Lab 7 Report

CSL 6010 - Cyber Security

Rahul Barodia

B20CS047

Task :

To define three agents: A, B, and a Trusted Third Party T

A and T share a symmetric key ANT

B and T share a symmetric key BAT

A wants to establish a symmetric session key AKB shared with B

$$A \rightarrow T \rightarrow B$$

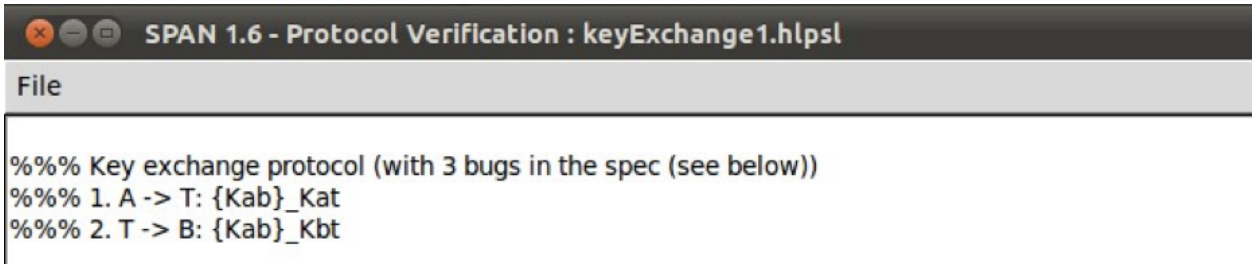
Let A and T share symmetric key Kat

B and T share symmetric key Kbt

And A wants to share the symmetric key Kab with B

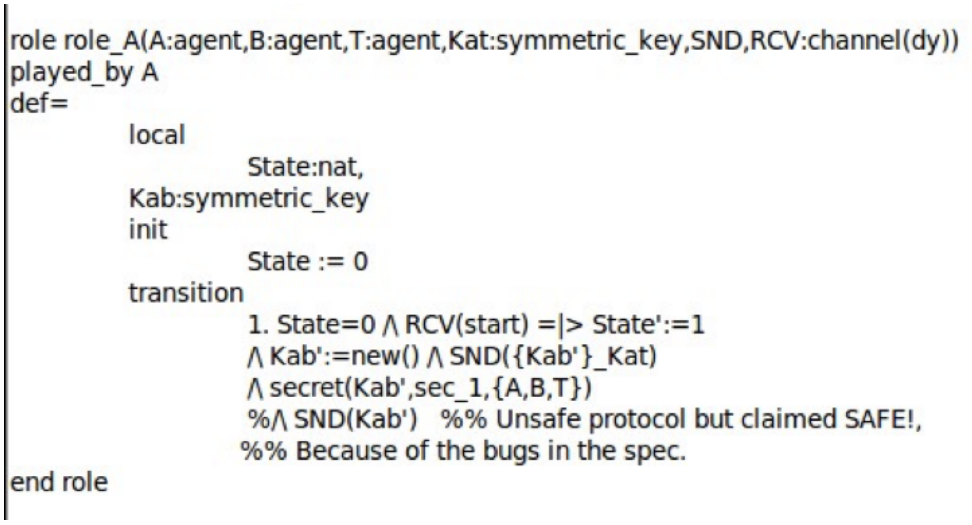
1) Specifying Protocol and Properties

Using keyExchange1.hlpsl :



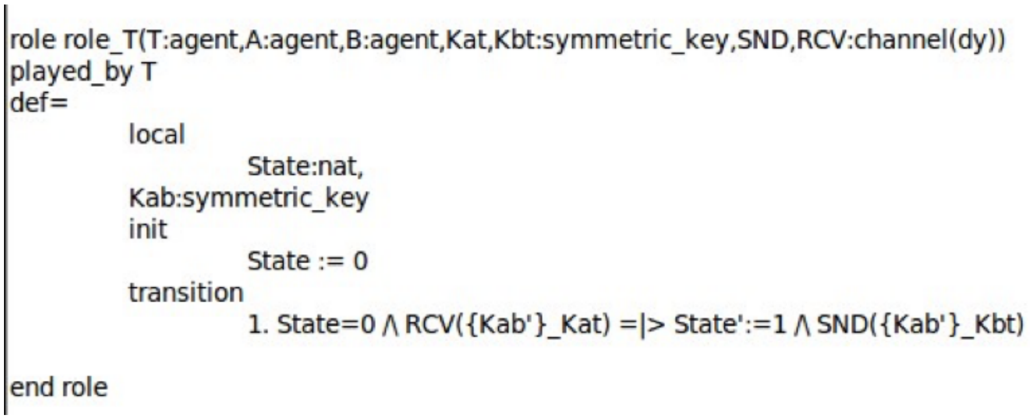
```
%%% Key exchange protocol (with 3 bugs in the spec (see below))
%%% 1. A -> T: {Kab}_Kat
%%% 2. T -> B: {Kab}_Kbt
```

A sends T the symmetric key Kab encrypted with the key Kat. Then T sends the symmetric key to be transferred Kab to B but here also the the key Kab is encrypted with the key Kbt.



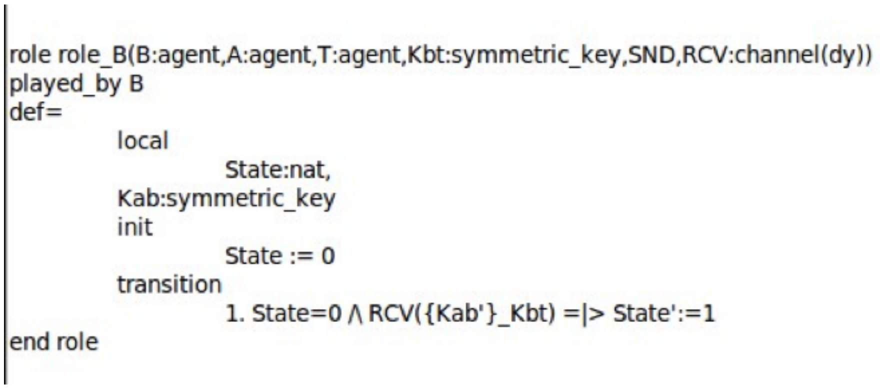
```
role role_A(A:agent,B:agent,T:agent,Kat:symmetric_key,SND,RCV:channel(dy))
played_by A
def=
            local
                        State:nat,
            Kab:symmetric_key
            init
                        State := 0
            transition
                        1. State=0 /\ RCV(start) =|> State':=1
                        /\ Kab':=new() /\ SND({Kab'}_Kat)
                        /\ secret(Kab',sec_1,{A,B,T})
                        %/\ SND(Kab')   %% Unsafe protocol but claimed SAFE!,
                       %% Because of the bugs in the spec.
end role
```

The above code is for the role of A. A creates a new symmetric key Kab’ and sends it to T upon encrypting it with the key Kat.



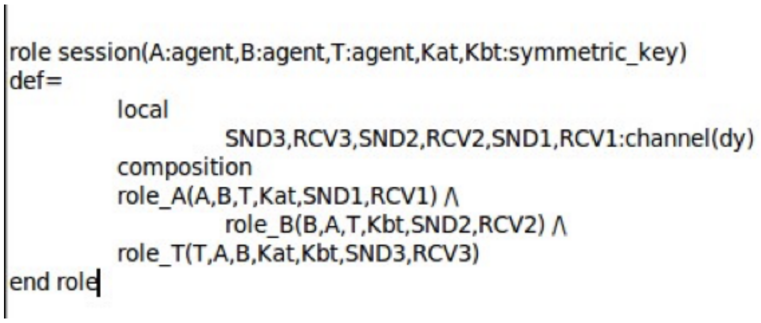
```
role role_T(T:agent,A:agent,B:agent,Kat,Kbt:symmetric_key,SND,RCV:channel(dy))
played_by T
def=
            local
                        State:nat,
            Kab:symmetric_key
            init
                        State := 0
            transition
                        1. State=0 /\ RCV({Kab'}_Kat) =|> State':=1 /\ SND({Kab'}_Kbt)

end role
```

The above code is for the role of Trusted Third Party T. T sends the Kab' received from A to B upon encrypting it with the key Kbt.



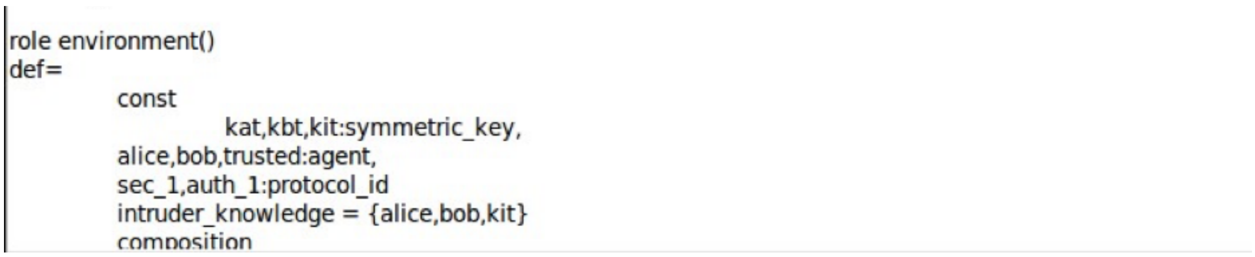
```
role role_B(B:agent,A:agent,T:agent,Kbt:symmetric_key,SND,RCV:channel(dy))
played_by B
def=
        local
                State:nat,
        Kab:symmetric_key
        init
                State := 0
        transition
                1. State=0 ∧ RCV({Kab'}_Kbt) =|> State':=1
end role
```

B receives the msg received from trusted third party T.

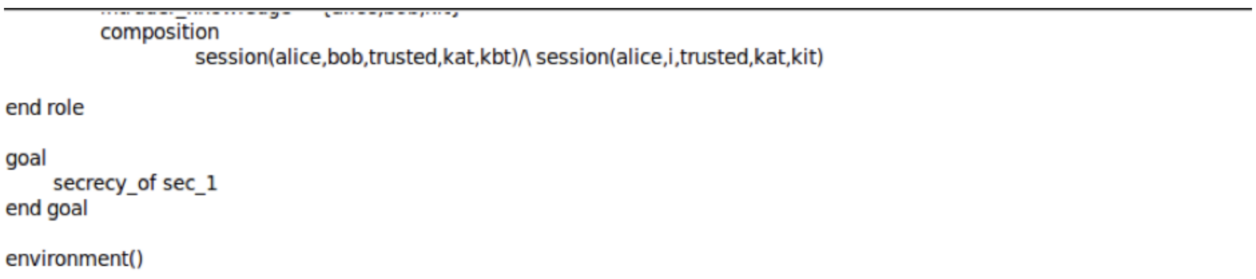


```
role session(A:agent,B:agent,T:agent,Kat,Kbt:symmetric_key)
def=
        local
                SND3,RCV3,SND2,RCV2,SND1,RCV1:channel(dy)
        composition
        role_A(A,B,T,Kat,SND1,RCV1) ∧
                role_B(B,A,T,Kbt,SND2,RCV2) ∧
        role_T(T,A,B,Kat,Kbt,SND3,RCV3)
end role
```

The role_A takes as input the channels SND1 and RCV1, and it interacts with agents B and T using the symmetric key Kat. Similarly, the role_B takes as input the channels SND2 and RCV2, and it interacts with agents A and T using the symmetric key Kbt. Finally, the role_T takes as input the channels SND3 and RCV3, and thus the symmetric key Kab is transferred between A and B with the help of trusted third party T.

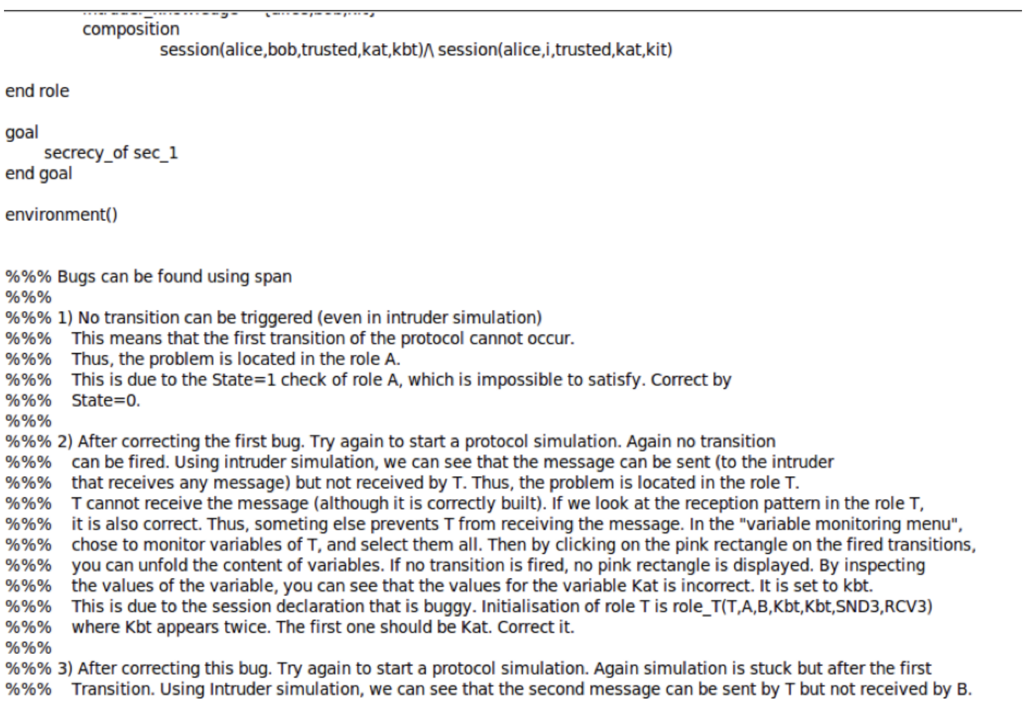


```
role environment()
def=
            const
                        kat,kbt,kit:symmetric_key,
            alice,bob,trusted:agent,
            sec_1,auth_1:protocol_id
            intruder_knowledge = {alice,bob,kit}
            composition
```



```
            composition
                        session(alice,bob,trusted,kat,kbt)/\ session(alice,i,trusted,kat,kit)

end role

goal
     secrecy_of sec_1
end goal

environment()
```

The environment defines three agents: Alice, Bob, and Trusted, and it defines the intruder's knowledge, which includes Alice, Bob, and the symmetric key kit.Finally, a security goal is defined using the "goal" construct in SPAN. .The goal specifies that the protocol should not reveal any information that is supposed to be kept secret, such as the shared session key or the symmetric keys kat and kbt.

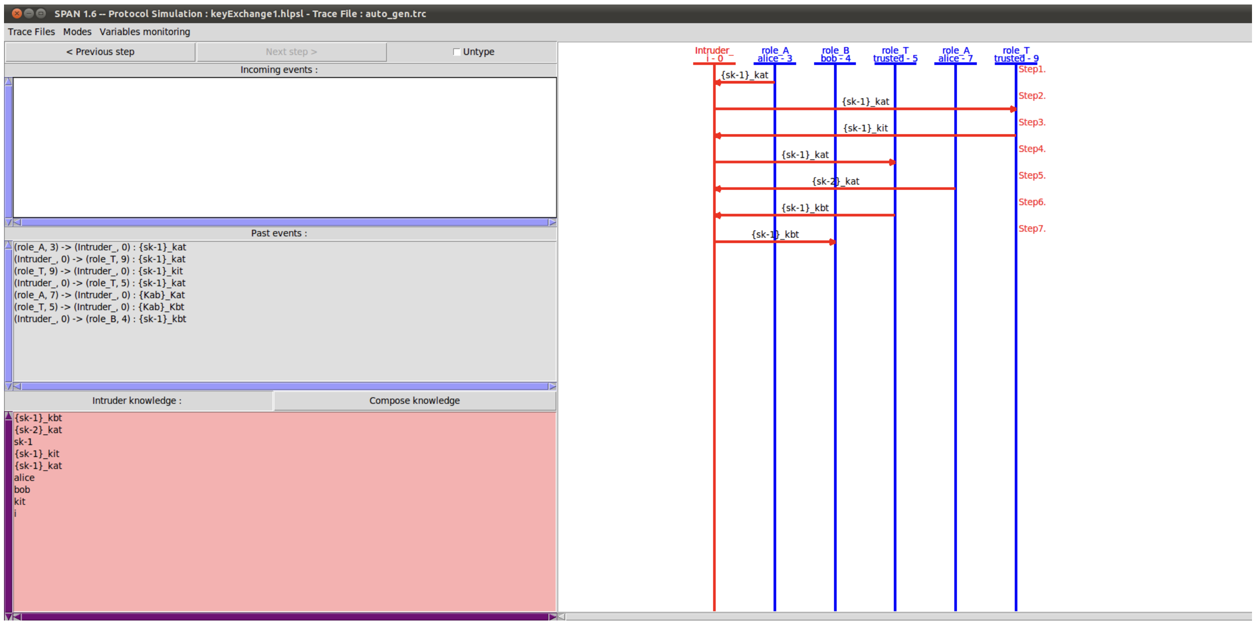


```
          composition
                    session(alice,bob,trusted,kat,kbt)/\ session(alice,i,trusted,kat,kit)

end role

goal
     secrecy_of sec_1
end goal

environment()
```

```
%%% Bugs can be found using span
%%%
%%% 1) No transition can be triggered (even in intruder simulation)
%%%    This means that the first transition of the protocol cannot occur.
%%%    Thus, the problem is located in the role A.
%%%    This is due to the State=1 check of role A, which is impossible to satisfy. Correct by
%%%    State=0.
%%%
%%% 2) After correcting the first bug. Try again to start a protocol simulation. Again no transition
%%%    can be fired. Using intruder simulation, we can see that the message can be sent (to the intruder
%%%    that receives any message) but not received by T. Thus, the problem is located in the role T.
%%%    T cannot receive the message (although it is correctly built). If we look at the reception pattern in the role T,
%%%    it is also correct. Thus, someting else prevents T from receiving the message. In the "variable monitoring menu",
%%%    chose to monitor variables of T, and select them all. Then by clicking on the pink rectangle on the fired transitions,
%%%    you can unfold the content of variables. If no transition is fired, no pink rectangle is displayed. By inspecting
%%%    the values of the variable, you can see that the values for the variable Kat is incorrect. It is set to kbt.
%%%    This is due to the session declaration that is buggy. Initialisation of role T is role_T(T,A,B,Kbt,Kbt,SND3,RCV3)
%%%    where Kbt appears twice. The first one should be Kat. Correct it.
%%%
%%% 3) After correcting this bug. Try again to start a protocol simulation. Again simulation is stuck but after the first
%%%    Transition. Using Intruder simulation, we can see that the second message can be sent by T but not received by B.
```

2) Debugging specification using animation: Find the blocking transition, monitor theVariables

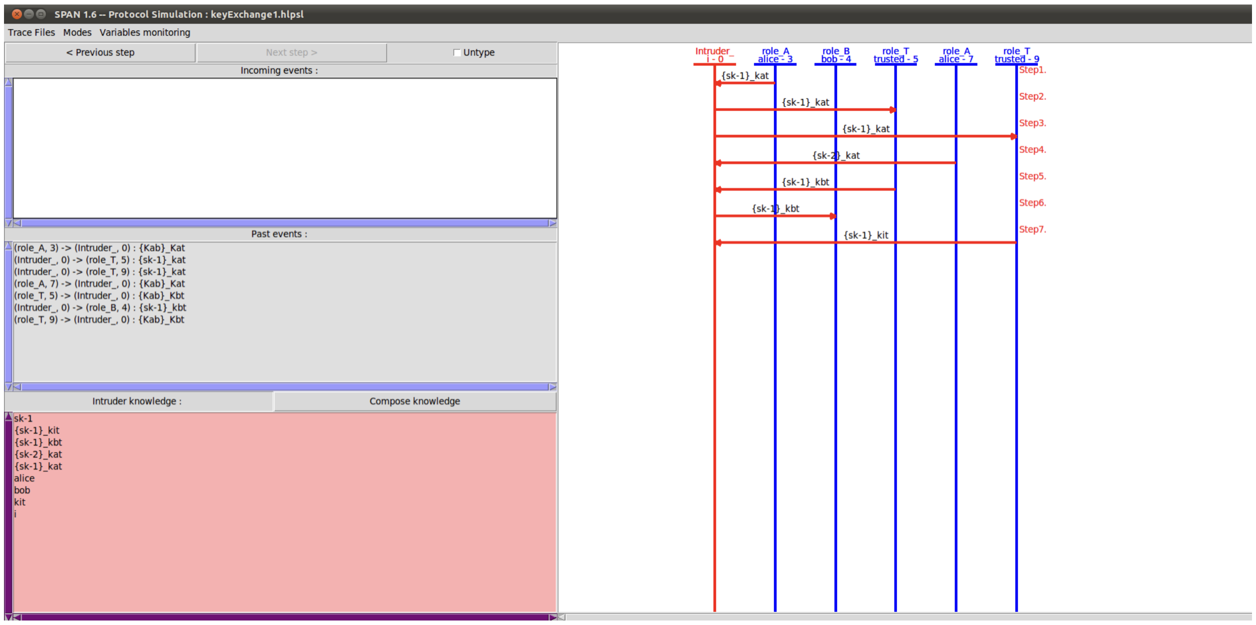
Attacker :



```
(role_A, 3) -> (Intruder_, 0) : {sk-1}_kat
(Intruder_, 0) -> (role_T, 9) : {sk-1}_kat
(role_T, 9) -> (Intruder_, 0) : {sk-1}_kit
(Intruder_, 0) -> (role_T, 5) : {sk-1}_kat
(role_A, 7) -> (Intruder_, 0) : {Kab}_Kat
(role_T, 5) -> (Intruder_, 0) : {Kab}_Kbt
(Intruder_, 0) -> (role_B, 4) : {sk-1}_kbt
```

```
{sk-1}_kbt
{sk-2}_kat
sk-1
{sk-1}_kit
{sk-1}_kat
alice
bob
kit
i
```

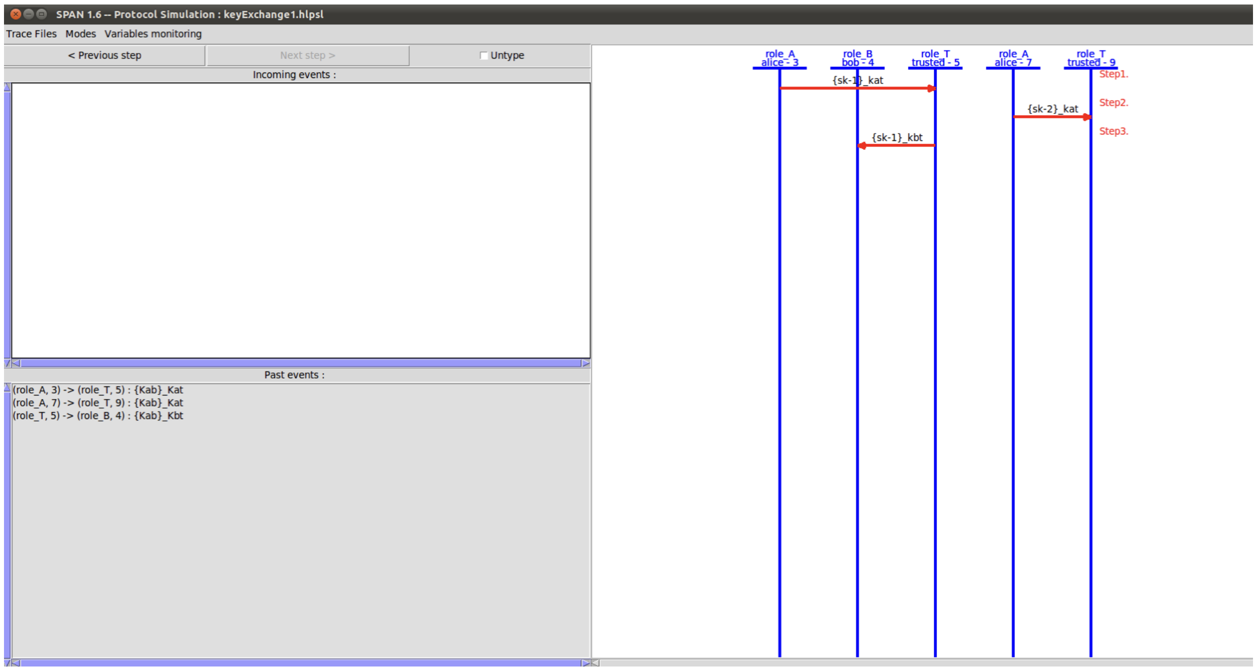
Intruder :



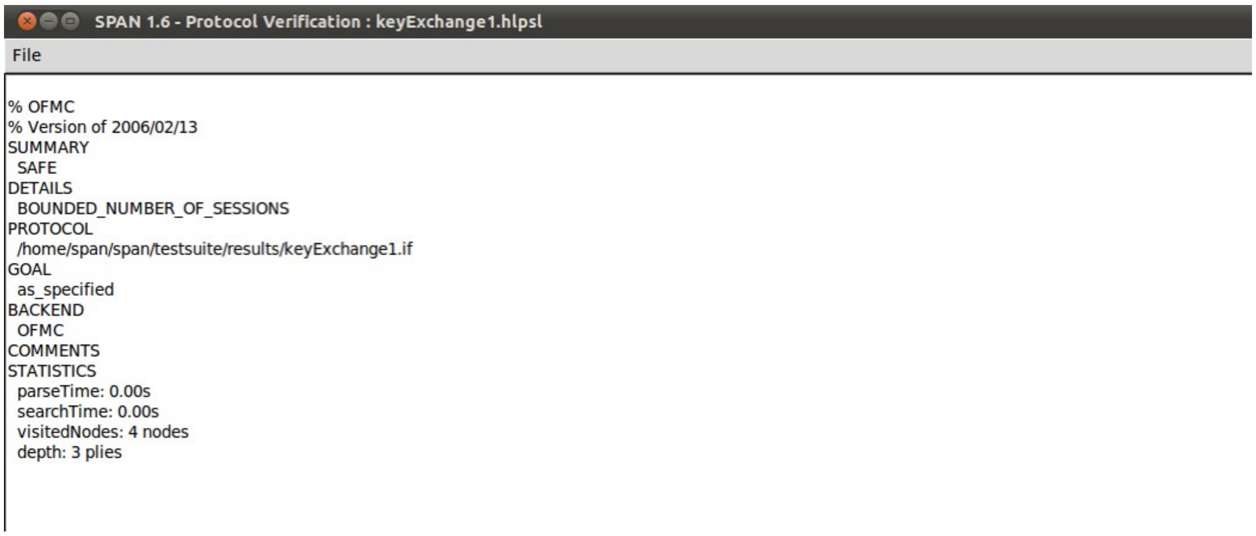
```
(role_A, 3) -> (Intruder_, 0) : {Kab}_Kat
(Intruder_, 0) -> (role_T, 5) : {sk-1}_kat
(Intruder_, 0) -> (role_T, 9) : {sk-1}_kat
(role_A, 7) -> (Intruder_, 0) : {Kab}_Kat
(role_T, 5) -> (Intruder_, 0) : {Kab}_Kbt
(Intruder_, 0) -> (role_B, 4) : {sk-1}_kbt
(role_T, 9) -> (Intruder_, 0) : {Kab}_Kbt
```

```
sk-1
{sk-1}_kit
{sk-1}_kbt
{sk-2}_kat
{sk-1}_kat
alice
bob
kit
i
```

Protocol :



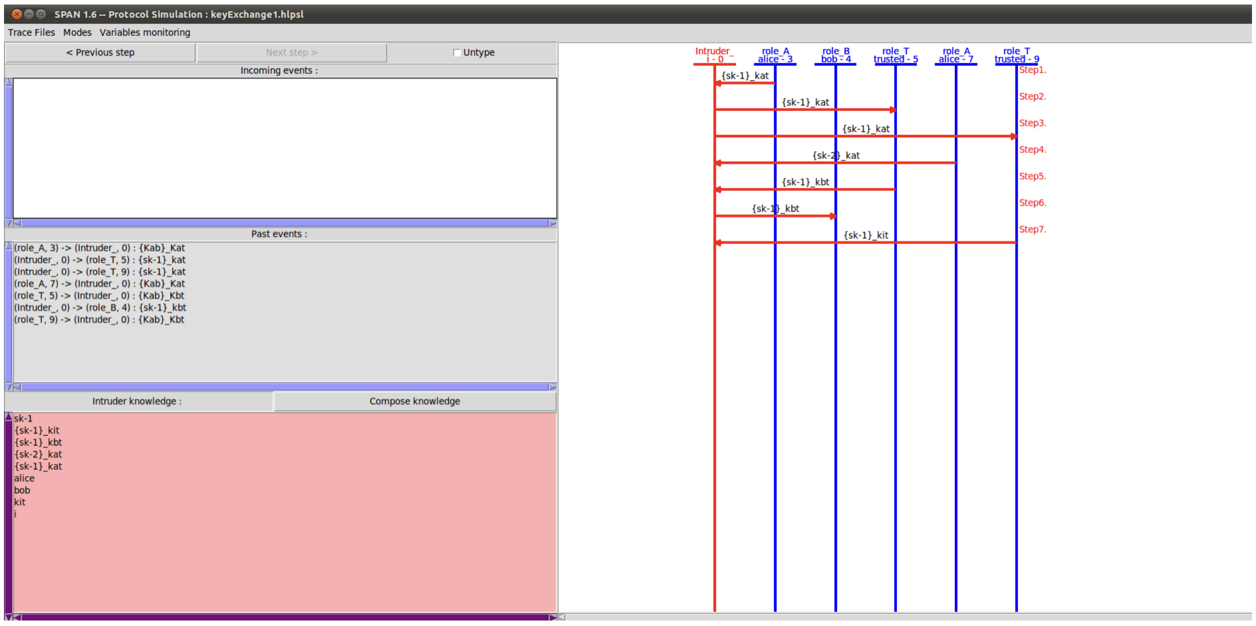
```
(role_A, 3) -> (role_T, 5) : {Kab}_Kat
(role_A, 7) -> (role_T, 9) : {Kab}_Kat
(role_T, 5) -> (role_B, 4) : {Kab}_Kbt
```



```
% OFMC
% Version of 2006/02/13
SUMMARY
  SAFE
DETAILS
  BOUNDED_NUMBER_OF_SESSIONS
PROTOCOL
  /home/span/span/testsuite/results/keyExchange1.if
GOAL
  as_specified
BACKEND
  OFMC
COMMENTS
STATISTICS
  parseTime: 0.00s
  searchTime: 0.00s
  visitedNodes: 4 nodes
  depth: 3 plies
```

We can see from the above screenshot that our protocol keyExchange1 is SAFE.

3) Attack discovery, strengthening the protocol



```
(role_A, 3) -> (Intruder_, 0) : {Kab}_Kat
(Intruder_, 0) -> (role_T, 5) : {sk-1}_kat
(Intruder_, 0) -> (role_T, 9) : {sk-1}_kat
(role_A, 7) -> (Intruder_, 0) : {Kab}_Kat
(role_T, 5) -> (Intruder_, 0) : {Kab}_Kbt
(Intruder_, 0) -> (role_B, 4) : {sk-1}_kbt
(role_T, 9) -> (Intruder_, 0) : {Kab}_Kbt
```

```
sk-1
{sk-1}_kit
{sk-1}_kbt
{sk-2}_kat
{sk-1}_kat
alice
bob
kit
i
```

Protocol strengthening can be done by adding more security mechanisms, repeating the process ,etc.

4)Tuning and optimizing the protocol

Optimization of the protocol can be done by reducing the number of messages excchnages, using caching and pre computation,etc.

Following is the optimized code:

```
role role_A(A:agent,B:agent,T:agent,Kat:symmetric_key,SND,RCV:channel(dy))

played_by A

def=

        local

                State:nat,

         Na,Nb:text,

         Kab:symmetric_key

        init

                State := 0

        transition

                1. State=0 /\ RCV(start) =|>

          State':=1 /\ Na':=new() /\ Kab':=new() /\ SND({B.Kab'}_Kat) /\
secret(Kab',sec_1,{A,B,T})


                2. State=1 /\ RCV({B.Nb'}_Kab) =|> State':=2  /\ SND({Nb'}_Kab)


          %% A checks that B uses the same key

          %% that he sent at step 1.

          /\ request(A,B,auth_1,Kab)


          %% A hopes that Nb will permit to authenticate him

          /\ witness(A,B,auth_2,Nb')
```

```
end role

role role_T(T:agent,A:agent,B:agent,Kat,Kbt:symmetric_key,SND,RCV:channel(dy))
played_by T
def=
          local
                    State:nat,Na:text,Kab:symmetric_key
          init
                    State := 0
          transition
                    1. State=0 /\ RCV({B.Kab'}_Kat) =|>

            State':=1 /\ SND({A.Kab'}_Kbt)
end role



role role_B(B:agent,A:agent,T:agent,Kbt:symmetric_key,SND,RCV:channel(dy))
played_by B
def=
          local
                    State:nat,Na,Nb:text,Kab:symmetric_key
          init
                    State := 0
          transition
                    1. State=0 /\ RCV({A.Kab'}_Kbt) =|>

            State':=1 /\ Nb':= new() /\ SND({B.Nb'}_Kab')
```

```
        %% B hopes that Kab will permit to authenticate him
        /\ witness(B,A,auth_1,Kab')

       2. State=1 /\ RCV({Nb}_Kab) =|> State':=2

       %% B checks that he receives the same nonce
       %% that he sent at step 1.

       /\ request(B,A,auth_2,Nb)
end role




role session(A:agent,B:agent,T:agent,Kat,Kbt:symmetric_key)
def=
       local
              SND3,RCV3,SND2,RCV2,SND1,RCV1:channel(dy)
       composition
       role_A(A,B,T,Kat,SND1,RCV1) /\
              role_B(B,A,T,Kbt,SND2,RCV2) /\
       role_T(T,A,B,Kat,Kbt,SND3,RCV3)
end role

role environment()
def=
       const
              kat,kbt,kit:symmetric_key,   %% we add a symmetric key: kit shared
between the intruder and T
```

```
        alice,bob,trusted:agent,
        sec_1,auth_1,auth_2:protocol_id
        intruder_knowledge = {alice,bob,kit}   %% ... and we give it to the intruder
        composition
           %% We run the regular session
               session(alice,bob,trusted,kat,kbt)
           %% in parallel with another regular session
         /\ session(alice,bob,trusted,kat,kbt)

           %% and a session between the intruder (with key kit) and bob
         /\ session(i,bob,trusted,kit,kbt)
           %% and a session between alice and the intruder (with key kit)
         /\ session(alice,i,trusted,kat,kit)
end role

goal
        secrecy_of sec_1
    authentication_on auth_1
    authentication_on auth_2
end goal

environment()
```