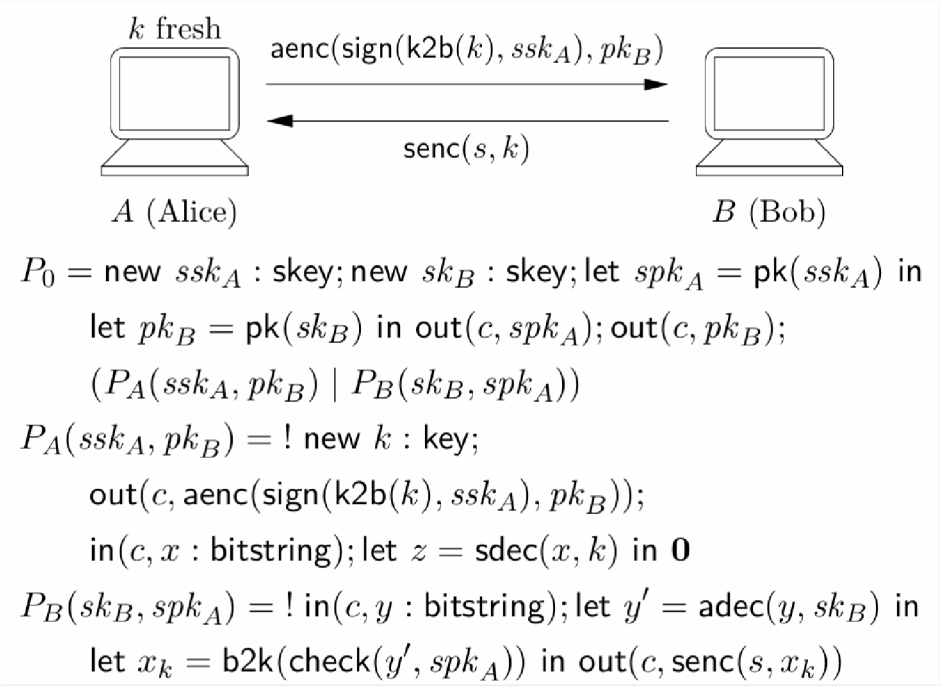
形式化方法 实验小作业3 Proverif

1. 运行结果

1.1 1st version

对进程的描述如下



第一版代码如下, 前半为规则的建立，描述了分别如何处理message，非对称解密，带有数字签名的非对称解密，对称解密。第二部分是specification，提出了`attacker(s)`的Query, 并描述了PA的流程、PB的流程和主流程（参照PPT的表达式）

```
free c: channel.

type key.(*symmetric key*)
type pkey.(*public key of B*)
type skey. (*secret key of B*)
type spkey.(*public key of A with signature*)
type sskey.(*seecret key of A with signature*)

(*rules of  messages*)
fun k2b(key):bitstring [data,typeConverter].
reduc forall k:key;b2k(k2b(k)) = k.

(*rules of unsymmetric keys*)
fun pk(skey): pkey.
fun aenc(bitstring,pkey): bitstring.
reduc forall x: bitstring,y: skey; adec(aenc(x,pk(y)),y) = x.
```

```
17
18  (*rules of unsymmetric keys with signature*)
19  fun spk(sskey): spkey.
20  fun sign(bitstring,sskey): bitstring.
21  reduc forall m: bitstring,k: sskey; checksign(sign(m,k),spk(k)) =
    m.
22
23  (*rules of symmetric keys*)
24  fun senc(bitstring,key): bitstring.
25  reduc forall x: bitstring,y: key; sdec(senc(x,y),y) = x.
26
27
28  (* Specification *)
29  free s: bitstring [private].
30  query attacker(s).
31
32  (* PA will be revised in 2nd version *)
33  let PA(sskA:sskey,pkB:pkey) =
34      new k:key;
35      out(c,aenc(sign(k2b(k),sskA),pkB));
36      in(c, x:bitstring);
37      let z = sdec(x,k) in 0.
38
39  let PB(skB:skey,spkA:spkey) =
40      in(c, y:bitstring);
41      let y1 = adec(y,skB) in
42      let xk = b2k(checksign(y1,spkA)) in
43      out(c,senc(s,xk)).
44
45  process
46      new sskA: sskey;
47      new skB: skey;
48      let spkA = spk(sskA) in
49      let pkB = pk(skB) in
50      out(c,spkA);
51      out(c,pkB);
52      (!PA(sskA,pkB) |!PB(skB,spkA))
```

将第一版代码运行后得到如下输出，可以看到转化的全过程

```
1  Process 0 (that is, the initial process):
2  {1}new sskA: sskey;
3  {2}new skB: skey;
4  {3}let spkA: spkey = spk(sskA) in
5  {4}let pkB: pkey = pk(skB) in
6  {5}out(c, spkA);
7  {6}out(c, pkB);
8  (
9      {7}!
```

```
    {8}let sskA_1: sskey = sskA in
    {9}new k: key;
    {10}out(c, aenc(sign(k,sskA_1),pkB));
    {11}in(c, x: bitstring);
    {12}let z: bitstring = sdec(x,k) in
    0
) | (
    {13}!
    {14}let skB_1: skey = skB in
    {15}in(c, y: bitstring);
    {16}let y1: bitstring = adec(y,skB_1) in
    {17}let xk: key = b2k(checksign(y1,spkA)) in
    {18}out(c, senc(s,xk))
)

--  Process 1 (that is, process 0, with let moved downwards):
{1}new sskA: sskey;
{2}new skB: skey;
{3}let spkA: spkey = spk(sskA) in
{5}out(c, spkA);
{4}let pkB: pkey = pk(skB) in
{6}out(c, pkB);
(
    {7}!
    {9}new k: key;
    {8}let sskA_1: sskey = sskA in
    {10}out(c, aenc(sign(k,sskA_1),pkB));
    {11}in(c, x: bitstring);
    {12}let z: bitstring = sdec(x,k) in
    0
) | (
    {13}!
    {15}in(c, y: bitstring);
    {14}let skB_1: skey = skB in
    {16}let y1: bitstring = adec(y,skB_1) in
    {17}let xk: key = b2k(checksign(y1,spkA)) in
    {18}out(c, senc(s,xk))
)

-- Query not attacker(s[]) in process 1.
Translating the process into Horn clauses...
Completing...
Starting query not attacker(s[])
RESULT not attacker(s[]) is true.

------------------------------------------------------------
Verification summary:

Query not attacker(s[]) is true.

------------------------------------------------------------

```

得到了正确的结果，说明攻击者虽然始终在模型里但是无法推出 `s` 的内容(即保密的数据的内容)

1.2 2nd version

把PA的进程进行以下修改

We can strengthen model by *replacing* the process P_A with the following process:

$$P_A(ssk_A, pk_B) = !\,\mathsf{in}(c, x_{pk_B} : \mathsf{pkey}); \mathsf{new}\ k : \mathsf{key};$$
$$\mathsf{out}(c, \mathsf{aenc}(\mathsf{sign}(\mathsf{k2b}(k), ssk_A), x_{pk_B}));$$
$$\mathsf{in}(c, x : \mathsf{bitstring}); \mathsf{let}\ z = \mathsf{sdec}(x, k)\ \mathsf{in}\ \mathbf{0}$$

PA段的代码修改为

```
(*  PA in 2nd version *)
let PA(sskA: sskey,pkB:pkey) =
    in(c, xpkB:pkey);
    new k:key;
    out(c,aenc(sign(k2b(k),sskA),xpkB));
    in(c, x:bitstring);
    let z = sdec(x,k) in 0.
```

同理使用 `proverif code2.pv` 指令运行得到

```
--------------------------------------------------------------
Verification summary:

Query not attacker(s[]) is false.

--------------------------------------------------------------
```

得到了正确的结果，此时为false，代表attacker能推出 `s` 的内容，机密性被破坏。因为此时A可以接受任何来自channel上的公钥，攻击者可以假装B

2. 继续完善

增加一个比对公钥的检验过程

```
(*  PA in 3rd version *)
let PA(sskA: sskey,pkB:pkey) =
    in(c, xpkB:pkey);
    if xpkB = pkB then
        new k:key;
        out(c,aenc(sign(k2b(k),sskA),xpkB));
        in(c, x:bitstring);
        let z = sdec(x,k) in 0
    else 0.
```

此时运行指令 `proverif code3.pv` 得到运行结果为真

```
--------------------------------

Query not attacker(s[]) is true.

--------------------------------
```

在发出信息之前通过比对检验先保证了公钥来自B，防止了攻击者的伪装，进而防止攻击者解出 `s`