RAPPORT
Sécurité des Protocoles

Part 1 (Design of a key exchange protocol)

Guillaume Zablot
Antoine Courtil

09/01/2019
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** : $\langle A, \{ \langle B, nonceA \rangle \} \rangle_{sym(kaTS)}$

TS **B** : $\{ A \}_{sym(kbTS)}$

B **TS** : $\{ nonceB \}_{sym(kbTS)}$

TS **A** : $\{ \langle cleSession, \{nonceB\}_{sym(kbTS)} \rangle \}_{sym(kaTS)}$

TS **B** : $\{ \langle cleSession, \{nonceA\}_{sym(kaTS)} \rangle \}_{sym(kbTS)}$

A **B** : $\{ \{nonceB\}_{sym(kbTS)} \}_{sym(cleSession)}$

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

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 A *kaTS*
    - clé symétrique avec BTS *kbTS*

PROPRIÉTÉS DE SÉCURITÉ

Authentification :

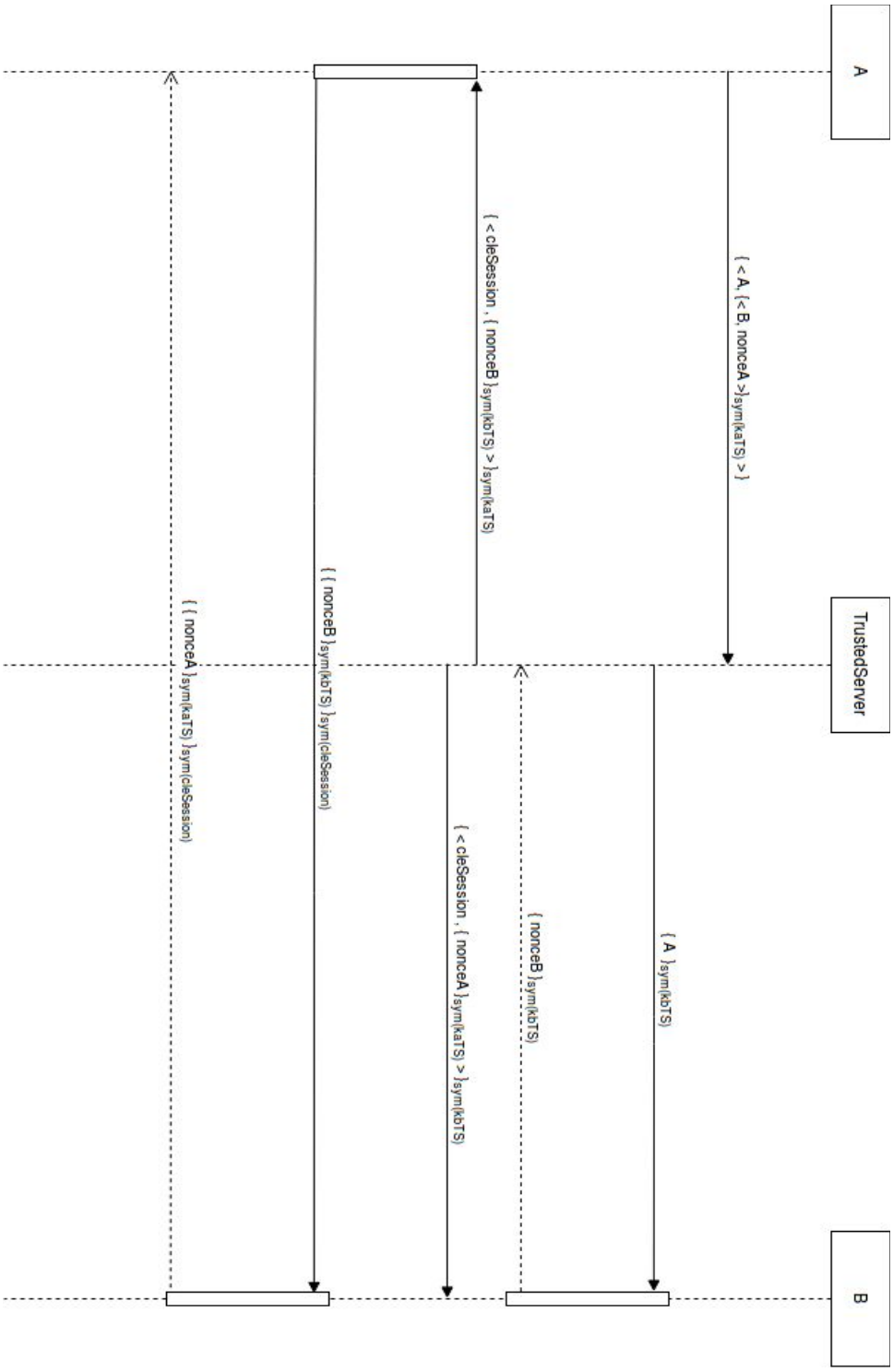
- A est sûr de communiquer avec TS grâce au chiffrement par la clé *kaTS*
- B est sûr de communiquer avec TS grâce au chiffrement par la clé *kbTS*
- A est sûr que seul B pourra déchiffrer nonceB grâce au chiffrement de TS par la clé *kbTS*
- B est sûr que seul A pourra déchiffrer nonceA grâce au chiffrement de TS par la clé *kaTS*

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 *kaTS* et *kbTS*

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 ce faire, *Alice* transmet sa requête à *TS*, qui est composée de son identité en clair et l'identité de *Bob* ainsi qu'un nonce, tous deux chiffrés avec la clé symétrique *kaTS*. Ceci implique donc que seul *TS* sait avec qui veut communiquer *Alice*.

TS va donc informer *Bob* qu'*Alice* veut communiquer avec. *Bob* va donc répondre avec un nonce, chiffré avec *kbTS* pour que seul *TS* soit au courant de sa valeur.

Une fois que *TS* possède les deux nonces, il va générer une clé de session et l'envoyer respectivement à *Alice* et *Bob* chiffré par *kaTS* et *kbTS*. De plus, *TS* va transmettre à *Alice* la valeur du nonce de *Bob*, chiffré par *kbTS* pour que seul *Bob* puisse déchiffrer son nonce. Et inversement.

Pour vérifier la clé de session, *Alice* va envoyer le nonce de *Bob* chiffré, rechiffré avec cette nouvelle clé. En recevant ce message, seul *Bob* peut le déchiffrer en connaissant à la fois la clé de session et *kbTS*. Puis il va faire de même en envoyant le nonce d'*Alice*. Si *Alice* reçoit son nonce, alors la clé de session est valide.

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.

Nous avons estimé une pseudo-attaque. En effet, un attaquant *Isaac* peut se faire passer pour *Alice* dans le premier message, puisque son identité est envoyée en clair. Cependant, *Isaac* ne connaissant pas *kaTS*, il va encoder le nonce avec *kiTS*. De ce fait, quand *Alice* va reçevoir un message non demandé avec une nouvelle clé de session, elle va s'apercevoir que le nonce est erroné et donc de ne pas prendre en compte cette nouvelle clé. Sans réponse, *Bob* fera de même.

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 langagte 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
- A
- B
- TS

RÉSULTATS

D'après les résultats d'**AVISPA**, on remarque que les nonces ainsi que la clé de session générée ne peuvent 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

PROTOCOL
  protocole.if

GOAL
  As specified

BACKEND
  CL-AtSe

STATISTICS
  Analysed   : 3 states
  Reachable  : 3 states
  Translation: 0.01 seconds
  Computation: 0.00 seconds
```

COÛTS

On va calculer le coût du protocole:

$$f(P) = f(\ <A, \{\ <B, nonceA>\ \}>_{sym(kaTS)}\ )$$
$$+ f(\ \{\ A\ \}_{sym(kbTS)}\ )$$
$$+ f(\ \{\ nonceB\ \}_{sym(kbTS)}\ )$$
$$+ f(\ \{\ <cleSession, \{nonceB\}_{sym(kbTS)}>\ \}_{sym(kaTS)}\ )$$
$$+ f(\{\ \{\ <cleSession, \{nonceA\}_{sym(kaTS)}>\ \}_{sym(kbTS)}\ )$$
$$+ f(\ \{\ \{nonceB\}_{sym(kbTS)}\ \}_{sym(cleSession)}\ )$$
$$+ f(\ \{\ \{nonceA\}_{sym(kaTS)}\ \}_{sym(cleSession)}\ )$$

<u>Détaillons :</u>

$$f(\ <A, \{\ <B, nonceA>\ \}>_{sym(kaTS)}\ )$$
$$= 50 + f(A) + f(\ \{\ <B, nonceA>\ \}>_{sym(kaTS)}\ )$$
$$= 50 + 1 + 10 + f(<B, nonceA>) + f(kaTS)$$
$$= 61 + 50 + f(B) + f(nonceA) + 1$$
$$= 112 + 1 + 1$$
$$= 114$$

$$f(\ \{\ A\ \}_{sym(kbTS)}\ )$$
$$= 10 + f(A) + f(kbTS)$$
$$= 10 + 1 + 1$$
$$= 12$$

$f(\{ \text{nonceB} \}_{sym(kbTS)})$

$= 10 + f(\text{nonceB}) + f(\text{kbTS})$

$= 10 + 1 + 1$

$= 12$

$f(\{ <\text{cleSession}, \{\text{nonceB}\}_{sym(kbTS)}> \}_{sym(kaTS)})$

$= 10 + f(<\text{cleSession}, \{\text{nonceB}\}_{sym(kbTS)}>) + f(\text{kaTS})$

$= 10 + 50 + f(\text{cleSession}) + f(\{\text{nonceB}\}_{sym(kbTS)}) + 1$

$= 61 + 1 + 10 + f(\text{nonceB}) + k(\text{kbTS})$

$= 72 + 1 + 1$

$= 74$

$f(\{ <\text{cleSession}, \{\text{nonceA}\}_{sym(kaTS)}> \}_{sym(kbTS)})$

$= f(\{ <\text{cleSession}, \{\text{nonceB}\}_{sym(kbTS)}> \}_{sym(kaTS)})$

$= 74$

$f(\{ \{\text{nonceB}\}_{sym(kbTS)} \}_{sym(cleSession)})$

$= 10 + f(\{\text{nonceB}\}_{sym(kbTS)}) + f(\text{cleSession})$

$= 10 + 10 + f(\text{nonceB}) + f(\text{kbTS}) + 1$

$= 21 + 1 + 1$

$= 23$

$f(\{ \{\text{nonceA}\}_{sym(kaTS)} \}_{sym(cleSession)})$

$= f(\{ \{\text{nonceB}\}_{sym(kbTS)} \}_{sym(cleSession)})$

$= 23$

Ce qui nous donne donc au final :

f(P) = 114 + 12 + 12 + 74 + 74 + 23 + 23

= **332**

CONCLUSION

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

Cependant, notre système de dialogue repose sur le fait de retransmettre des nonces chiffrés avec une clé que l'on ne connaît pas. Par exemple, *Alice* va retransmettre $\{nonceB\}_{sym(kbTS)}$ à *Bob* sans pour autant effectuer le calcul puisqu'*Alice* le reçoit de *TS* et ne peut le modifier. On peut donc en conclure que 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, KBts : symmetric_key

  const ok: text

  init State:=0

  transition

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


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

    08. State=2 /\ RCV({{Na}_KAts}_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, KAts: symmetric_key

  init State:=0

  transition

    03. State=0 /\ RCV({A}_KBts) =|>
            State':=1 /\
            Nb':=new() /\
            secret(Nb', nb, {A,B}) /\
            SND({Nb'}_KBts)

    05. State=1 /\ RCV({CleSession'.{Na'}_KAts}_KBts) =|>
            State':=2 /\
            request(A, B, a_b_CleSession, CleSession')

    07. State=2 /\ RCV({{Nb}_KBts}_CleSession) =|>
            State':=3 /\
            SND({{Na}_KAts}_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(A.{B.Na'}_KAts) =|>
            State':=1 /\
            SND({A}_KBts)

    04. State=1 /\ RCV({Nb'}_KBts) =|>
            State':=2 /\
            CleSession':=new() /\
            secret(CleSession', cleSession, {A,B}) /\
            witness(A, B, a_b_CleSession, CleSession') /\
            SND({CleSession'.{Nb'}_KBts}_KAts) /\
            SND({CleSession'.{Na}_KAts}_KBts)

'
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()
```