TLS: Transport Layer Security

Protocol Purpose

TLS is intended to provide privacy and data integrity of communication over the Internet.

Definition Reference

- [DA99, Pau99]

Model Authors

- Paul Hankes Drielsma, ETH Zürich, November 2003

Alice&Bob style

The protocol proceeds between a client A and a server B with respective public keys Ka and Kb. These two agents generate nonces Na and Nb, respectively. In addition, we assume the existence of a trusted third party (in essence, a certificate authority) S whose public key is Ks. The agents possess certificates of the form {X,Kx}inv(Ks). Each session is identified by a unique ID Sid. The protocol also makes use of a pseudo-random number generator PRF which we model as a hash function.

```
 0. A -> B: A, Na, Sid, Pa         where Pa is a cryptosuite offer
 1. B -> A: Nb, Sid, Pb where Pb is B's counteroffer
 2. B -> A: {B, Kb}inv(Ks) optional certificate exchange
 3. A -> B: {A, Ka}inv(Ks) optional certificate exchange
 4. A -> B: {PMS}Kb where PMS is a nonce generated by A
 5. A -> B: {H(Nb,B,PMS)}inv(Ka) optional certificate verify message
 6. A -> B: {Finished}Keygen(A, Na, Nb, M)
  where  M = PRF(PMS,Na,Nb)
Finished = H(M,messages) for all messages 0 - 5
 7. B -> A: {Finished}Keygen(B, Na, Nb, M)
```

Note that Paulson leaves messages 2., 3., and 5. as optional. We include them in this model. Note also that in order to minimize the number of transitions specified, we have combined the sending of messages 1. and 2. as well as the sending of messages 3. 4. 5. and 6. into single transitions.

Model Limitations

This formalisation is based on the abstracted version of TLS presented by Paulson in [Pau99]. In addition to the abstractions made in this paper, we further abstract away from the negotiation of cryptographic algorithms. Our model assumes that one offer for a crypto suite is made and only that offer will be accepted. This may exclude cipher-suite rollback attacks like the one that was possible on SSLv2.

Problems considered: 3

Attacks Found

None

HLPSL Specification

```
role alice(A, B : agent,
           H, PRF, KeyGen: function,
           Ka, Ks: public_key,  %% Ks is the public key of a T3P (ie. CA)
           SND, RCV: channel (dy))
played_by A
def=

   local Na, Sid, Pa, PMS: text,
         Nb: text,
         State: nat,
         Finished, ClientK, ServerK: message,
         Kb: public_key,
         M: message

   const sec_clientk, sec_serverk : protocol_id

   init  State := 0

   transition

   1.  State = 0
```

```
       /\ RCV(start)
       =|>
       State' := 2
       /\ Na' := new()
       /\ Pa' := new()
       /\ Sid' := new()
       /\ SND(A.Na'.Sid'.Pa')

   % Since we abstract away from the negotiation
   % of cryptographic algorithms, here I simply assume
   % that the server must send back Pa.  (Essentially
   % modelling that the client makes only one offer.)

   2.  State = 2
       /\ RCV(Nb'.Sid.Pa.{B.Kb'}_(inv(Ks)))
       =|>
       State' := 3
       /\ PMS' := new()
       /\ M' := PRF(PMS'.Na.Nb')
       /\ Finished' = H(PRF(PMS'.Na.Nb').A.B.Na.Pa.Sid)
       /\ ClientK' = KeyGen(A.Na.Nb'.PRF(PMS'.Na.Nb'))
       /\ ServerK' = KeyGen(B.Na.Nb'.PRF(PMS'.Na.Nb'))
       /\ SND({PMS'}_Kb'.
              {A.Ka}_(inv(Ks)).
              {H(Nb'.B.PMS')}_(inv(Ka)).
              {H(PRF(PMS'.Na.Nb').
               A.B.Na.Pa.Sid)
              }_KeyGen(A.Na.Nb'.PRF(PMS'.Na.Nb')))
       /\ witness(A,B,na_nb2,Na.Nb')

   4.  State = 3
       /\ RCV({Finished}_ServerK)
       =|>
       State' := 5
       /\ request(A,B,na_nb1,Na.Nb)
       /\ secret(ClientK,sec_clientk,{A,B})
       /\ secret(ServerK,sec_serverk,{A,B})

end role
```

```
role bob(A, B : agent,
         H, PRF, KeyGen: function,
         Kb, Ks: public_key,
         SND, RCV: channel (dy))
played_by B
def=

   local Na, Nb, Sid, Pa, PMS: text,
         State: nat,
         Ka: public_key

   init  State := 1

   transition

   1.  State = 1
       /\ RCV(A.Na'.Sid'.Pa')
       =|>
       State' := 3
       /\ Nb' := new()
       /\ SND(Nb'.Sid'.Pa'.{B.Kb}_(inv(Ks)))
       /\ witness(B,A,na_nb1,Na'.Nb')

   2.  State = 3
       /\ RCV({PMS'}_Kb.{A.Ka'}_(inv(Ks)).
              {H(Nb.B.PMS')}_(inv(Ka')).
              {H(PRF(PMS'.Na.Nb).
               A.B.Na.Pa.Sid)
              }_KeyGen(A.Na.Nb.PRF(PMS'.Na.Nb)))
       =|>
       State' := 5
       /\ SND({H(PRF(PMS'.Na.Nb).
               A.B.Na.Pa.Sid)
              }_KeyGen(B.Na.Nb.PRF(PMS'.Na.Nb)))
       /\ request(B,A,na_nb2,Na.Nb)

end role
```

```
role session(A,B: agent,
             Ka, Kb, Ks: public_key,
             H, PRF, KeyGen: function)
def=

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

   composition
                alice(A,B,H,PRF,KeyGen,Ka,Ks,SA,RA)
           /\   bob(A,B,H,PRF,KeyGen,Kb,Ks,SB,RB)

end role
```

```
role environment()
def=

   const na_nb1, na_nb2 : protocol_id,
         h, prf, keygen : function,
         a, b           : agent,
         ka, kb, ki, ks : public_key

   intruder_knowledge = { a, b, ka, kb, ks, ki, inv(ki),
                          {i.ki}_(inv(ks)) }

   composition
        session(a,b,ka,kb,ks,h,prf,keygen)
    /\  session(a,i,ka,ki,ks,h,prf,keygen)
    /\  session(i,b,ki,kb,ks,h,prf,keygen)

end role
```

```
goal

        secrecy_of sec_clientk,sec_serverk
        %Alice authenticates Bob on na_nb1
        authentication_on na_nb1
        %Bob authenticates Alice on na_nb2
        authentication_on na_nb2

end goal
```

```
environment()
```

References

[DA99] T. Dierks and C. Allen. RFC 2246: The TLS Protocol Version 1.0, January 1999. Status: Proposed Standard.

[Pau99] Lawrence C. Paulson. Inductive analysis of the internet protocol TLS. *ACM Transactions on Computer and System Security*, 2(3):332–351, 1999.