Verification Tools - Security Model

AVISPA

export AVISPA_PACKAGE=/Users/qingmang/JrX_Code/avispa-1.1

export PATH=$PATH:$AVISPA_PACKAGE

avispa --output=/Users/qingmang/JrX_Code/avispa

avispa /Users/qingmang/JrX_Code/avispa-1.1/testsuite/hlpsl/NSPK.hlpsl --ofmc

1. 4 Tool
    - OFMC

```
avispa /Users/qingmang/JrX_Code/avispa-1.1/testsuite/hlpsl/NSPK.hlpsl
--ofmc

PROTOCOL
  /Users/qingmang/JrX_Code/avispa/NSPK.if
GOAL
  secrecy_of_nb
BACKEND
  OFMC
COMMENTS
STATISTICS
  parseTime: 0.00s
  searchTime: 0.02s
  visitedNodes: 10 nodes
  depth: 2 plies
ATTACK TRACE
i -> (a,6): start
(a,6) -> i: {Na(1).a}_ki
i -> (b,3): {Na(1).a}_kb
(b,3) -> i: {Na(1).Nb(2)}_ka
i -> (a,6): {Na(1).Nb(2)}_ka
(a,6) -> i: {Nb(2)}_ki
i -> (i,17): Nb(2)
i -> (i,17): Nb(2)
% Reached State:
%
% secret(Nb(2),nb,set_70)
% witness(b,a,alice_bob_nb,Nb(2))
% contains(a,set_70)
% contains(b,set_70)
% secret(Na(1),na,set_74)
% witness(a,i,bob_alice_na,Na(1))
% contains(a,set_74)
```

```
% contains(i,set_74)
% state_bob(b,i,ki,kb,1,dummy_nonce,dummy_nonce,set_78,10)
% state_alice(a,i,ka,ki,4,Na(1),Nb(2),set_74,6)
% state_bob(b,a,ka,kb,3,Na(1),Nb(2),set_70,3)
% state_alice(a,b,ka,kb,0,dummy_nonce,dummy_nonce,set_62,3)
% request(a,i,alice_bob_nb,Nb(2),6)
```

- Cl-Atse

```
avispa /Users/qingmang/JrX_Code/avispa-1.1/testsuite/hlpsl/NSPK.hlpsl  --cl-atse

SUMMARY
  UNSAFE
DETAILS
  ATTACK_FOUND
  TYPED_MODEL
PROTOCOL
  /Users/qingmang/JrX_Code/avispa/NSPK.if
GOAL
  Secrecy attack on (n5(Nb))
BACKEND
  CL-AtSe
STATISTICS
  Analysed   : 9 states
  Reachable  : 8 states
  Translation: 0.00 seconds
  Computation: 0.00 seconds
ATTACK TRACE
 i -> (a,6):  start
 (a,6) -> i:  {n9(Na).a}_ki
              & Secret(n9(Na),set_74);  Add a to set_74;  Add i to set_74;
 i -> (a,3):  start
 (a,3) -> i:  {n1(Na).a}_kb
              & Secret(n1(Na),set_62); Witness(a,b,bob_alice_na,n1(Na));
              & Add a to set_62;  Add b to set_62;
 i -> (b,4):  {n9(Na).a}_kb
 (b,4) -> i:  {n9(Na).n5(Nb)}_ka
              & Secret(n5(Nb),set_70); Witness(b,a,alice_bob_nb,n5(Nb));
              & Add a to set_70;  Add b to set_70;
 i -> (a,6):  {n9(Na).n5(Nb)}_ka
 (a,6) -> i:  {n5(Nb)}_ki
```

- SATMC

```
avispa /Users/qingmang/JrX_Code/avispa-1.1/testsuite/hlpsl/NSPK.hlpsl  --satmc --solver=sim

SUMMARY
  UNSAFE
DETAILS
  ATTACK_FOUND
  BOUNDED_NUMBER_OF_SESSIONS
  BOUNDED_SEARCH_DEPTH
  BOUNDED_MESSAGE_DEPTH
PROTOCOL
  /Users/qingmang/JrX_Code/avispa/NSPK.if
GOAL
  secrecy_of_nb(nb0(b,4),set_70)
BACKEND
  SATMC
COMMENTS
STATISTICS
  attackFound                true      boolean
  upperBoundReached          false     boolean
  graphLeveledOff            no        boolean
  satSolver                  sim       solver
  maxStepsNumber             30        steps
  stepsNumber                5         steps
  atomsNumber                379       atoms
  clausesNumber              993       clauses
  encodingTime               0.09      seconds
  solvingTime                0.0       seconds
  if2sateCompilationTime     0.04      seconds
ATTACK TRACE
  i       ->    (a,6)    : start
  (a,6)   ->    i        : {na0(a,6).a}_ki
  i       ->    (b,4)    : {na0(a,6).a}_kb
  (b,4)   ->    i        : {na0(a,6).nb0(b,4)}_ka
  i       ->    (a,6)    : {na0(a,6).nb0(b,4)}_ka
  (a,6)   ->    i        : {nb0(b,4)}_ki
```

- TA4SP

```
avispa /Users/qingmang/JrX_Code/avispa-1.1/testsuite/hlpsl/NSPK.hlpsl  --ta4sp

SUMMARY
   INCONCLUSIVE
DETAILS
   OVER_APPROXIMATION
   UNBOUNDED_NUMBER_OF_SESSIONS
   TYPED_MODEL
```

```
PROTOCOL
   /Users/qingmang/JrX_Code/avispa/NSPK.if.ta4sp
GOAL
   SECRECY - Property with identitier: nb
BACKEND
   TA4SP
COMMENTS
   Use an under-approximation in order to show a potential attack
   The intruder might know some critical information
STATISTICS
   Translation: 0.01 seconds
   Computation 0.68 seconds
ATTACK TRACE
   No Trace can be provided with the current version.
```

2. Under Different results
   - OFMC (On-the-fly Model-Checker): Based on the description of the IF language requirements, OFMC can complete the tampering and limited confirmation of the protocol by detecting the changes of the system. OFMC supports the specification of the algebraic nature of cryptographic operations, as well as various protocol models.
   - CL-AtSe (Constraint-Logic-based Attack Searcher): CL-AtSe implements protocols through powerful simplified detection and redundancy elimination techniques. It is built on a modeled approach and is an extension of the algebraic nature of cryptographic operations. CL-AtSe supports input defect detection and processing message concatenation.
   - SATMC (SAT-based Model-Checker): The formula for the coding of the SATMC based on the finite-domain transition relationship described by the IF language. The description of the initial state and state set represents the security characteristics of the entire protocol. This formula will be fed back to the SAT Status Initiator and any model created will be converted into an attack event.
   - TA4SP (Tree Automata based on Automatic Approximations for the Analysis of Security Protocols): TA4SP estimates the intruder's knowledge through a tree language and rewriting mechanism. Depending on the privacy features, the TA4SP can determine if a protocol is defective or if it is safe after several conversations.
3. Correct the protocol like following. When *I* received the message from Attacker, it will use R and the identify of Attacker to judge whether Attacker is *R* or not
   - $I \rightarrow R:(N_I, I)KP_R$
     $R \rightarrow I:(N_I, N_R, R)KP_I$
     $I \rightarrow R:(N_I)KP_R$
4. I found the basic component of the protocol, but not all of them.

Proverif

./proverif examples/horn/secr/needham

1. Needham

```
./proverif  needham.horn
```

```
[jrxie:proverif2.00]$ proverif -in horn needham.horn
-bash: proverif: command not found
[jrxie:proverif2.00]$ ./proverif  needham.horn
Initial clauses:
Clause 14: c:(v_25,v_26) -> c:v_26
Clause 13: c:(v_20,v_21) -> c:v_20
Clause 12: c:v_18 & c:v_19 -> c:(v_18,v_19)
Clause 11: c:c[]
Clause 10: c:pk(sA[])
Clause 9: c:pk(sB[])
Clause 8: c:x_17 & c:encrypt(m_16,pk(x_17)) -> c:m_16
Clause 7: c:x_15 -> c:pk(x_15)
Clause 6: c:x_14 & c:y_13 -> c:encrypt(x_14,y_13)
Clause 5: c:pk(x_12) -> c:encrypt((Na[pk(x_12)],pk(sA[])),pk(x_12))
Clause 4: c:pk(x_10) & c:encrypt((Na[pk(x_10)],y_11),pk(sA[])) -> c:encrypt((y_11,k[pk(x_10)]),p
k(x_10))
Clause 3: c:encrypt((x_8,y_9),pk(sB[])) -> c:encrypt((x_8,Nb[x_8,y_9]),y_9)
Clause 2: c:encrypt((x_6,pk(sA[])),pk(sB[])) & c:encrypt((Nb[x_6,pk(sA[])],z_7),pk(sB[])) -> c:e
ncrypt(secret[],pk(z_7))
Clause 1: c:new-name[!att = v_4]
Completing...
goal reachable: c:secret[]
Abbreviations:
Na_190 = Na[pk(x_174)]
Nb_191 = Nb[Na_190,pk(sA[])]
k_192 = k[pk(x_174)]
clause 8 c:secret[]
    duplicate c:x_188
    clause 2 c:encrypt(secret[],pk(x_188))
        duplicate c:encrypt((Na_190,pk(sA[])),pk(sB[]))
        clause 6 c:encrypt((Nb_191,x_188),pk(sB[]))
            apply 2-tuple c:(Nb_191,x_188)
                apply 1-proj-2-tuple c:Nb_191
```

2. Understand the track

3. Correct it

```
pred c/1 elimVar,decompData.
nounif c:x.
fun pk/1.
fun encrypt/2.
query c:secret[].
reduc
(* Initialization *)
c:c[];
c:pk(sA[]);
c:pk(sB[]);
(* The attacker *)
c:x & c:encrypt(m,pk(x)) -> c:m;
c:x -> c:pk(x);
c:x & c:y -> c:encrypt(x,y);
(* The protocol *)
(* A *)
c:pk(x) -> c:encrypt((Na[pk(x)], pk(sA[])), pk(x));
c:pk(x) & c:encrypt((Na[pk(x)], y, pk(x)), pk(sA[]))
   -> c:encrypt((y,k[pk(x)]), pk(x));
```

```
(* B *)
c:encrypt((x,y), pk(sB[])) -> c:encrypt((x, Nb[x,y], pk(sB[])), y);
c:encrypt((x,pk(sA[])), pk(sB[])) & c:encrypt((Nb[x, pk(sA[])], z), pk(sB[]))
   -> c:encrypt(secret[], pk(z)).
```