Cyber Security

Mruganshi Gohel

B20CS014

Question- 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. Protocol development using SPAN+AVISPA:

1 - Specifying protocol and properties
2 - Debugging specification using animation: Find the blocking transition, monitor the
variables
3 - Attack discovery, strengthening the protocol
4 - Tuning and optimizing the protocol

We have to define a simple protocol with SPAN+AVISPA, in which there are three agents A, B, and trusted third-party T. In which A and T are going to share a symmetric key K_AT, B, and T are going to share a symmetric key K_BT and the objective of this protocol is to establish a symmetric session key K_AB shared by A and B. Now to develop the protocol we will test three protocols and every time we will try to optimize that protocol and jump to a new protocol that will be an optimized version of previous one.

1. **keyexchangeprotocol_1**

   So first we will go to the key exchange protocol in the RESSI_tutorial part of the SPAN VirtualBox and will open keyexchangeprotocol_1.

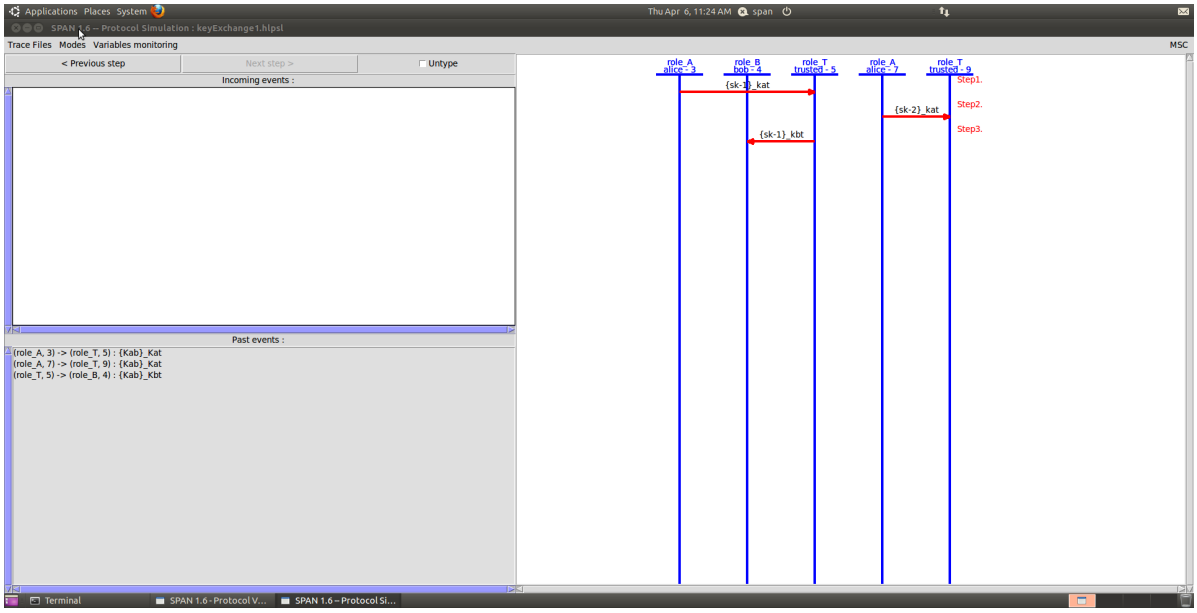
   But when we execute it using ATSE it shows that the protocol is unsafe, below result is attached along with hlpsl code.

   - **Specifying protocol and properties**

1. **A -> T: {Kab}_Kat**

2. **T -> B: {Kab}_Kbt**

Protocol simulation:



- **Debugging specification using animation: Find the blocking transition, monitor the variables**

A is going to sends a key to T, he generates a key K_AB and is going to cipher it with that share symmetric key K_AT and then T receive it , he can decrypt because he has the key K_AT and then cipher it with K_AB and sends it to B.

So A has a symmetric key K_AT, T has two symmetric keys K_AT, K_BT, and B has of course one symmetric key that is K_BT.

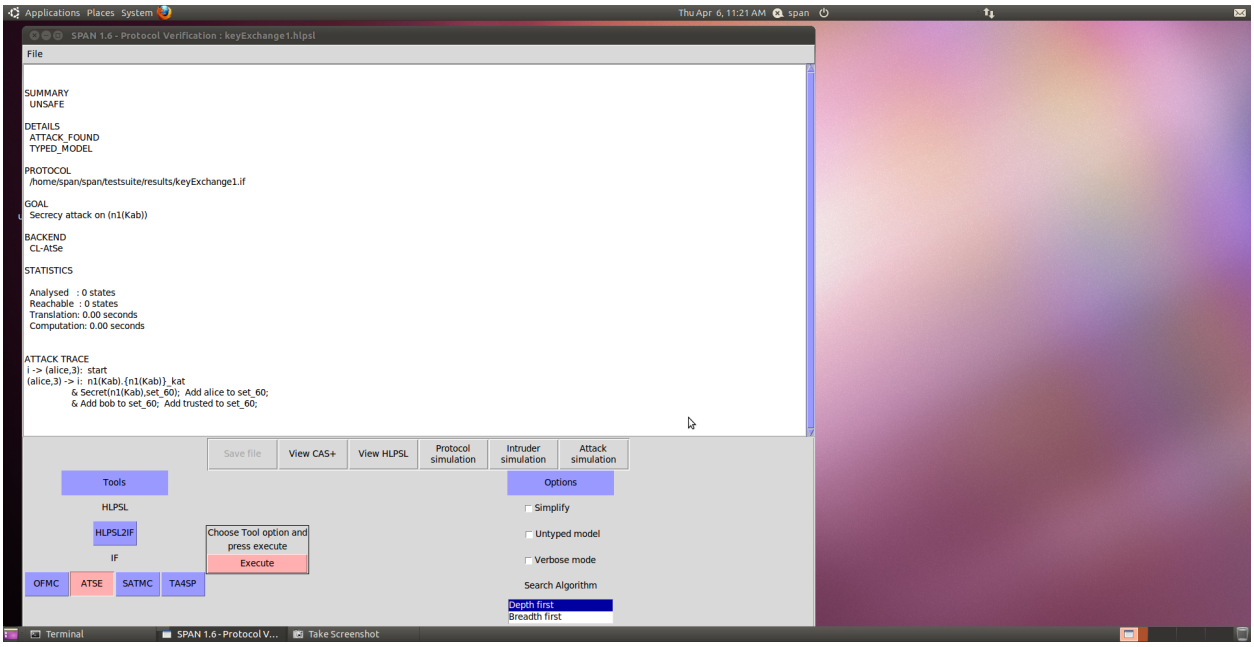
```
role role_A(A:agent,B:agent,T:agent,Kat:symmetric_key,SND,RCV:channel(dy))
played_by A
def=
local
State:nat,
                Kab:symmetric_key
init
State := 0
transition
1. State=0 /\ RCV(start) =|> State':=1
/\ Kab':=new() /\ SND({Kab'}_Kat)
```

```
/\ secret(Kab',sec_1,{A,B,T})
/\ SND(Kab')   %% Unsafe protocol but claimed SAFE!,
                                %% Because of the bugs in the spec.
end role

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


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


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,
                alice,bob,trusted:agent,
                sec_1,auth_1:protocol_id
intruder_knowledge = {alice,bob,kit}
composition
session(alice,bob,trusted,kat,kbt) /\
session(alice,i,trusted,kat,kit)

end role
```

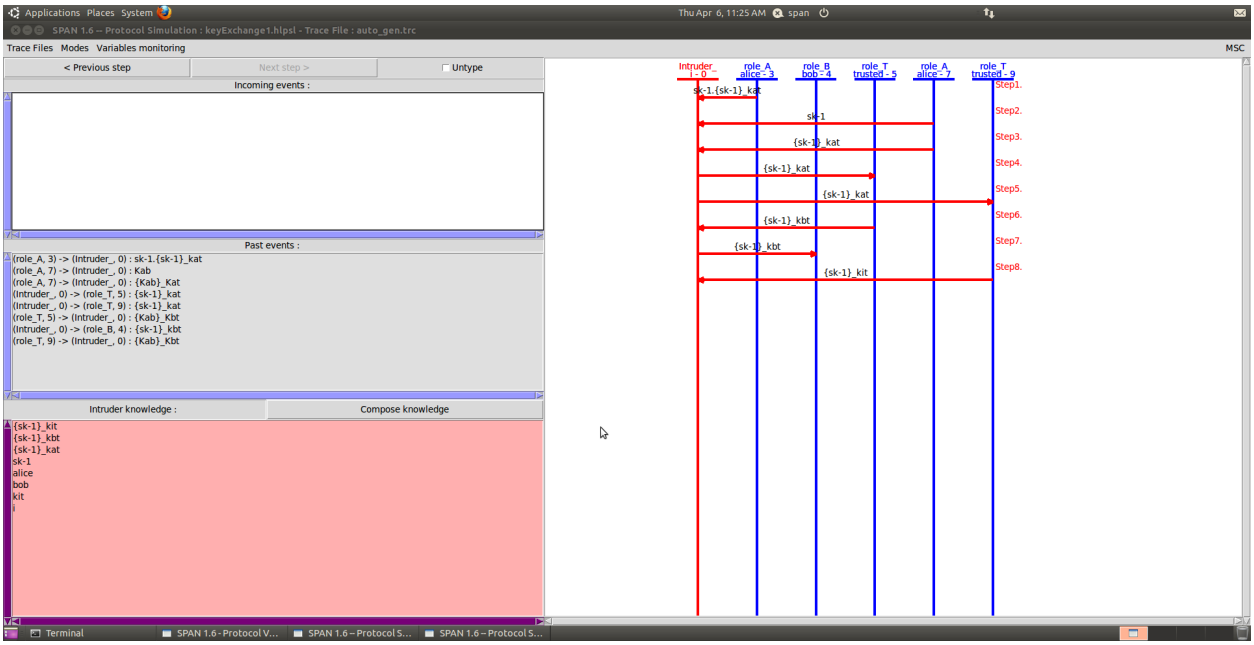
```
goal
        secrecy_of sec_1
end goal

environment()
```

- **Attack discovery, strengthening the protocol**

this protocol is unsafe we can see below.



below is the intruder simulation:



- **Tuning and optimizing the protocol**

  Since there are three bugs in the code of keyexchangeprotocol_1 the protocol is UNSAFE. and that's why we will check keyexchangeprotocol_5 which is the optimized version of keyexchangeprotocol_1.

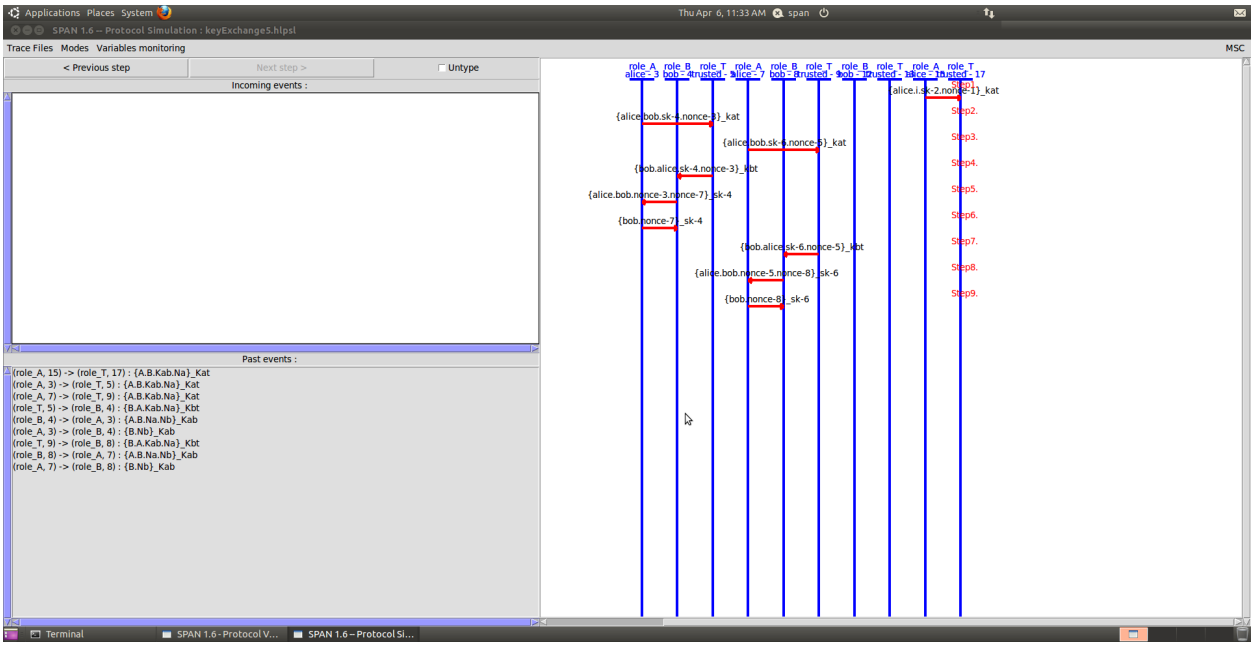
2. **keyexchangeprotocol_5**

   In this protocol, there are identities and norms everywhere. it's very big and complex protocol. In this protocol there is secrecy and mutual authentication between A and B.Now we will see all results and check all steps are valid for protocol development or not.

- **Specifying protocol and properties**

  1. **A -> T : {A,B,Kab,Na}_Kat**
  2. **T -> B : {B,A,Kab,Na}_Kbt**
  3. **B -> A : {A,B,Na,Nb}_Kab**
  4. **A -> B : {A,B,Nb}_Kab**

Protocol simulation



in above protocol simulation we can see that A - T using key K_AT. B-T using key K_BT and A - B using key K_AB.

- **Debugging specification using animation: Find the blocking transition, monitor the variables**

there is one session. but we can put more than one session in parallel. A is communicating with T using K_AT. B is communicating with T using the symmetric key K_BT. and after that A and B are talking with the key K_AB generated by trusted third party T.

So A has a symmetric key K_AT, T has two symmetric keys K_AT, K_BT, and B has of course one symmetric key that is K_BT.

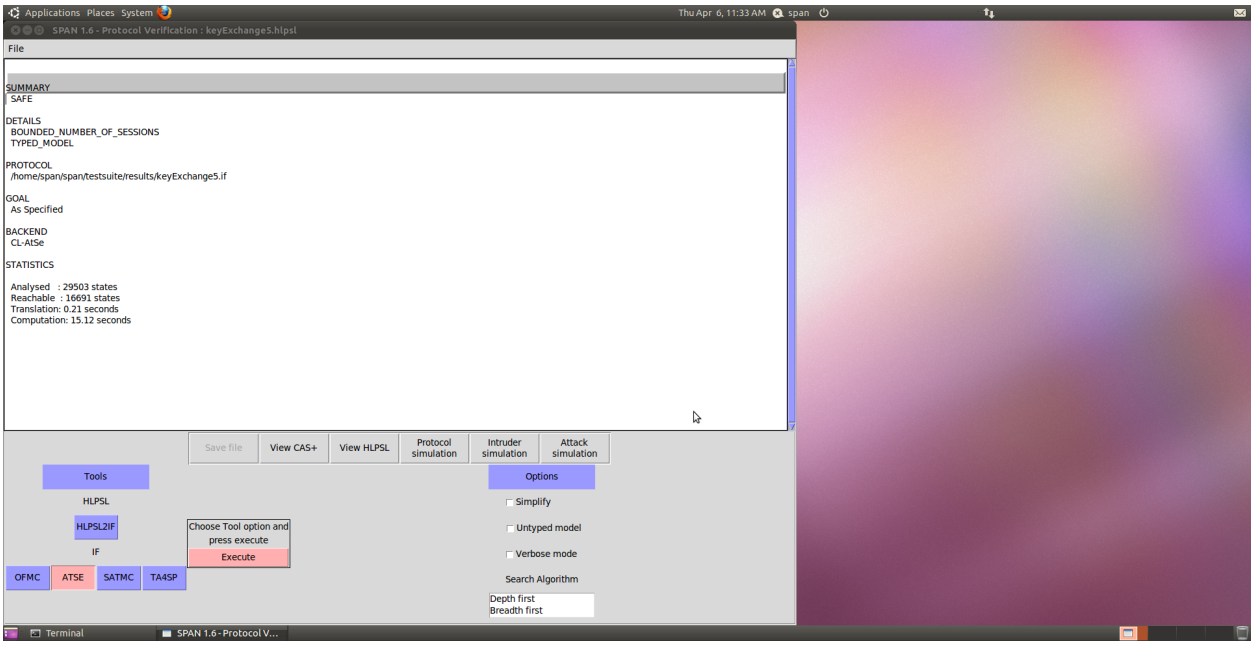
```
%%% Key exchange protocol, secured for secrecy, mutual authentication
%%%of A and B (but unoptimized)
%%% 1. A -> T : {A,B,Kab,Na}_Kat
%%% 2. T -> B : {B,A,Kab,Na}_Kbt
%%% 3. B -> A : {A,B,Na,Nb}_Kab
%%% 4. A -> B : {A,B,Nb}_Kab

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({A.B.Kab'.Na'}_Kat)
/\ secret(Kab',sec_1,{A,B,T})

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

                    %% A checks that he receives the same nonce
                    %% that he sent at step 1.
                    /\ request(A,B,auth_1,Na)

                    %% A hopes that Nb will permit to authenticate him
                    /\ witness(B,A,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({A.B.Kab'.Na'}_Kat) =|>

                    State':=1 /\ SND({B.A.Kab'.Na'}_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({B.A.Kab'.Na'}_Kbt) =|>

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

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

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

                %% B checks that he receives the same nonce
```

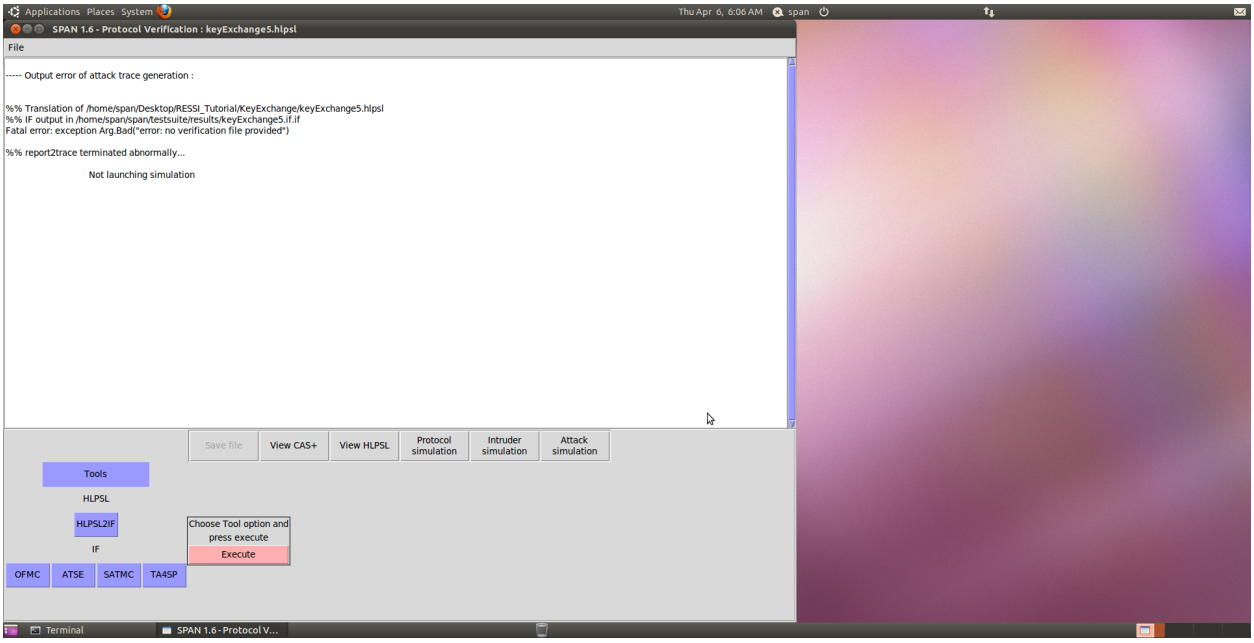
```
                %% that he sent at step 1.

                /\ request(A,B,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()
```

- **Attack discovery, strengthening the protocol**

this protocol is safe that is optimized version of protol_1 but it is not optimized.



we can say that it is safe as there is no attack simulation



- **Tuning and optimizing the protocol**

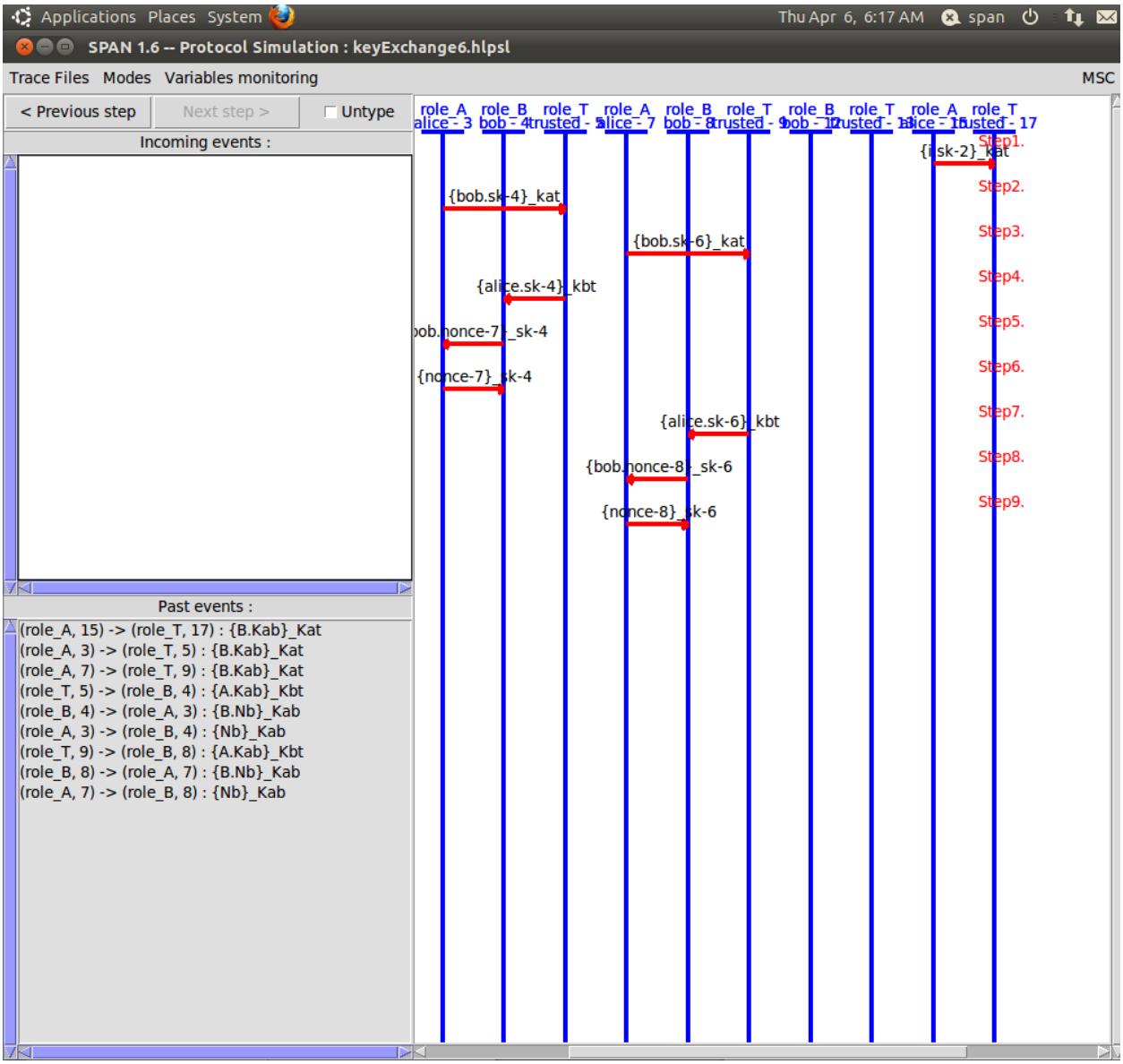
keyexchangeprotocol_5 is SAFE but not optimized. and that’s why we will check keyexchangeprotocol_6 which is the optimized version of keyexchangeprotocol_5 and 1.

3. **keyexchangeprotocol_6**

this protocol is simplified, safe and optimized over all key exchange protocol.

- **Specifying protocol and properties**
    1. **A -> T : {B,Kab}_Kat**
    2. **T -> B : {A,Kab}_Kbt**
    3. **B -> A : {B,Nb}_Kab**
    4. **A -> B : {Nb}_Kab**

Protocol simulation



in above protocol simulation we can see that A - T using key K_AT. B-T using key K_BT and A - B using key K_AB.

- **Debugging specification using animation: Find the blocking transition, monitor the variables**

there is one session. but we can put more than one session in parallel. A is communicating with T using K_AT. B is communicating with T using the symmetric key

K_BT. and after that A and B are talking with the key K_AB generated by trusted third party T.

So A has a symmetric key K_AT, T has two symmetric keys K_AT, K_BT, and B has of course one symmetric key that is K_BT.

```
%%% Key exchange protocol, secured for secrecy, mutual authentication of
%%%A and B (optimized)
%%% 1. A -> T : {B,Kab}_Kat
%%% 2. T -> B : {A,Kab}_Kbt
%%% 3. B -> A : {B,Nb}_Kab
%%% 4. A -> B : {Nb}_Kab

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
```

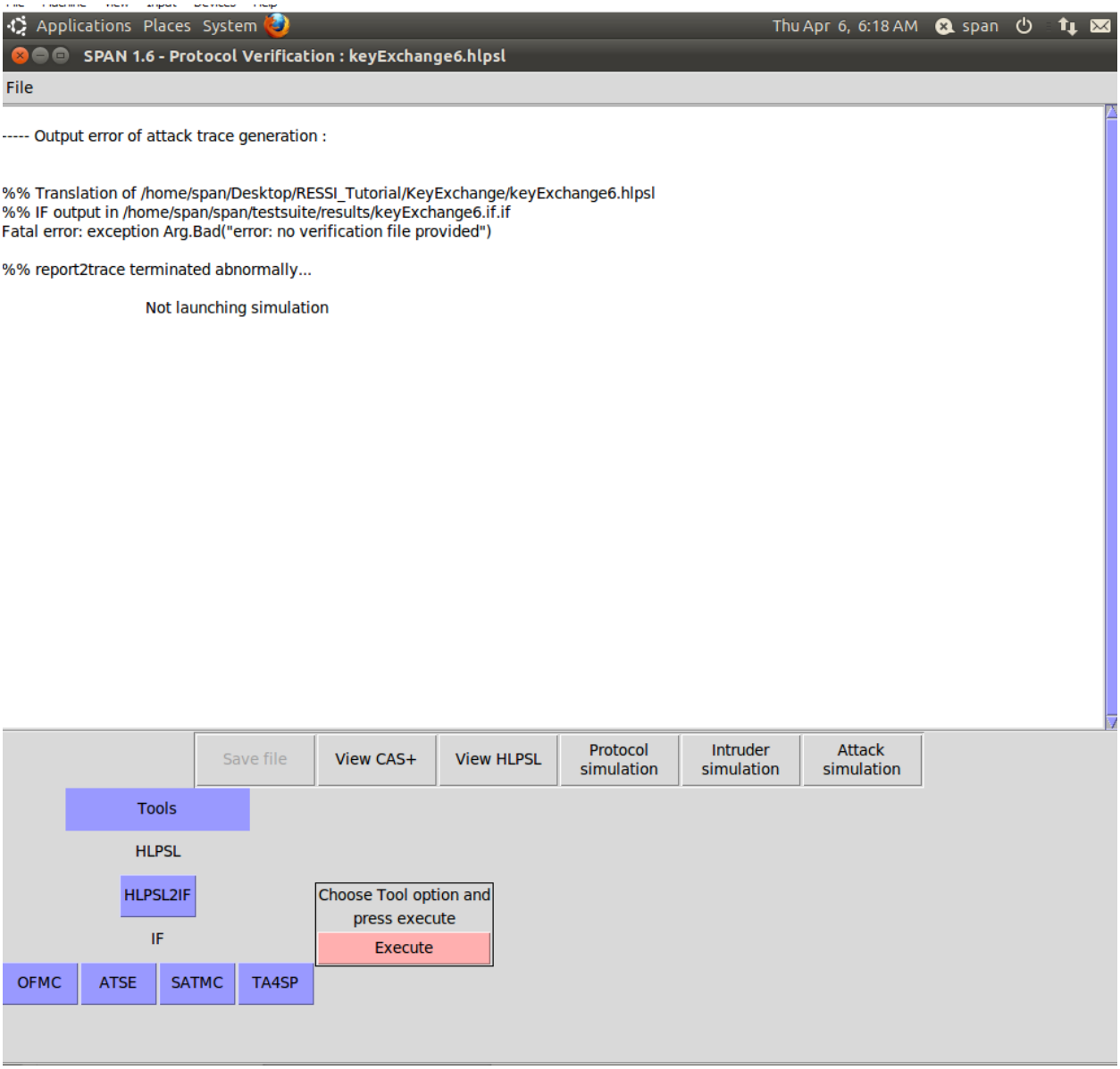
```
role role_B(B:agent,A:agent,T:agent,Kbt:symmetric_key,SND,RCV:channel(dy))
played_by B
def=
local
State:nat,Na,Nb:text,Kab:symmetric_key
init
State := 0
transition
1. State=0 /\ RCV({A.Kab'}_Kbt) =|>

                   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()
```

- **Attack discovery, strengthening the protocol**

this protocol is safe that is optimized version of protol_5 and optimized.

there is no attack simulation and hence safe.



- **Tuning and optimizing the protocol**

It is the most optimized protocol simulation, which satisfies all steps that are need for the Protocol development using SPAN+AVISPA