_of sec_a_Kcg,
             sec_t_Kcg, sec_t_Kcs,
             sec_s_Kcs,
             sec_c_Kcs,sec_c_Kcg

  %Client authenticates AuthenticationServer on n1
  authentication_on n1
  %Client authenticates TicketGrantingServer on n2
  authentication_on n2
  %Client authenticates Server on t2a
  authentication_on t2a
  %Server authenticates Client on t2b
  authentication_on t2b
  %TicketGrantingServer weakly authenticates Client on t1
  authentication_on t1
  %AuthenticationServer weakly authenticates Client on t0
  authentication_on t0

end goal
```

```
environment()
```

References