with forwardable ticket

Protocol Purpose

Mutual authentication

Definition Reference

- http://www.ietf.org/internet-drafts/draft-ietf-krb-wg-kerberos-clarifications-07.txt

Model Authors

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

Alice&Bob style

```
C -> A: U,G,N1
A -> C: U,Tcg,{G,Kcg,T1start,T1expire,N1}_Kca

where Tcg := {U,C,G,Kcg,T1start,T1expire}_Kag
      A := Authentication Server

C -> G: IP-ADDR,S,N2,Tcg,Acg,FORWARDABLE
G -> C: U,Tcs1,{S,Kcs,T2start,T2expire,N2}_Kcg

where Acg  := {C,T1}_Kcg  (T1 is a timestamp)
      Tcs1 := {IP-ADDR,U,C,S,Kcs,T2start,T2expire,FORWARDABLE}_Kgs

C -> G: IP-ADDR,S,N2,Tcs1,Acg
G -> C: U,Tcs2,{S,Kcs,T2start,T2expire,N2}_Kcg

where Acg  := {C,T1}_Kcg  (T1 is a timestamp)
      Tcs2 := {IP-ADDR,U,C,S,Kcs,T2start,T2expire,FORWARDABLE}_Kgs

C -> S: Tcs2,Acs
S -> C: {T2'}_Kcs
```

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

************************************************************************

 An alternative instance of the protocol in action.  The
 client does not request a forwardable ticket, and does not
 change IP address.

  C -> A: U,G,N1
  A -> C: U,Tcg,{G,Kcg,T1start,T1expire,N1}_Kca

  where Tcg := {U,C,G,Kcg,T1start,T1expire}_Kag
        A := Authentication Server

  C -> G: IP-ADDR,S,N2,Tcg,Acg,NOT_FORWARDABLE
  G -> C: U,Tcs1,{S,Kcs,T2start,T2expire,N2}_Kcg

  where Acg  := {C,T1}_Kcg  (T1 is a timestamp)
        Tcs1 := {IP-ADDR,U,C,S,Kcs,T2start,T2expire,NOT_FORWARDABLE}_Kgs

  C -> S: Tcs1,Acs
  S -> C: {T2'}_Kcs

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

Problems considered: 6

Attacks Found

None

Further Notes

- Same as plain Kerberos V except that if the client requests a forwardable ticket from the TGS, then sends this back to the TGS to get a ticket for a new IP address.
- IP address is a local nonce to client, and is included in requests and tickets.
- The IP address is also changed before requesting a new ticket, naturally.

HLPSL Specification

```
role authenticationServer(
    A,C,G    : agent,
    Kca,Kag   : symmetric_key,
    SND, RCV : channel(dy))
played_by A def=

  local
    State    : nat,
    N1       : text,
    U        : text,
    Kcg      : symmetric_key,
    T1start  : text,
    T1expire : text

  const sec_a_Kcg : protocol_id

  init
    State := 11

  transition

 1. State  = 11  /\ RCV(U'.G.N1') =|>
    State' := 12 /\ Kcg' := new()
                 /\ T1start' := new()
                 /\ T1expire' := new()
                 /\ SND(U'.{U'.C.G.Kcg'.T1start'.T1expire'}_Kag.
                        {G.Kcg'.T1start'.T1expire'.N1'}_Kca       )
                 /\ witness(A,C,n1,N1')
                 /\ 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  : text,
    T2start  : text,
    T1expire : text,
    T2expire : text,
    T1       : text,
    IP_ADDR  : text,
    Forwardable_or_not  : protocol_id

  const  forwardable,
         sec_t_Kcg,
         sec_t_Kcs   : protocol_id

  init   State := 21

  transition

 1. State  = 21
      /\ RCV(IP_ADDR'.S.N2'.
             {U'.C.G.Kcg'.T1start'.T1expire'}_Kag.
             {C.T1'}_Kcg'.
             Forwardable_or_not')
      %% T1' should not have been received before
      /\ not(in(T1',L))
    =|>
    State' := 22
      /\ Kcs' := new()
      /\ T2start' := new()
      /\ T2expire' := new()
      /\ SND(U'.
```

```
              {IP_ADDR'.U'.C.S.Kcs'.T2start'.T2expire'.Forwardable_or_not'}_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})

 3. State  = 22
      /\ RCV(IP_ADDR.S.N2.
              {IP_ADDR.U.C.S.Kcs.T2start.T2expire.forwardable}_Kgs.
              {C.T1}_Kcg)
      /\ Forwardable_or_not = forwardable
    =|>
    State' := 23
      /\ SND(U.
              {IP_ADDR.U.C.S.Kcs.T2start.T2expire.forwardable}_Kgs.
              {S.Kcs.T2start.T2expire.N2}_Kcg)

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,
    IP_ADDR : text,
    Forwardable_or_not : protocol_id

  const sec_s_Kcs : protocol_id
```

```
  init  State := 31

  transition

 1. State  = 31
      /\ RCV({IP_ADDR'.U'.C.S.Kcs'.T2start'.T2expire'.Forwardable_or_not'}_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,
    U               : text,
    Kca             : symmetric_key,
    SND,RCV         : channel(dy))
played_by C def=

  local
    State   : nat,
    Kcs     : symmetric_key,
    T1expire: text,
    T2expire: text,
    T1start : text,
    T2start : text,
    Kcg     : symmetric_key,
    T1,T2   : text,
    IP_ADDR : text,
    Tcg     : {text.agent.agent.symmetric_key.text.text}_symmetric_key,
    Tcs1, Tcs2:
     {text.text.agent.agent.symmetric_key.text.text.protocol_id}_symmetric_key,
    N1, N2  : text
```

```
  const forwardable,
        un_forwardable : protocol_id,
        sec_c_Kcg1,
        sec_c_Kcg2,
        sec_c_Kcs      : protocol_id

  init  State := 1

  transition

 1. State  = 1  /\ RCV(start)  =|>
    State' := 2 /\ N1' := new()
                /\ SND(U.G.N1')

21. State  = 2  /\ RCV(U.Tcg'.{G.Kcg'.T1start'.T1expire'.N1}_Kca) =|>
    State' := 3 /\ N2' := new()
                /\ T1' := new()
                /\ IP_ADDR' := new()
                /\ SND(IP_ADDR'.S.N2'.Tcg'.{C.T1'}_Kcg'.forwardable)
                /\ witness(C,G,t1,T1')
                /\ request(C,A,n1,N1)
                /\ secret(Kcg',sec_c_Kcg1,{A,C,G})

22. State  = 2  /\ RCV(U.Tcg'.{G.Kcg'.T1start'.T1expire'.N1}_Kca) =|>
    State' := 4 /\ SND(IP_ADDR'.S.N2'.Tcg'.{C.T1'}_Kcg'.un_forwardable)
                /\ witness(C,G,t1,T1')
                /\ request(C,A,n1,N1)
                /\ secret(Kcg',sec_c_Kcg2,{A,C,G})

 3. State  = 3  /\ RCV(U.Tcs1'.{S.Kcs'.T2start'.T2expire'.N2}_Kcg) =|>
    State' := 4 /\ SND(IP_ADDR.S.N2.Tcs1'.{C.T1}_Kcg)
                /\ request(C,G,n2,N2)
                /\ secret(Kcs',sec_c_Kcs,{G,C,S})

 4. State  = 4  /\ RCV(U.Tcs2'.{S.Kcs'.T2start.T2expire.N2}_Kcg) =|>
    State' := 5 /\ T2' := new()
                /\ SND(Tcs2'.{C.T2'}_Kcs')
                /\ witness(C,S,t2b,T2')

 5. State  = 5  /\ RCV({T2}_Kcs) =|>
    State' := 6 /\ request(C,S,t2a,T2)
```

```
end role
```

```
role session(
    A,G,C,S                  : agent,
    U                        : text,
    Kca,Kgs,Kag              : symmetric_key,
    LS,LG                    : text set) def=

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

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

end role
```

```
role environment() def=

  local LS, LG : text set

  const
    a,g,c,s                      : agent,
    u1,u2                        : text,
    k_ca,k_gs,k_ag,k_ia          : symmetric_key,
    t1,t2a,t2b,n1,n2             : protocol_id,
    forwardable, un_forwardable  : protocol_id

  init LS = {} /\ LG = {}

  intruder_knowledge = {a,g,c,s,k_ia,forwardable,u1,u2
```

```
                    }

  composition

        session(a,g,c,s,u1,k_ca,k_gs,k_ag,LS,LG)
   /\   session(a,g,i,s,u2,k_ia,k_gs,k_ag,LS,LG)

end role
```

```
goal

  %secrecy_of Kcg, Kcs
  secrecy_of sec_a_Kcg,
             sec_t_Kcg,sec_t_Kcs,
             sec_s_Kcs,
             sec_c_Kcg1,sec_c_Kcg2,sec_c_Kcs

  %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

end goal
```

```
environment()
```

References