基于 AVISPA 工具进行 NSPK 协议分析

一、AVISPA 工具介绍

AVISPA 分析工具的结构图如图 1 所示。HLPSL 是一种丰富的、模块化的、基于角色的形式语言，提供了一套包括控制流模式、数据结构、可选择入侵者模式、复杂的安全目标以及不同的密码初始值和代数性质的说明。这些特性能够使 HLPSL 很好的描述现代的、工业化规模的协议。而且，HLPSL 不仅支持基于时间片段的逻辑行为的公开语义，还支持基于重写的中间形式化语言 IF。

HLPSL2IF 自动将 HLPSL 语言翻译成 IF 语言，并将它们依次反馈给测试后端。AVISPA 使用了 4 种后端分析工具来解决安全协议的确认问题：

（1）OFMC(On-the-fly Model-Checker)：基于 IF 语言需求驱使的描述，通过探测系统的变迁，OFMC 能够完成协议的篡改和有限制的确认。OFMC 支持密码操作的代数性质的规范，以及各种协议模型。

（2）CL-AtSe（Constraint-Logic-based Attack Searcher)：CL-AtSe 通过强大的简化探测法和冗余排除技术来执行协议。它建立在模型化的方式上，并且是对密码操作的代数性质的延伸。CL-AtSe 支持输入缺陷探测和处理消息串联。

（3）SATMC（SAT-based Model-Checker)：SATMC 建立在通过 IF 语言描述的，有限域上变迁关系的编码的公式，初始状态和状态集合的说明代表了整个协议的安全特性。此公式将反馈给 SAT 状态推导机，并且建立的任何一个模型都将转化为一个攻击事件。

（4）TA4SP(Tree Automata based on Automatic Approximations for the Analysis of Security Protocols)：TA4SP 通过树形语言和重写机制估计入侵者的知识。根据不同的保密特性，TA4SP 能够判断一个协议是否有缺陷，或者是几个会合的对话后是否安全。

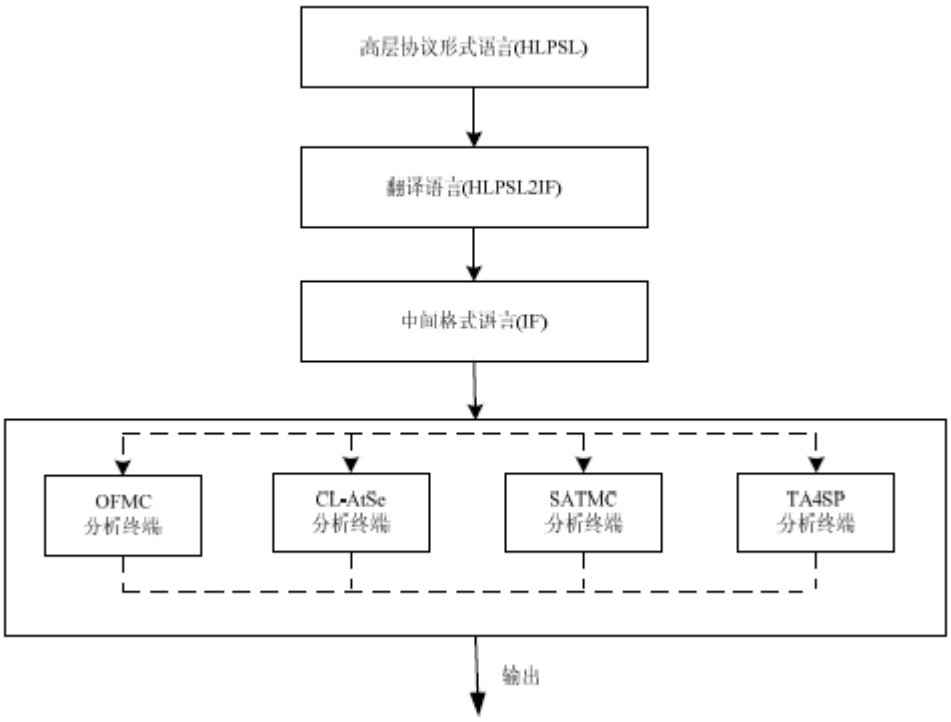

图 1 AVISPA 工具结构

二、AVISPA 工具安装

AVISPA 工具在 AVISPA 官方网站上可以下载，运行在 Linux 操作系统环境下。首先，下载并安装 AVISPA v1.1 版本（本文使用 AVISPA1.0 版本为例），解压安装包到对应目录； 其次， 需要设置工具集的环境变量， 将 AVISPA_PACKAGE 关联到安装包的绝对路径；最后将 avispa 脚本语言设置在命令行解释器的执行目录中。例如：

用户想安装 AVISPA 工具集在/opt 路径下，命令如下[5]：

```
cd /opt
tar -xzf /home/xyz/avispa-package-X.Y_Linux-i686.tgz
export AVISPA_PACKAGE=/opt/avispa-X.Y
export PATH=$PATH:$AVISPA_PACKAGE
```

三、XEmacs 模式的使用

（1）XEmacs 工具的安装。

AVISPA 提供和 XEmacs 工具的用户友好接口（XEmacs 工具是 Linux 操作系统下的一种编辑器），它们之间支持用户和 AVISPA 工具集之间的简单交互。首先，Linux 操作系统需要安装 XEmacs 编辑器；其次，需要对 AVISPA 进行设置，使其支持 XEmacs 模式，命令如下：

```
cd /opt/others
tar -xzvf avispa-mode.tgz
cd temporary-avispa
make install
```

（2）XEmacs 工具按钮。

AVISPA XEmacs 模式提供了一套完整而直观的编译环境对安全协议进行说明和分析。最常用的分析出发点是通过 hlpsl 文件开始。在 AVISPA 模式下，当 XEmacs 工具自动侦测出后缀名为“.hlpsl”的文件时，会在 XEmacs 工具栏上出现对应的 AVISPA 按钮。

AVISPA 按钮的大致功能如下：

AVISPA：提供选项、模式的定制和改变后端分析工具等功能；

<<和>>：提供在同一个协议不同的分析文件（例如“.if”，“.atk”等）之间进行导航；

Process file：导入编译器对中间文件进行编译和分析，得出结论；

Update：当一个工具被 XEmacs 异步导入时，一旦此工具被中断，此按钮将刷新当前的缓存区。

NSPK 协议分析

（1） NSPK 协议简介

非对称密码体制 NSPK 协议，协议双方身份认证部分为：

1)A—B：{N:，A}K

A 首先生成临时值 M，加上自己的身份，用 B 的公开密钥 KB 加密后发送给 B。

2)B—A：{N:，N6}K.

B 生成临时值 N6，加上 A 的临时值 Nn，用 A 的公开密钥 KA 加密后发送给 A。

3)A—B：{M}K。

A 向 B 发送经过 KB 加密的 Nj．整个协议采用公开密钥系统，K4，K8 分别是 A 和 B 的公开密钥．N 口，N6 是 A 和 B 发布的具有新鲜性的临时值(nonce)。

（2）NSPK 分析实验

使用 AVISPA 协议分析工具对安全协议进行分析的一般性过程如下：首先，将安全协

议编码为某种形式化描述语言；然后，根据协议目标和安全属性，给出不同的消息成分的类型；最后，根据分析工作的结果判断协议是否安全，是否达到了预期目标。

（1）分析安全协议，并根据 HLPSL 语法，将协议进行建模，编辑成后缀名为“.hlpsl”的文件，具体语法见《安全协议形式化分析的研究和实现》；

如下为 NSPK 协议的 hlpsl 实现：

```
role alice (A, B: agent,
            Ka, Kb: public_key,
            SND, RCV: channel (dy))
played_by A def=
  local State : nat,
        Na, Nb: text
  init State := 0
  transition
    0.  State  = 0 /\ RCV(start) =|>
    State'= 2 /\ Na' := new() /\ SND({Na'.A}_Kb)
            /\ secret(Na',na,{A,B})
            /\ witness(A,B,bob_alice_na,Na')
    2.  State  = 2 /\ RCV({Na.Nb'}_Ka) =|>
    State'= 4 /\ SND({Nb'}_Kb)
            /\ request(A,B,alice_bob_nb,Nb')
end role

role bob(A, B: agent,
         Ka, Kb: public_key,
         SND, RCV: channel (dy))
played_by B def=
  local State : nat,
    Na, Nb: text
  init State := 1
  transition
    1.  State  = 1 /\ RCV({Na'.A}_Kb) =|>
    State'= 3 /\ Nb' := new() /\ SND({Na'.Nb'}_Ka)
            /\ secret(Nb',nb,{A,B})
            /\ witness(B,A,alice_bob_nb,Nb')
    3.  State  = 3 /\ RCV({Nb}_Kb) =|>
    State'= 5 /\ request(B,A,bob_alice_na,Na)
end role

role session(A, B: agent, Ka, Kb: public_key) def=
  local SA, RA, SB, RB: channel (dy)
  composition
    alice(A,B,Ka,Kb,SA,RA)
     /\ bob   (A,B,Ka,Kb,SB,RB)
end role
```

```
role environment() def=
      const a, b              : agent,
        ka, kb, ki      : public_key,
        na, nb,
        alice_bob_nb,
        bob_alice_na : protocol_id
      intruder_knowledge = {a, b, ka, kb, ki, inv(ki)}
      composition
      session(a,b,ka,kb)
       /\ session(a,i,ka,ki)
       /\ session(i,b,ki,kb)
end role

goal
   secrecy_of na, nb
   authentication_on alice_bob_nb
   authentication_on bob_alice_na
end goal
```

（2）在终端中执行分析命令：./avispa /tmp/NSPK.hlpsl --output=/opt –ofmc，软件将输入的 NSPK.hlpsl 文件转化为 NSPK.if 文件，并将其作为分析器的输入语言，其语法格式如下：

```
%% IF specification of contrib/avispa-library/NSPK.hlpsl

section signature:

state_bob: agent * agent * public_key * public_key * nat * text * text * set(agent) * nat ->
fact
state_alice: agent * agent * public_key * public_key * nat * text * text * set(agent) * nat ->
fact



section types:

nb, alice_bob_nb, bob_alice_na, na: protocol_id
Na, Nb, Dummy_Nb, Dummy_Na, dummy_nonce: text
set_78, set_77, set_76, set_74, set_70, set_62: set
5, 3, 1, State, 10, 6, 4, SID, 2, Dummy_State, 0, SID2, SID1: nat
Set_38, Dummy_Set_38, Set_18, Dummy_Set_18, ASGoal: set(agent)
start, MGoal: message
Ka, Kb, ka, kb, ki: public_key
A, B, A2Goal, A1Goal, i, a, b: agent
```

```
section inits:

initial_state init1 :=
  iknows(start).
  iknows(a).
  iknows(b).
  iknows(ka).
  iknows(kb).
  iknows(ki).
  iknows(inv(ki)).
  iknows(i).
  state_alice(a,b,ka,kb,0,dummy_nonce,dummy_nonce,set_62,3).
  state_bob(b,a,ka,kb,1,dummy_nonce,dummy_nonce,set_70,4).
  state_alice(a,i,ka,ki,0,dummy_nonce,dummy_nonce,set_74,6).
  state_bob(b,i,ki,kb,1,dummy_nonce,dummy_nonce,set_78,10)


section rules:

step step_0 (A,B,Ka,Kb,Dummy_Na,Nb,Dummy_Set_18,SID,Na) :=
  state_alice(A,B,Ka,Kb,0,Dummy_Na,Nb,Dummy_Set_18,SID).
  iknows(start)
=[exists Na]=>
  state_alice(A,B,Ka,Kb,2,Na,Nb,Dummy_Set_18,SID).
  iknows(crypt(Kb,pair(Na,A))).
  secret(Na,na,Dummy_Set_18).
  witness(A,B,bob_alice_na,Na).
  contains(A,Dummy_Set_18).
  contains(B,Dummy_Set_18)

step step_1 (A,B,Ka,Kb,Na,Dummy_Nb,Set_18,SID,Nb) :=
  state_alice(A,B,Ka,Kb,2,Na,Dummy_Nb,Set_18,SID).
  iknows(crypt(Ka,pair(Na,Nb)))
  =>
  state_alice(A,B,Ka,Kb,4,Na,Nb,Set_18,SID).
  iknows(crypt(Kb,Nb)).
  request(A,B,alice_bob_nb,Nb,SID)

step step_2 (B,A,Ka,Kb,Dummy_Na,Dummy_Nb,Dummy_Set_38,SID,Na,Nb) :=
  state_bob(B,A,Ka,Kb,1,Dummy_Na,Dummy_Nb,Dummy_Set_38,SID).
  iknows(crypt(Kb,pair(Na,A)))
=[exists Nb]=>
  state_bob(B,A,Ka,Kb,3,Na,Nb,Dummy_Set_38,SID).
  iknows(crypt(Ka,pair(Na,Nb))).
```

```
  secret(Nb,nb,Dummy_Set_38).
  witness(B,A,alice_bob_nb,Nb).
  contains(A,Dummy_Set_38).
  contains(B,Dummy_Set_38)

step step_3 (B,A,Ka,Kb,Na,Nb,Set_38,SID) :=
  state_bob(B,A,Ka,Kb,3,Na,Nb,Set_38,SID).
  iknows(crypt(Kb,Nb))
  =>
  state_bob(B,A,Ka,Kb,5,Na,Nb,Set_38,SID).
  request(B,A,bob_alice_na,Na,SID)


section properties:

property secrecy_of_na (MGoal,ASGoal) :=
  [] ((secret(MGoal,na,ASGoal) /\ iknows(MGoal))
        => contains(i,ASGoal))

property secrecy_of_nb (MGoal,ASGoal) :=
  [] ((secret(MGoal,nb,ASGoal) /\ iknows(MGoal))
        => contains(i,ASGoal))

property authentication_on_alice_bob_nb (A1Goal,A2Goal,MGoal,SID,SID1,SID2) :=
  [] (((request(A1Goal,A2Goal,alice_bob_nb,MGoal,SID)
           /\ ~ equal(A2Goal,i))
         => witness(A2Goal,A1Goal,alice_bob_nb,MGoal))
       /\ ((request(A1Goal,A2Goal,alice_bob_nb,MGoal,SID1)
                /\ request(A1Goal,A2Goal,alice_bob_nb,MGoal,SID2)
                /\ ~ equal(A2Goal,i))
              => equal(SID1,SID2)))

property authentication_on_bob_alice_na (A1Goal,A2Goal,MGoal,SID,SID1,SID2) :=
  [] (((request(A1Goal,A2Goal,bob_alice_na,MGoal,SID)
           /\ ~ equal(A2Goal,i))
         => witness(A2Goal,A1Goal,bob_alice_na,MGoal))
       /\ ((request(A1Goal,A2Goal,bob_alice_na,MGoal,SID1)
                /\ request(A1Goal,A2Goal,bob_alice_na,MGoal,SID2)
                /\ ~ equal(A2Goal,i))
              => equal(SID1,SID2)))


section attack_states:
```

```
attack_state secrecy_of_na (MGoal,ASGoal) :=
  iknows(MGoal).
  secret(MGoal,na,ASGoal) &
  not(contains(i,ASGoal))

attack_state secrecy_of_nb (MGoal,ASGoal) :=
  iknows(MGoal).
  secret(MGoal,nb,ASGoal) &
  not(contains(i,ASGoal))

attack_state authentication_on_alice_bob_nb (A1Goal,A2Goal,MGoal,SID) :=
  request(A1Goal,A2Goal,alice_bob_nb,MGoal,SID) &
  not(witness(A2Goal,A1Goal,alice_bob_nb,MGoal)) &
  not(equal(A2Goal,i))
attack_state replay_protection_on_alice_bob_nb (A2Goal,A1Goal,MGoal,SID1,SID2) :=
  request(A1Goal,A2Goal,alice_bob_nb,MGoal,SID1).
  request(A1Goal,A2Goal,alice_bob_nb,MGoal,SID2) &
  not(equal(SID1,SID2)) &
  not(equal(A2Goal,i))

attack_state authentication_on_bob_alice_na (A1Goal,A2Goal,MGoal,SID) :=
  request(A1Goal,A2Goal,bob_alice_na,MGoal,SID) &
  not(witness(A2Goal,A1Goal,bob_alice_na,MGoal)) &
  not(equal(A2Goal,i))
attack_state replay_protection_on_bob_alice_na (A2Goal,A1Goal,MGoal,SID1,SID2) :=
  request(A1Goal,A2Goal,bob_alice_na,MGoal,SID1).
  request(A1Goal,A2Goal,bob_alice_na,MGoal,SID2) &
  not(equal(SID1,SID2)) &
  not(equal(A2Goal,i))
```

（3）分四种方式对 NSPK 协议进行分析，操作命令如下：

```
./avispa contrib/avispa-library/NSPK.hlpsl --output=/opt --ofmc
./avispa contrib/avispa-library/NSPK.hlpsl --output=/opt --cl-atse
./avispa contrib/avispa-library/NSPK.hlpsl --output=/opt --satmc --solver=sim
./avispa contrib/avispa-library/NSPK.hlpsl --output=/opt --ta4sp
```

软件经过分析，会给出分析结果：

1） OFMC 模式

分析结果如下：

```
% OFMC
% Version of 2006/02/13
SUMMARY
   UNSAFE
DETAILS
   ATTACK_FOUND
```

```
PROTOCOL
  /opt/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)
```

看到如上的针对 NSPK 协议进行的攻击，发现攻击过程的参与者有三个：主体 A、主体 B 和攻击者 C，其中 A 作为 NS 公钥协议的初始者，C 作为响应者并假冒 A 和 B 进行通信和欺骗，从而实现对 NS 公钥协议的攻击。整个协议采用公开密钥系统，ABCEEE、、分别是 A、B 和 C 的公开密钥，ABNN、是 A 和 B 发布的具有新鲜性的随机数（也称临时值，nonce）。攻击过程为：首先主体 A 向 C 发送包含 AN 和自己身份的消息 1，并用 C 的公钥

CE 加密消息 1；C 收到消息并马上对消息进行解密，而后假冒 A 向 B 发送 AN 和 A 身份（让 B 误认为是和 A 进行通信）的消息 1'，并用 B 的公钥 BE 进行加密；B 接到消息，并向假冒 A 的 C 发送用 A 的公钥加密的消息 ABNN、；而后 C 接到消息 2'并立即向 A 发送；对协议攻击最后一步，A 向 C 发送 A、B 之间用于通信的共享秘密 BN。这样 C 就能得到 A 和 B 之间的共享秘密 BN，从而实现以后对 A 和 B 通信内容的监听。了解完上述攻击的详细过程，我们不难发现根据上述攻击可以使 B 不能确认最后一条消息是否来自 A。这是由于 A 从未详细地声明她欲与 B 对话，因此 B 不能得到任何保证 A 知道 B 是她的对等实体。

2）cl-atse 模式

分析结果如下：

```
SUMMARY
  UNSAFE

DETAILS
  ATTACK_FOUND
  TYPED_MODEL

PROTOCOL
  /opt/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
```

3）satmc 模式

分析结果如下：

```
SUMMARY
   UNSAFE

DETAILS
   ATTACK_FOUND
   BOUNDED_NUMBER_OF_SESSIONS
   BOUNDED_SEARCH_DEPTH
   BOUNDED_MESSAGE_DEPTH

PROTOCOL
   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

4）ta4sp 模式

分析结果如下：

SUMMARY

INCONCLUSIVE

DETAILS

OVER_APPROXIMATION

UNBOUNDED_NUMBER_OF_SESSIONS

TYPED_MODEL

PROTOCOL

/opt/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.

四、总结

本文对 NSPK 协议进行验证分析，在 HLPSL 建模及 if 语法转换的基础上，使用 AVISPA 工具对其进行了安全分析，检测出了协议存在的漏洞，得到其攻击的序列。