Security Verification of Key Exchange in Ciphertext-Policy Attribute Based Encryption

Bhardwaj Aditya

Cryptography Protocols - Practice

Dec 05, 2023

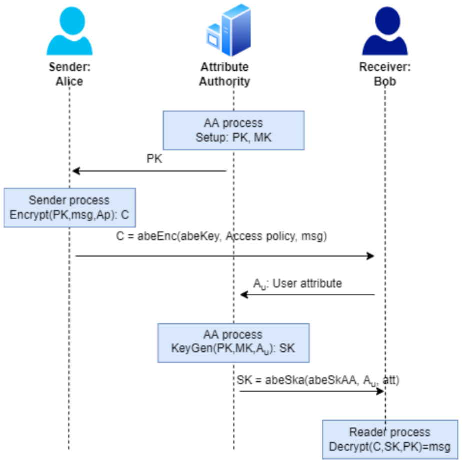
Contents

Attribute Based Encryption - ABE

Fine-grained access control of encrypted data using authorization policies. KP-ABE and CP-ABE

- What is Ciphertext-policy ABE
- An example

CP-ABE



Sender: Alice
Attribute Authority
Receiver: Bob
AA process
Setup: PK, MK
PK
Sender process
Encrypt(PK,msg,Ap): C
C = abeEnc(abeKey, Access policy, msg)
A_u: User attribute
AA process
KeyGen(PK,MK,A_u): SK
SK = abeSka(abeSkAA, A_u, att)
Reader process
Decrypt(C,SK,PK)=msg

Sending keys securely

How are the generated **secret keys** sent to the corresponding user securely ?

- KDC
- PKI (what I have chosen)

Security properties to verify

- Confidentiality
- Authentication

Making this work

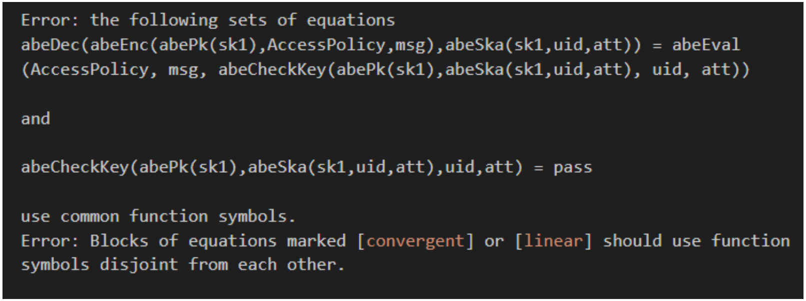
- First - Running the CP-ABE scheme
- Second - Adding PKI

What PKI scheme to use - Needham-Schroeder protocol

- Code available in Proverif
- Proven security properties

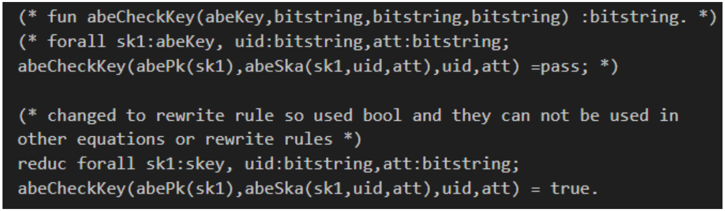
Running the CP-ABE scheme

- Added some declarations, authentication and secrecy queries
- Made changes to code

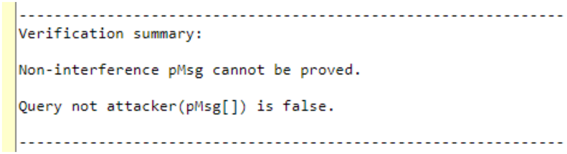


```
Error: the following sets of equations
abeDec(abeEnc(abePk(sk1),AccessPolicy,msg),abeSka(sk1,uid,att)) = abeEval
(AccessPolicy, msg, abeCheckKey(abePk(sk1),abeSka(sk1,uid,att), uid, att))

and

abeCheckKey(abePk(sk1),abeSka(sk1,uid,att),uid,att) = pass

use common function symbols.
Error: Blocks of equations marked [convergent] or [linear] should use function
symbols disjoint from each other.
```

Running the CP-ABE scheme - 2



```
(* fun abeCheckKey(abeKey,bitstring,bitstring,bitstring) :bitstring. *)
(* forall sk1:abeKey, uid:bitstring,att:bitstring;
abeCheckKey(abePk(sk1),abeSka(sk1,uid,att),uid,att) =pass; *)

(* changed to rewrite rule so used bool and they can not be used in
other equations or rewrite rules *)
reduc forall sk1:skey, uid:bitstring,att:bitstring;
abeCheckKey(abePk(sk1),abeSka(sk1,uid,att),uid,att) = true.
```

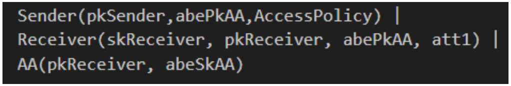


```
--------------------------------------------------------------
Verification summary:

Non-interference pMsg cannot be proved.

Query not attacker(pMsg[]) is false.

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

Adding the PKI scheme

Original process

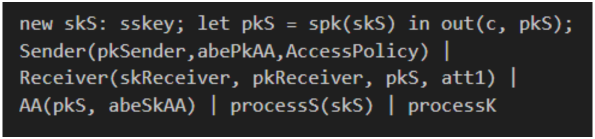
- Public keys are assumed



```
Sender(pkSender,abePkAA,AccessPolicy) |
Receiver(skReceiver, pkReceiver, abePkAA, att1) |
AA(pkReceiver, abeSkAA)
```

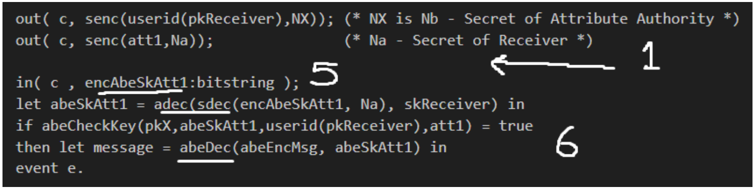
After changes

- Only CA's public key - pkS is made available



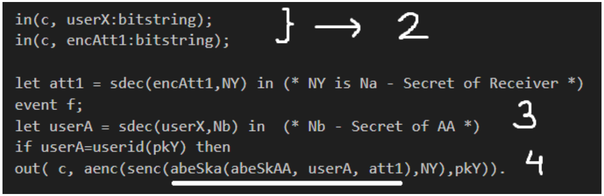
```
new skS: sskey; let pkS = spk(skS) in out(c, pkS);
Sender(pkSender,abePkAA,AccessPolicy) |
Receiver(skReceiver, pkReceiver, pkS, att1) |
AA(pkS, abeSkAA) | processS(skS) | processK
```

Receiver - Decryption



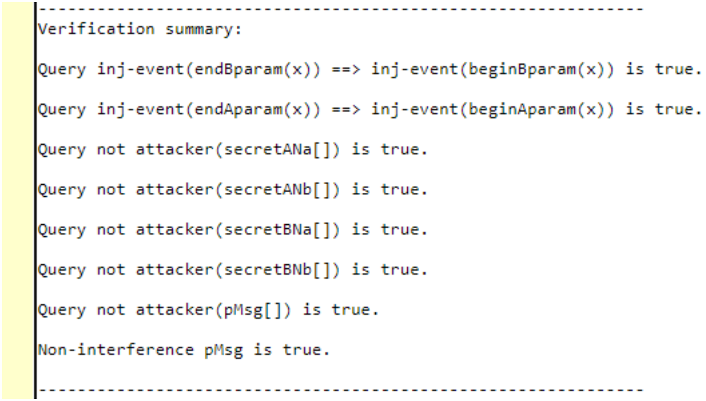
```
out( c, senc(userid(pkReceiver),NX)); (* NX is Nb - Secret of Attribute Authority *)
out( c, senc(att1,Na));               (* Na - Secret of Receiver *)

in( c , encAbeSkAtt1:bitstring );
let abeSkAtt1 = adec(sdec(encAbeSkAtt1, Na), skReceiver) in
if abeCheckKey(pkX,abeSkAtt1,userid(pkReceiver),att1) = true
then let message = abeDec(abeEncMsg, abeSkAtt1) in
event e.
```

Attribute Authority - Encryption



```
in(c, userX:bitstring);
in(c, encAtt1:bitstring);

let att1 = sdec(encAtt1,NY) in (* NY is Na - Secret of Receiver *)
event f;
let userA = sdec(userX,Nb) in  (* Nb - Secret of AA *)
if userA=userid(pkY) then
out( c, aenc(senc(abeSka(abeSkAA, userA, att1),NY),pkY)).
```

Final Result



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

Query inj-event(endBparam(x)) ==> inj-event(beginBparam(x)) is true.

Query inj-event(endAparam(x)) ==> inj-event(beginAparam(x)) is true.

Query not attacker(secretANa[]) is true.

Query not attacker(secretANb[]) is true.

Query not attacker(secretBNa[]) is true.

Query not attacker(secretBNb[]) is true.

Query not attacker(pMsg[]) is true.

Non-interference pMsg is true.

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

http://proverif20.paris.inria.fr/index.php