RAPPORT
Sécurité des Protocoles

Part 1 (Design of a key exchange protocol)

Guillaume Zablot
Antoine Courtil

28/12/2018
M2 - SIRAV

INTRODUCTION

L'objectif de ce projet est de concevoir un protocole permettant à deux agents A et B d'échanger une clé qui a été générée durant la session. À l'issu de ce protocole, les deux agents doivent partager la même clé, confidentielle et mutuellement authentifiée. Au départ, les agents A et B connaîtront leurs clés de chiffrement publiques. Ils peuvent également partager une clé symétrique avec un serveur de confiance. Cependant, vous ne pouvez pas supposer que A et B partagent initialement une clé symétrique.

DESCRIPTION

Voici la description de notre protocole proposé :

A **TS** : { <nonceA, $\{pkA'\}_{priv(skA)}$> }$_{sym(kaTS)}$

TS **B** : { <nonceA, $\{pkA'\}_{priv(skA)}$> }$_{sym(kbTS)}$

B **TS** : { <nonceA, <nonceB, $\{pkB'\}_{priv(skB)}$>> }$_{sym(kbTS)}$

TS **A** : { <nonceA, <nonceB, $\{pkB'\}_{priv(skB)}$>> }$_{sym(kaTS)}$

A **B** : { <nonceA', <nonceB, cleSession>> }$_{pub(pkB)}$

B **A** : { nonceA' }$_{sym(cleSession)}$

DONNÉES GÉNÉRÉES

Durant le protocole, plusieurs données sont générées :

- nonceA généré par A
- nonceB généré par B
- cleSession généré par A

HYPOTHÈSES

Dans ce protocole, nous émettons les hypothèses de connaissances suivantes :

- Agent A :
    - clé publique *pkA*
    - clé secrète *skA*
    - clé publique *pkB*
    - clé symétrique avec TS *kaTS*

- Agent B :
    - clé publique *pkB*
    - clé secrète *skB*
    - clé publique *pkA*
    - clé symétrique avec TS *kbTS*

- Serveur de confiance TS :
    - clé publique *pkA*
    - clé publique *pkB*
    - clé symétrique avec TS *kaTS*
    - clé symétrique avec TS *kbTS*

PROPRIÉTÉS DE SÉCURITÉ

Authentification :

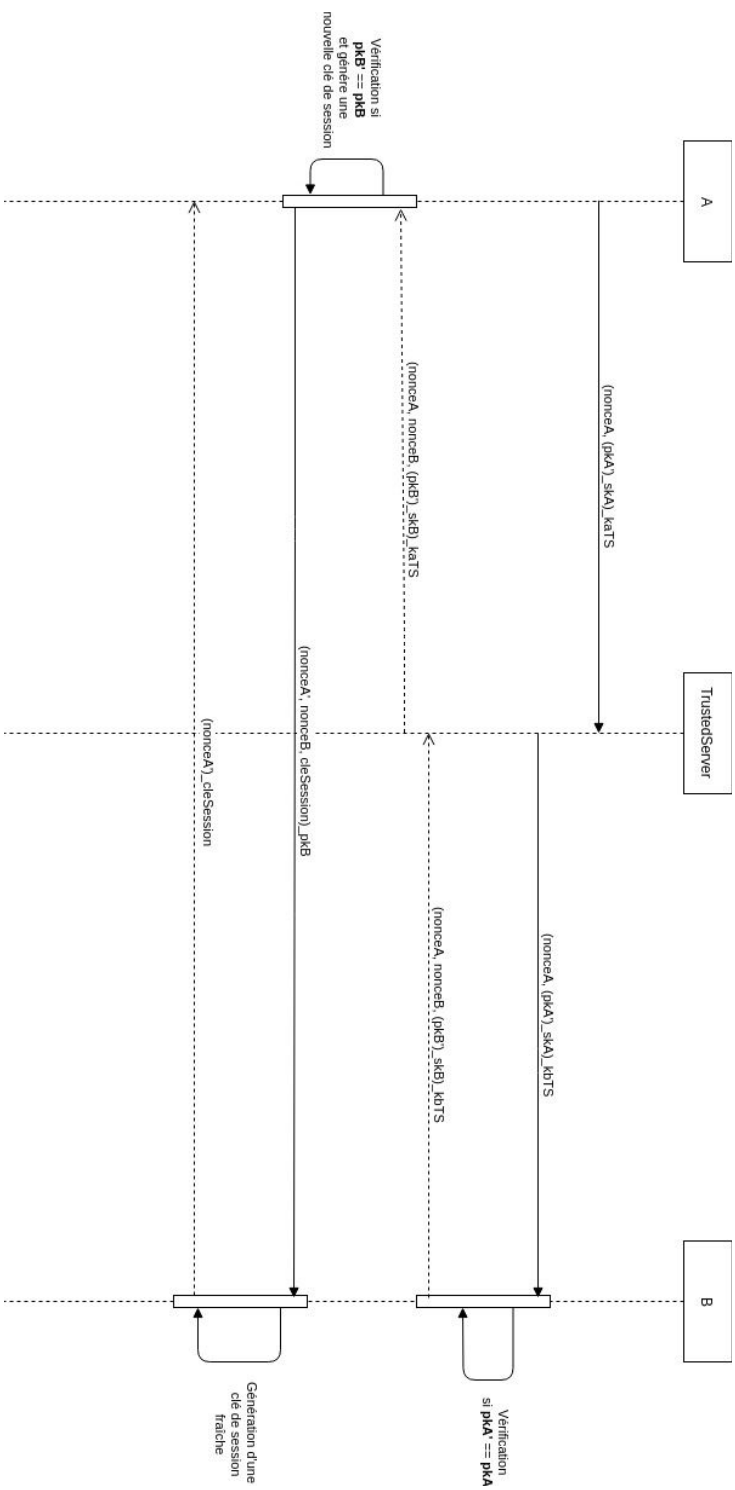
- A est sûr de communiquer avec TS grâce au chiffrement par la clé *kaTS*
- B est sûr de communiquer avec TS grâce au chiffrement par la clé *kbTS*
- A est sûr d'envoyer *cleSession* à B car il a vérifié sa clé publique par le biais de TS auparavant

Confidentialité :

- Personne d'autre que A, B et TS peuvent connaître nonceA et nonceB car ceux-ci sont chiffrés par *kaTS* et *kbTS*
- Seuls A et B connaissent *cleSession* car elle est chiffrée par la clé publique vérifiée de B

PROCÉDURE

Voici un schéma détaillant les fonctionnalités du protocole :



FONCTIONNEMENT

Le fonctionnement de ce protocole réside sur le fait que les deux acteurs, appelons-les *Alice* et *Bob,* connaissent déjà la même entité, le serveur de confiance *TS*.

Partons du principe qu'*Alice* initie la conversation avec *Bob*. Pour vérifier l'identité d'*Alice,* elle transmet sa clé publique encryptée par sa clé privée. Cela permet donc de retrouver la clé publique si on l'a possède déjà, ce que *Bob* doit connaître d'après les hypothèses de départ. De plus, encrypté avec sa clé privée permet d'assurer que c'est *Alice* qui a composé ce message.

Les nonce permettent de s'assurer aussi qu'il n'y est pas d'attaque de type "*man in the middle*".

Le fait d'utiliser le serveur de confiance *TS* permet aussi de s'assurer de la qualité de l'information dans les messages reçus. Ces derniers encryptés avec les clés symétriques garantissent qu'aucun attaquant ne peut comprendre le message même s'il est intercepté.

TESTS

Afin de pouvoir vérifier les propriétés de sécurité, nous avons décidé de modéliser notre protocole avec **AVISPA**. **AVISPA** est un outils de vérification de protocole. Pour ce faire, nous avons traduit notre protocole en langage HLPSL (*cf. annexe*) qui permet de tenter des attaques sur le protocole par un intrus appelé *i*. Pour ce faire, l'intru possède des connaissances et avec celles-ci, il va essayer de porter atteinte au protocole en récupérant des informations ou en usurpant des identités. Les connaissances de cet intrus sont les suivantes :

- clé publique pkI
- clé secrète skI
- clé publique pkA
- clé publique pkB

RÉSULTATS

D'après les résultats d'**AVISPA**, on remarque que les nonces ainsi que la clé de session générée ne peut être récupérée. De plus, il n'y a aucune possibilité d'usurpation sur la clé de session puisque la propriété "*authentication_on*" est respectée.

```
%% Translation of protocole.hlpsl
%% IF output in ./protocole.if
%% Constraint Logic-based ATtack SEarcher (CL-ATSE) Version 2.5-18 (2012-septembre-
26).

SUMMARY
  SAFE

DETAILS
  BOUNDED NUMBER OF SESSIONS
  TYPED MODEL
  BOUNDED SPEC. READING DEPTH

PROTOCOL
  protocole.if

GOAL
  As specified

BACKEND
  CL-AtSe

STATISTICS
  Analysed   : 6 states
  Reachable  : 4 states
  Translation: 0.01 seconds
  Computation: 0.00 seconds
```

COÛTS

On va calculer le coût du protocole:

$f(P) = f(\{ \langle nonceA, \{pkA'\}_{priv(skA)} \rangle \}_{sym(kaTS)})$

$+ f(\{ \langle nonceA, \{pkA'\}_{priv(skA)} \rangle \}_{sym(kbTS)})$

$+ f(\{ \langle nonceA, \langle nonceB, \{pkB'\}_{priv(skB)} \rangle\rangle \}_{sym(kbTS)})$

$+ f(\{ \langle nonceA, \langle nonceB, \{pkB'\}_{priv(skB)} \rangle\rangle \}_{sym(kaTS)})$

$+ f(\{ \langle nonceA', \langle nonceB, cleSession \rangle\rangle \}_{pub(pkB)})$

$+ f(\{ nonceA' \}_{sym(cleSession)})$

Détaillons :

$f(\{ \langle nonceA, \{pkA'\}_{priv(skA)} \rangle \}_{sym(kaTS)})$

$= 10 + f(\langle nonceA, \{pkA'\}_{priv(skA)} \rangle) + f(kaTS)$

$= 10 + 50 + f(nonceA) + \{pkA'\}_{priv(skA)} + 1$

$= 61 + 1 + 1 + f(pkA') + k(skA)$

$= 63 + 1 + 1$

$= 65$

$f(\{ \langle nonceA, \{pkA'\}_{priv(skA)} \rangle \}_{sym(kbTS)})$

$= f(\{ \langle nonceA, \{pkA'\}_{priv(skA)} \rangle \}_{sym(kaTS)})$

$= 65$

$f(\{ \langle nonceA, \langle nonceB, \{pkB'\}_{priv(skB)} \rangle\rangle \}_{sym(kbTS)})$

$= 10 + f(\langle nonceA, \langle nonceB, \{pkB'\}_{priv(skB)} \rangle\rangle) + f(kbTS)$

$= 10 + 50 + f(nonceA) + f(\langle nonceB, \{pkB'\}_{priv(skB)} \rangle\rangle) + 1$

$= 61 + 1 + 50 + f(nonceB) + f(\{pkB'\}_{priv(skB)})$

$= 112 + 1 + 1 + f(pkB) + f(skB)$

$= 114 + 1 + 1$

$= 116$

$f(\{ \langle nonceA, \langle nonceB, \{pkB'\}_{priv(skB)} \rangle\rangle \}_{sym(kaTS)})$

$= f(\{ \langle nonceA, \langle nonceB, \{pkB'\}_{priv(skB)} \rangle\rangle \}_{sym(kbTS)})$

$= 116$

$f(\{ \langle nonceA', \langle nonceB, cleSession \rangle\rangle \}_{pub(pkB)})$

$= 1 + f(\langle nonceA', \langle nonceB, cleSession \rangle\rangle) + f(pkB)$

$= 1 + 50 + f(nonceA) + f(\langle nonceB, cleSession \rangle) + 1$

$= 52 + 1 + 50 + f(nonceB) + f(cleSession)$

$= 103 + 1 + 1$

$= 105$

$f(\{ nonceA' \}_{sym(cleSession)})$

$= 10 + f(nonceA) + f(cleSession)$

$= 10 + 1 + 1$

$= 12$

Ce qui nous donne donc au final :

f(P) = 65 + 65 + 116 + 116 + 105 + 12

= **479**

CONCLUSION

D'après nos tests et nos connaissances, notre protocole n'est pas attaquable.

Cependant, notre système de dialogue en utilisant le serveur de confiance duplique la quasi totalité des messages. Mais d'un point de vue plus précis, le serveur ne fait pas le calcul de l'ensemble du message, puisqu'au final il le retransmet en utilisant une autre clé symétrique, sans s'occuper des informations. Donc notre coût en calcul est amoindri par rapport à f(P) calculé précédemment. De plus, comparé au coût du protocole de Woo Lam, qui est de 1071, notre protocole est moins coûteux.

ANNEXE - protocole.hlpsl

```
%% PROTOCOL: Echange de clé
%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
%% définition du rôle Alice, initiant le protocole
%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

role alice (A, B, TS: agent,
            PKa, PKb: public_key,
            KAts: symmetric_key,
            SND, RCV: channel(dy))
played_by A def=

  local State: nat,
        Na, Nb: text,
        PKaBIS, PKbBIS: public_key,
        CleSession : symmetric_key

  const ok: text

  init State:=0

  transition

    01.  State=0 /\ RCV(start) =|>
            State':=1 /\
            Na':=new() /\
            secret(Na', na, {A,TS}) /\
            PKaBIS' := PKa /\
            SND({Na'.{PKaBIS'}_inv(PKa)}_KAts)


    05.  State=1 /\ RCV({Na.Nb'.{PKbBIS'}_inv(PKb)}_KAts) =|>
            State':=2 /\
            equal(PKb, PKbBIS') /\
            CleSession':=new() /\
            secret(CleSession', cleSession, {A,B}) /\
            witness(A, B, a_b_CleSession, CleSession') /\
            Na':=new() /\
            secret(Na', na, {A,TS}) /\
            SND({Na'.Nb'.CleSession'}_PKb)

    07.  State=2 /\ RCV({Na}_CleSession) =|>
            State':=3

end role
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
%% définition du rôle Bob
%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

role bob (A, B, TS: agent,
            PKa, PKb: public_key,
            KBts: symmetric_key,
            SND, RCV: channel(dy))
played_by B def=

  local State: nat,
        Na, Nb: text,
        PKaBIS, PKbBIS: public_key,
        CleSession: symmetric_key

  init State:=0

  transition

    03.  State=0 /\ RCV({Na'.{PKaBIS'}_inv(PKa)}_KBts) =|>
            State':=1 /\
            equal(PKa, PKaBIS') /\
            Nb':=new() /\
            secret(Nb', nb, {B,TS}) /\
            PKbBIS':=PKb /\
            SND({Na'.Nb'.{PKbBIS'}_inv(PKb)}_KBts)


    06.  State=1 /\ RCV({Na'.Nb.CleSession'}_PKb) =|>
            State':=2 /\
            request(A, B, a_b_CleSession, CleSession') /\
            SND({Na'}_CleSession')



end role
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
%% définition du rôle Serveur de Confiance
%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

role ts (A, B, TS: agent,
            PKa, PKb: public_key,
            KAts, KBts: symmetric_key,
            SND, RCV: channel(dy))
played_by TS def=

  local State: nat,
        Na, Nb: text,
        PKaBIS, PKbBIS: public_key,
        CleSession: symmetric_key

  const ok: text

  init State:=0

  transition

    02.  State=0 /\ RCV({Na'.{PKaBIS'}_inv(PKa)}_KAts) =|>
            SND({Na'.{PKaBIS'}_inv(PKa)}_KBts)

    04.  State=1 /\ RCV({Na'.Nb'.{PKbBIS'}_inv(PKb)}_KBts) =|>
            State':=2 /\
            SND({Na'.Nb'.{PKbBIS'}_inv(PKb)}_KAts)


end role

'%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
%% définition du rôle Session
%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

role session(A, B, TS: agent, PKa, PKb: public_key, KAts, KBts: symmetric_key) def=

  local SA, RA, SB, RB, SS, RS: channel(dy)

  composition

        alice(A,B,TS,PKa,PKb,KAts,SA,RA) /\
        bob(A,B,TS,PKa,PKb,KBts,SB,RB) /\
        ts(A,B,TS,PKa,PKb,KAts,KBts,SS,RS)
end role
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
%% définition du Scenario
%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

role environment() def=


    const a, b, ts: agent,
          pka, pkb, pki: public_key,
          cleSession: symmetric_key,
          kats, kbts : symmetric_key,
          na, nb, a_b_CleSession : protocol_id,
          h : hash_func


    intruder_knowledge = {a, b, ts, pka, pkb, pki, inv(pki), h}


    composition

        session(a,b,ts,pka,pkb,kats,kbts)

end role

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
%% définition des Propriétés à vérifier
%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

goal
    secrecy_of na
    secrecy_of nb
    secrecy_of cleSession
    authentication_on a_b_CleSession
end goal


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
%% lancement du rôle principal
%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

environment()
```