Vérification automatisée des protocoles cryptographiques

Claire Sondès LARAFA

ANSSI/SDE/LRP

26 mars 2014

Agence nationale de la sécurité des systèmes d'information

Plan

Plan

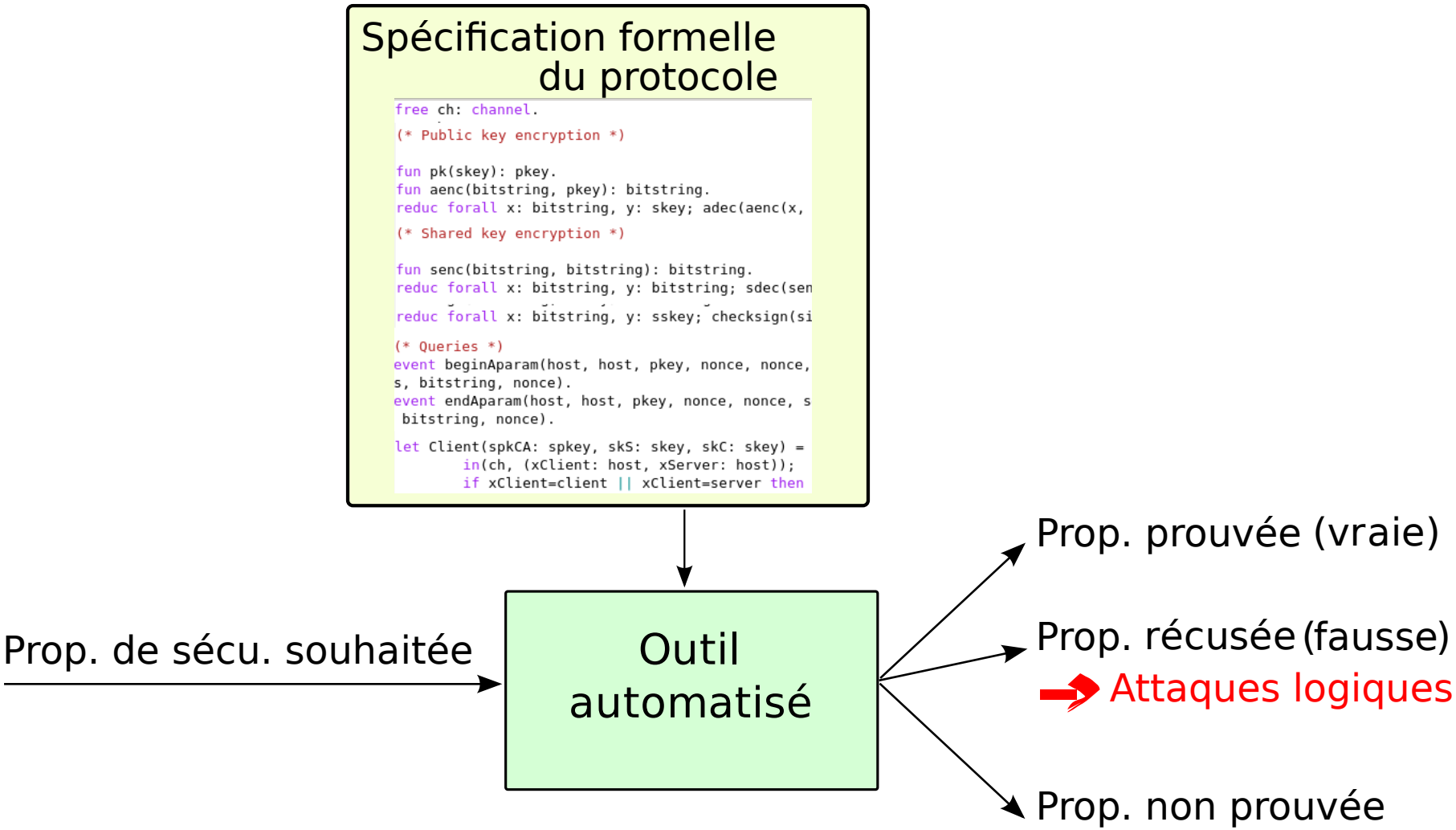
Introduction : qu'est ce qu'un protocole cryptographique ?

- Un protocole cryptographique est un protocole qui offre des services de sécurité au moyen de méthodes cryptographiques
- Services de sécurité : confidentialité, authentification, persistance de confidentialité (PFS), protection de la vie privée, etc.
- Méthodes cryptographiques : AES, RC4, RSA, DSA, SHA256, etc.
- Protocoles cryptographiques : IKE, SSL/TLS, WPA, EAP, etc.

Introduction : qu'est ce qu'un protocole cryptographique ?

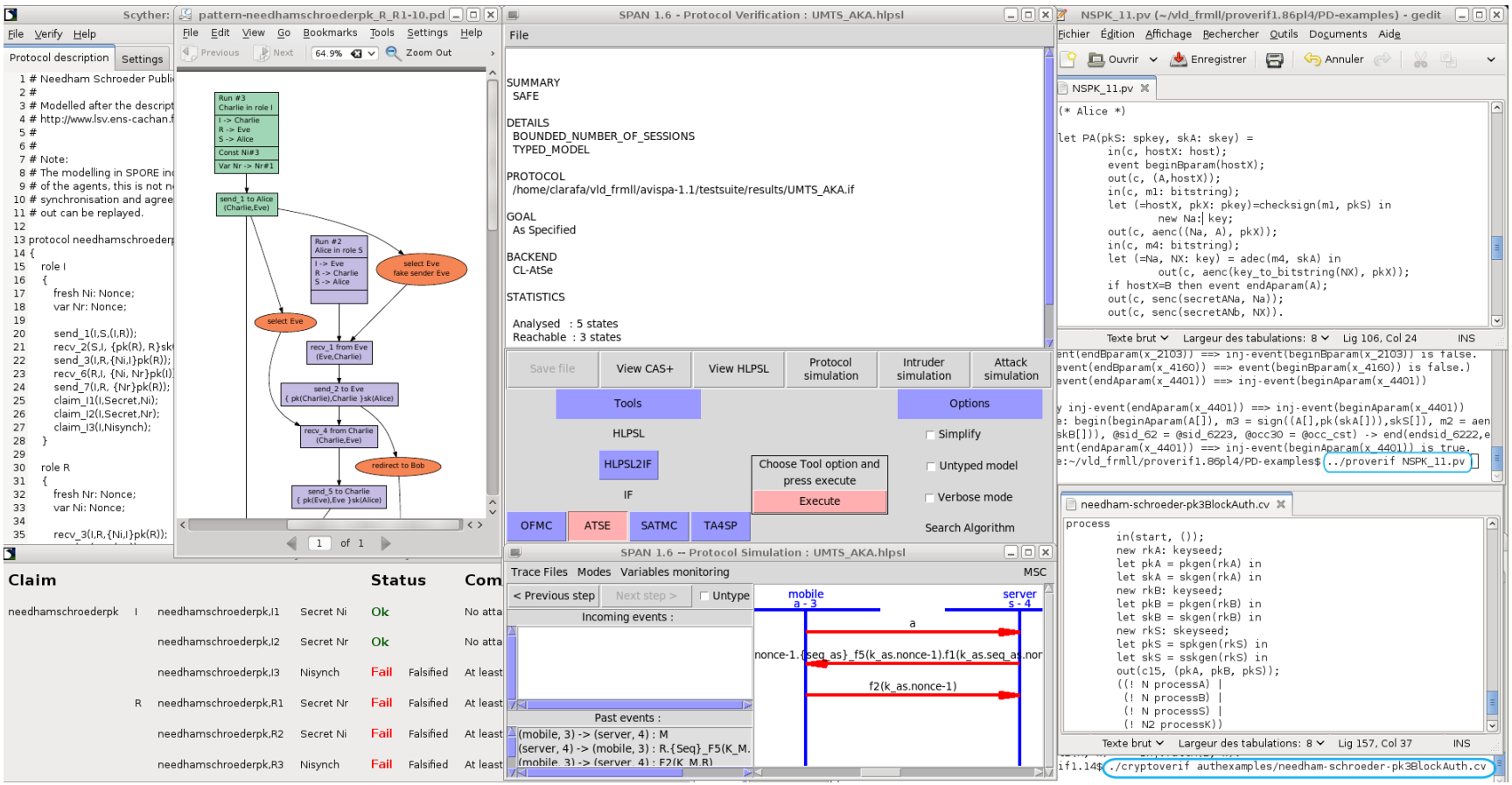
- Un protocole cryptographique est un protocole qui offre des services de sécurité au moyen de méthodes cryptographiques
- Services de sécurité : confidentialité, authentification, persistance de confidentialité (PFS), protection de la vie privée, etc.
- Méthodes cryptographiques : AES, RC4, RSA, DSA, SHA256, etc.
- Protocoles cryptographiques : IKE, SSL/TLS, WPA, EAP, etc.
- Bibliothèques en ligne :
  - Clarck et Jacob : `www.csl.sri.com/users/millen/capsl/library.html`
  - SPORE : www.lsv.ens-cachan.fr/Software/spore/table.html

Introduction : vérification automatisée



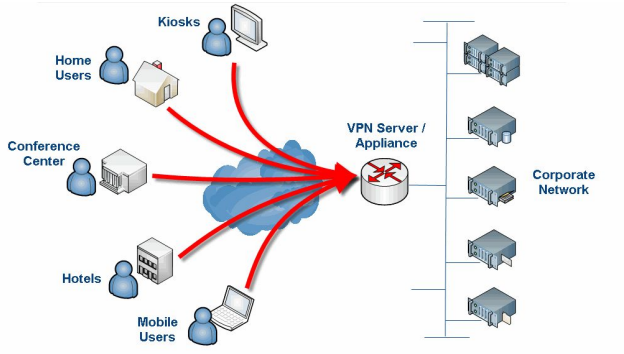
Introduction : outils automatisés

Scyther AVISPA ProVerif



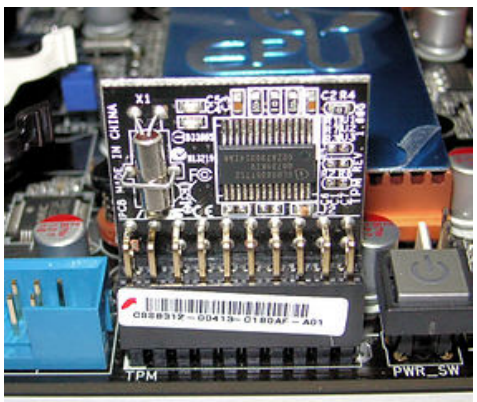
Introduction : domaines d'utilisation

- Télécom : protocoles réseau



source : www.petri.co.il

- Puces électroniques : protocoles d'authentification dans TPM

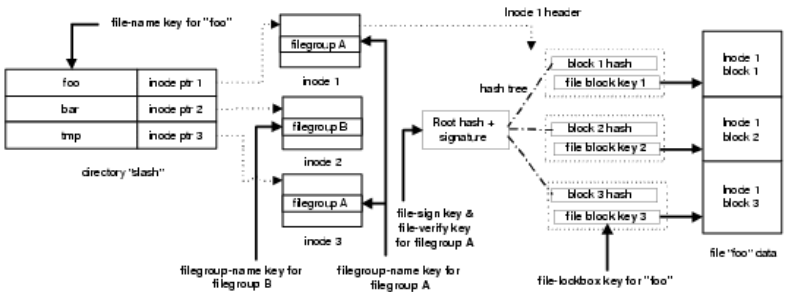


source : wikipedia

- Vote électronique : protection de la vie privée

source : www.macgeneration.com

- Système de fichier : protocole de stockage sécurisé dans Plutus



source : Usenix

Introduction : un exemple simple

Protocole :

$$A \rightarrow B :< A, \{K\}_{pk(B)} >$$
$$B \rightarrow A :< B, \{K\}_{pk(A)} >$$

Hypothèses :

$pk(A)$ et $pk(B)$: clés publiques certifiées

Objectif du protocole :

Établir une clé de session K entre A et B

Services de sécurité attendus :

Introduction : un exemple simple

Protocole :

$$A \rightarrow B :< A, \{K\}_{pk(B)} >$$
$$B \rightarrow A :< B, \{K\}_{pk(A)} >$$

Hypothèses :

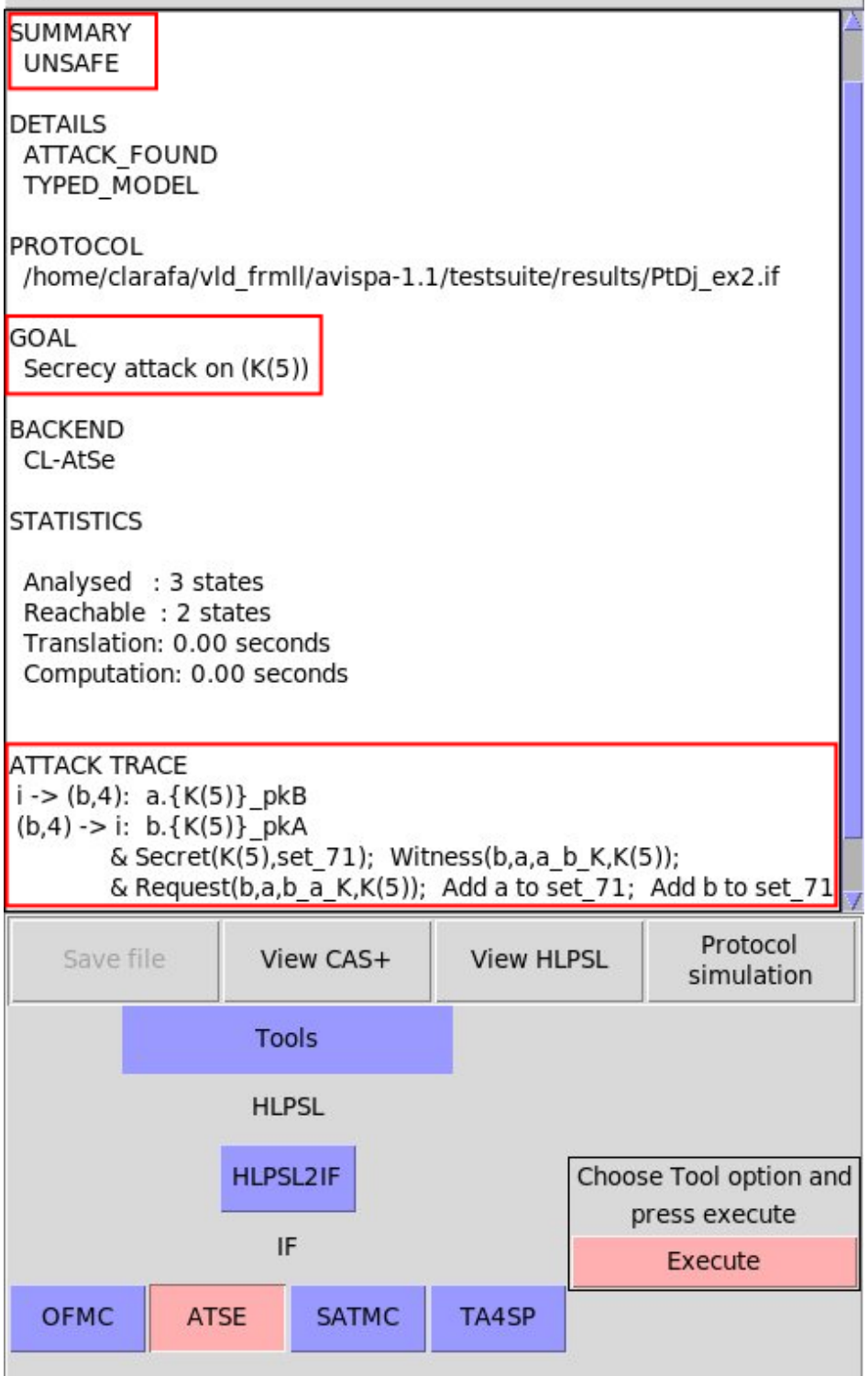
$pk(A)$ et $pk(B)$: clés publiques certifiées

Objectif du protocole :

Établir une clé de session K entre A et B

Services de sécurité attendus :

1. secret de K,
2. authentification de A vis-à-vis de B
3. authentification de B vis-à-vis de A



Attaque détectée par AVISPA :

$$I(A) \rightarrow B : <A, \{K_I\}_{pk(B)}>$$
$$B \rightarrow I(A) : <B, \{K_I\}_{pk(A)}>$$

$\rightarrow$ usurpation d'identité

$\Rightarrow$Violation du secret de K et de l'authentification de A vis-à-vis de B

```
8. The message pk(y_1085) that the attacker may have by 4 may be received at input {19}.
The message (pk(y_1085),aenc(K_1093,pk(skB[]))) that the attacker may have by 7 may be received at input
The event beginAparam(pk(y_1085)) (with environment m2 = (pk(y_1085),aenc(K_1093,pk(skB[]))), pkY = pk(y
So the message (pk(skB[]),aenc(K_1093,pk(y_1085))) may be sent to the attacker at output {24}.
attacker((pk(skB[]),aenc(K_1093,pk(y_1085)))).

9. By 8, the attacker may know (pk(skB[]),aenc(K_1093,pk(y_1085))).
Using the 1th inverse of function 2-tuple the attacker may obtain aenc(K_1093,pk(y_1085)).
attacker(aenc(K_1093,pk(y_1085))).

10. By 9, the attacker may know aenc(K_1093,pk(y_1085)).
By 3, the attacker may know y_1085.
Using the function adec the attacker may obtain K_1093.
attacker(K_1093).

11. By 10, the attacker may know K_1093.
By 2, the attacker may know pk(skA[]).
Using the function aenc the attacker may obtain aenc(K_1093,pk(skA[])).
attacker(aenc(K_1093,pk(skA[]))).

12. By 1, the attacker may know pk(skB[]).
By 11, the attacker may know aenc(K_1093,pk(skA[])).
Using the function 2-tuple the attacker may obtain (pk(skB[]),aenc(K_1093,pk(skA[]))).
attacker((pk(skB[]),aenc(K_1093,pk(skA[])))).

13. The message pk(skB[]) that the attacker may have by 1 may be received at input {8}.
The message (pk(skB[]),aenc(K_1093,pk(skA[]))) that the attacker may have by 12 may be received at input {12}.
So event endAparam(pk(skA[])) may be executed at {16} in session endsid_1091.
end(endsid_1091,endAparam(pk(skA[]))).

The event endAparam(pk(skA_1097)) is executed in session a_1094.
A trace has been found.
RESULT inj-event(endAparam(x_630)) ==> inj-event(beginAparam(x_630)) is false.
RESULT (even event(endAparam(x_1070)) ==> event(beginAparam(x_1070)) is false.)
```

Attaque détectée par ProVerif

Attaque détectée par ProVerif :

Hypothèse : l'attaquant dispose d'une clé $pk(I)$ connue de A et de B

$$A \rightarrow I(B) :< A, \{K\}_{pk(B)} >$$

$$I \rightarrow B :< I, \{K\}_{pk(B)} >$$

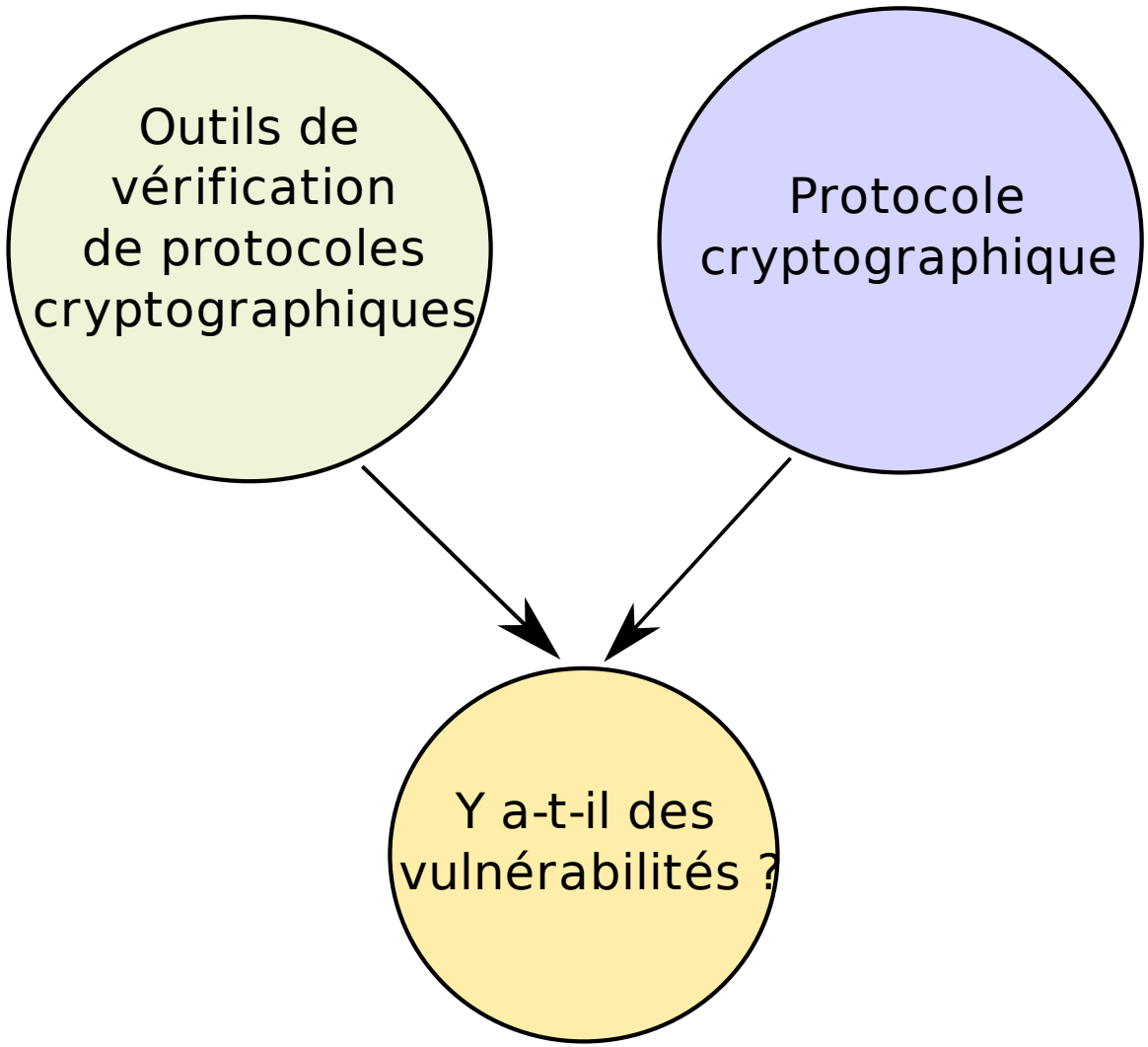
$$B \rightarrow I :< B, \{K\}_{pk(I)} >$$

$$I(B) \rightarrow A :< B, \{K\}_{pk(A)} >$$

$\rightarrow$ Homme du milieu.

$\Rightarrow$ Violation du secret de K et de l'authentification de B vis-à-vis de A

Introduction : tester les outils de vérification de protocoles cryptographiques



Plan

Modèle formel : le protocole NSPK

- Proposé par Needham et Schroeder (1978)
- Utilise une cryptographie à clés publiques
- Objectif : établir une clé de session K commune à A et B
- Hypothèses: $pk(A)$ et $pk(B)$ clés publiques certifiées

Modèle formel : le protocole NSPK

- Proposé par Needham et Schroeder (1978)
- Utilise une cryptographie à clés publiques
- Objectif : établir une clé de session K commune à A et B
- Hypothèses: $pk(A)$ et $pk(B)$ clés publiques certifiées
- Déroulement :
  A et B choisissent aléatoirement deux nombres Na et Nb
  $A \rightarrow B : \{< Na, A >\}_{pk_B}$
  $B \rightarrow A : \{< Na, Nb >\}_{pk_A}$
  $A \rightarrow B : \{Nb\}_{pk_B}$
  A et B calculent K à partir de Na et Nb

Modèle formel : le protocole NSPK

- Proposé par Needham et Schroeder (1978)
- Utilise une cryptographie à clés publiques
- Objectif : établir une clé de session K commune à A et B
- Hypothèses: $pk(A)$ et $pk(B)$ clés publiques certifiées
- Déroulement :
  A et B choisissent aléatoirement deux nombres Na et Nb
  $A \rightarrow B : \{< Na, A >\}_{pk_B}$
  $B \rightarrow A : \{< Na, Nb >\}_{pk_A}$
  $A \rightarrow B : \{Nb\}_{pk_B}$
  A et B calculent K à partir de Na et Nb

- Services de sécurité attendus :
  1. secret de la clé de session,
  2. authentification mutuelle entre A et B

Modèle formel : le protocole NSPK

- Proposé par Needham et Schroeder (1978)
- Utilise une cryptographie à clés publiques
- Objectif : établir une clé de session K commune à A et B
- Hypothèses: $pk(A)$ et $pk(B)$ clés publiques certifiées
- Déroulement :
  A et B choisissent aléatoirement deux nombres Na et Nb
  $A \rightarrow B : \{< Na, A >\}_{pk_B}$
  $B \rightarrow A : \{< Na, Nb >\}_{pk_A}$
  $A \rightarrow B : \{Nb\}_{pk_B}$
  A et B calculent K à partir de Na et Nb

- Services de sécurité attendus :
  1. secret de la clé de session,
  2. authentification mutuelle entre A et B

- Vérification en 1990 : Burrow, Abadi et Needham en utilisant la logique BAN

$\rightarrow$ Protocole démontré sûr

Modèle formel : naissance de l'automatisation

- En 1996, Lowe trouve une attaque au moyen de son vérifieur FDR

- Déroulement :

$A \rightarrow I : \{< Na, A >\}_{pk_I}$

$I(A) \rightarrow B : \{< Na, A >\}_{pk_B}$

$B \rightarrow I(A) : \{< Na, Nb >\}_{pk_A}$

$I \rightarrow A : \{< Na, Nb >\}_{pk_A}$

$A \rightarrow I : \{Nb\}_{pk_I}$

$I(A) \rightarrow B : \{Nb\}_{pk_B}$

Modèle formel : naissance de l'automatisation

- En 1996, Lowe trouve une attaque au moyen de son vérifieur FDR
- Déroulement :

$A \rightarrow I : \{< Na, A >\}_{pk_I}$

$I(A) \rightarrow B : \{< Na, A >\}_{pk_B}$

$B \rightarrow I(A) : \{< Na, Nb >\}_{pk_A}$

$I \rightarrow A : \{< Na, Nb >\}_{pk_A}$

$A \rightarrow I : \{Nb\}_{pk_I}$

$I(A) \rightarrow B : \{Nb\}_{pk_B}$

- L'attaquant I détient la clé de session

Modèle formel : naissance de l'automatisation

- En 1996, Lowe trouve une attaque au moyen de son vérifieur FDR
- Déroulement :

$A \rightarrow I : \{< Na, A >\}_{pk_I}$

$I(A) \rightarrow B : \{< Na, A >\}_{pk_B}$

$B \rightarrow I(A) : \{< Na, Nb >\}_{pk_A}$

$I \rightarrow A : \{< Na, Nb >\}_{pk_A}$

$A \rightarrow I : \{Nb\}_{pk_I}$

$I(A) \rightarrow B : \{Nb\}_{pk_B}$

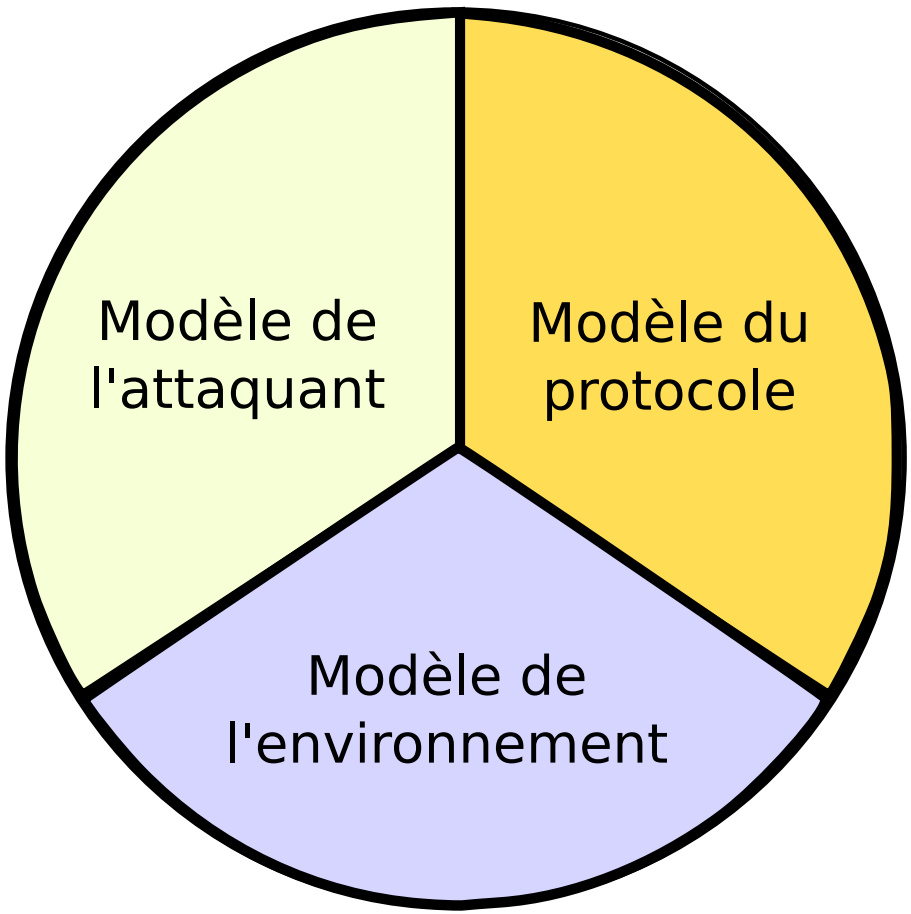
- L'attaquant I détient la clé de session

$\Rightarrow$ La vérification d'absence d'attaques est un problème complexe

$\Rightarrow$ L'automatisation des méthodes permet de prendre en compte un attaquant ayant des possibilités plus étendues

Modèle formel : vers une automatisation des preuves

Un outil automatisé met en œuvre plusieurs modèles



Modèle formel : l'attaquant de Dolev-Yao

- Modèle apparu en 1983
- Attaquant présent partout dans le réseau
- Il peut écouter, modifier, supprimer et/ou rejouer les messages
- Il peut forger des nouveaux messages en utilisant tout le matériel public

$\Rightarrow$ Attaquant actif

Modèle formel : l'attaquant de Dolev-Yao

- Modèle apparu en 1983
- Attaquant présent partout dans le réseau
- Il peut écouter, modifier, supprimer et/ou rejouer les messages
- Il peut forger des nouveaux messages en utilisant tout le matériel public

$\Rightarrow$ Attaquant actif

- Il peut seulement déchiffrer quand il dispose de la clé de déchiffrement
- Il est incapable de casser la cryptographie

$\Rightarrow$ Ses pouvoirs sont limités comparé à un attaquant réel

Modèle formel : système de déduction

Constructeurs :

$$\frac{x \qquad y}{\langle x,y \rangle}\ \text{pair}$$

$$\frac{x \qquad y}{senc(x,y)}\ \text{senc}$$

$$\frac{x \qquad y}{aenc(x,y)}\ \text{aenc}$$

Destructeurs :

$$\frac{\langle x,y \rangle}{x}\ \pi_1$$

$$\frac{\langle x,y \rangle}{y}\ \pi_2$$

$$\frac{senc(x,y) \qquad y}{x}\ \text{sdec}$$

$$\frac{aenc(x,y) \qquad sk(y)}{x}\ \text{adec}$$

Modèle formel : exemple d'utilisation du système de déduction

- Attaque de Lowe sur NSPK :

$A \rightarrow I : \{< Na, A >\}_{pk_I}$

$I(A) \rightarrow B : \{< Na, A >\}_{pk_B}$

$B \rightarrow I(A) : \{< Na, Nb >\}_{pk_A}$

$I \rightarrow A : \{< Na, Nb >\}_{pk_A}$

$A \rightarrow I : \{Nb\}_{pk_I}$

$I(A) \rightarrow B : \{Nb\}_{pk_B}$

Modèle formel : exemple d'utilisation du système de déduction

- Attaque de Lowe sur NSPK :

$$A \to I : \{< Na, A >\}_{pk_I}$$

$$I(A) \to B : \{< Na, A >\}_{pk_B}$$

$$B \to I(A) : \{< Na, Nb >\}_{pk_A}$$

$$I \to A : \{< Na, Nb >\}_{pk_A}$$

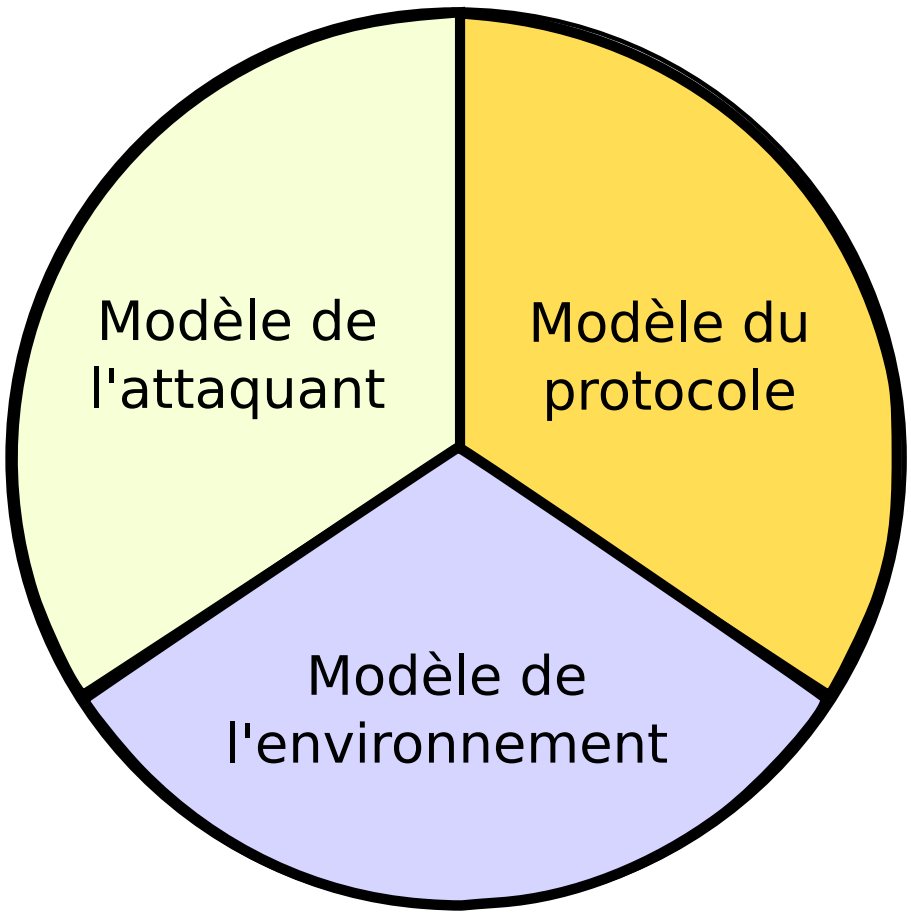
$$A \to I : \{Nb\}_{pk_I}$$

$$I(A) \to B : \{Nb\}_{pk_B}$$

$$\dfrac{\dfrac{aenc(< Na, A >,\ pk_I) \qquad sk(pk_I)}{< Na, A >}\ \text{adec} \qquad pk_B}{aenc(< Na, A >,\ pk_B)}\ \text{aenc}$$

Modèle formel : vers une automatisation des preuves

Un outil automatisé met en œuvre plusieurs modèles



Modèle formel : spécification d'un protocole

Protocole, rôle, action, terme

Structure :

- un protocole est un ensemble fini de rôles
- un rôle est une suite finie d'actions d'un participant légitime
- action = envoi ou réception d'un message
- un message est modélisé par des symboles : termes
- Terme :
  - nom (participant, clé, nonce),
  - variable,
  - cons*$(t_1, \cdot, t_n)$.

Structure présente dans les spécifications AVISPA, Scyther et ProVerif.

Modèle formel : spécification d'un protoc

Exemple spécifié en Spi Calcul pour ProVerif

$A \rightarrow B :< A, \{K\}_{pk(B)} >$
$B \rightarrow A :< B, \{K\}_{pk(A)} >$

```
(* Authentication queries *)

event beginAparam(pkey).
event endAparam(pkey).
event beginBparam(pkey).
event endBparam(pkey).

query x: pkey; inj-event(endAparam(x)) ==> inj-event(beginAparam(x)).
query x: pkey; inj-event(endBparam(x)) ==> inj-event(beginBparam(x)).

(* Secrecy queries *)

free secretAK, secretBK: bitstring [private].

query attacker(secretAK);
      attacker(secretBK).
(* Role Alice  *)

let PA(pkB: pkey, skA: skey) =
      in(c, pkX: pkey);
      event beginBparam(pkX);
      new K: bitstring;
      out(c, (pk(skA), aenc(K, pkX)));
      in(c, m: bitstring);
      let (=pkX, EKA: bitstring) = m in
          let (=K)=adec(EKA, skA) in
              if pkX=pkB then event endAparam(pk(skA));
      out(c, senc(secretAK,bitstring_to_key(K))).

(* Role Bob *)

let PB(pkA: pkey, skB: skey) =
      in(c, pkY: pkey);
      in(c, m2: bitstring);
      let (=pkY, EKB: bitstring) = m2 in
          let KY=adec(EKB, skB) in
              event beginAparam(pkY);
      out(c, (pk(skB), aenc(KY, pkY)));
      if pkY = pkA then event endBparam(pk(skB));
      out(c,senc(secretBK,bitstring_to_key(KY))).

(* Main *)

process
      new skA: skey; let pkA = pk(skA) in out(c, pkA);
      new skB: skey; let pkB = pk(skB) in out(c, pkB);
      ((!PA(pkB, skA)) | (!PB(pkA, skB)))
```

réception

envoi

--:--- PtDj_ex_5.pv   Bot L52   (ProVerif Typed Pi)----------

Modèle formel : modélisation de protocoles

Version en Spi Calcul simplifiée :

$PA(\mathrm{A}, \mathrm{B}) \triangleq \mathbf{out}(\mathrm{c}, (\mathrm{A}, \mathrm{aenc}(K, pkB))).\ \mathbf{in}(\mathrm{c}, (X, \mathrm{aenc}(KX, pkA))).0$

$PB(\mathrm{B}, \mathrm{A}) \triangleq \mathbf{in}(\mathrm{c}, (Y, \mathrm{aenc}(KY, pkB))).\ \mathbf{out}(\mathrm{c}, (\mathrm{B}, \mathrm{aenc}(KY, pkY))).0$

$Protocol \triangleq PA(A, B) \mid PB(B, A)$

Modèle formel : modélisation de protocoles

Version en Spi Calcul simplifiée :

$PA(\mathrm{A}, \mathrm{B}) \triangleq \mathbf{out}(\mathrm{c}, (\mathrm{A}, \mathrm{aenc}(K, pkB))).\ \mathbf{in}(\mathrm{c}, (X, \mathrm{aenc}(KX, pkA))).0$

$PB(\mathrm{B}, \mathrm{A}) \triangleq \mathbf{in}(\mathrm{c}, (Y, \mathrm{aenc}(KY, pkB))).\ \mathbf{out}(\mathrm{c}, (\mathrm{B}, \mathrm{aenc}(KY, pkY))).0$

$Protocol \triangleq PA(A, B) \mid PB(B, A)$

Transitions

$$
\begin{array}{lll}
Protocol & (1) \xrightarrow{\mathbf{out}(\mathrm{c},(\mathrm{A},\mathrm{aenc}(K,pkB)))} & \mathbf{in}(\mathrm{c}, (X, \mathrm{aenc}(KX, pkA))).0 \mid PB(B, A) \\
& (2) \xrightarrow{\mathbf{in}(\mathrm{c},(\mathrm{A},\mathrm{aenc}(K,pkB)))} & \mathbf{in}(\mathrm{c}, (X, \mathrm{aenc}(KX, pkA))).0 \mid \mathbf{out}(\mathrm{c}, (\mathrm{B}, \mathrm{aenc}(K, pkA))).0 \\
& (3) \xrightarrow{\mathbf{out}(\mathrm{c},(\mathrm{B},\mathrm{aenc}(K,pkA)))} & \mathbf{in}(\mathrm{c}, (X, \mathrm{aenc}(KX, pkA))).0 \mid 0 \\
& (4) \xrightarrow{\mathbf{in}(\mathrm{c},(B,\mathrm{aenc}(K,pkA)))} & 0
\end{array}
$$

Modèle formel : modélisation de protocoles

Version en Spi Calcul simplifiée :

$PA(\mathrm{A}, \mathrm{B}) \triangleq \mathbf{out}(\mathrm{c}, (\mathrm{A}, \mathrm{aenc}(K, pkB))).\ \mathbf{in}(\mathrm{c}, (X, \mathrm{aenc}(KX, pkA))).0$

$PB(\mathrm{B}, \mathrm{A}) \triangleq \mathbf{in}(\mathrm{c}, (Y, \mathrm{aenc}(KY, pkB))).\ \mathbf{out}(\mathrm{c}, (\mathrm{B}, \mathrm{aenc}(KY, pkY))).0$

$Protocol \triangleq PA(A, B) \mid PB(B, A)$

Transitions

$$
\begin{array}{lll}
Protocol & (1) \xrightarrow{\mathbf{out}(\mathrm{c},(\mathrm{A},\mathrm{aenc}(K,pkB)))} & \mathbf{in}(\mathrm{c}, (X, \mathrm{aenc}(KX, pkA))).0 \mid PB(B, A) \\
& (2) \xrightarrow{\mathbf{in}(\mathrm{c},(\mathrm{A},\mathrm{aenc}(K,pkB)))} & \mathbf{in}(\mathrm{c}, (X, \mathrm{aenc}(KX, pkA))).0 \mid \mathbf{out}(\mathrm{c}, (\mathrm{B}, \mathrm{aenc}(K, pkA))).0 \\
& (3) \xrightarrow{\mathbf{out}(\mathrm{c},(\mathrm{B},\mathrm{aenc}(K,pkA)))} & \mathbf{in}(\mathrm{c}, (X, \mathrm{aenc}(KX, pkA))).0 \mid 0 \\
& (4) \xrightarrow{\mathbf{in}(\mathrm{c},(B,\mathrm{aenc}(K,pkA)))} & 0
\end{array}
$$

L'ensemble des transitions $\{\tau_i\}_i$ tel que $Protocol \xrightarrow{\tau_1} \cdots \xrightarrow{\tau_n} 0$ forme une trace.

Parmi les traces, il ne faut pas négliger celles qui résultent de l'action de l'attaquant.

Modèle formel : modélisation de protocoles

Version en Spi Calcul simplifiée :

$PA(\mathrm{A}, \mathrm{B}) \triangleq \mathbf{out}(\mathrm{c}, (\mathrm{A}, \mathrm{aenc}(K, pkB))).\ \mathbf{in}(\mathrm{c}, (X, \mathrm{aenc}(KX, pkA))).0$

$PB(\mathrm{B}, \mathrm{A}) \triangleq \mathbf{in}(\mathrm{c}, (Y, \mathrm{aenc}(KY, pkB))).\ \mathbf{out}(\mathrm{c}, (\mathrm{B}, \mathrm{aenc}(KY, pkY))).0$

$Protocol \triangleq PA(A, B) \mid PB(B, A)$

Infinité de transitions

$Protocol \quad (1) \xrightarrow{\mathbf{out}(\mathrm{c},(\mathrm{A},\mathrm{aenc}(K,pkB)))} \mathbf{in}(\mathrm{c}, (X, \mathrm{aenc}(KX, pkA))).0 \mid PB(B, A)$

(2)

Modèle formel : modélisation de protocoles

Version en Spi Calcul simplifiée :

$PA(\mathrm{A}, \mathrm{B}) \triangleq \mathbf{out}(\mathrm{c}, (\mathrm{A}, \mathrm{aenc}(K, pkB))).\ \mathbf{in}(\mathrm{c}, (X, \mathrm{aenc}(KX, pkA))).0$

$PB(\mathrm{B}, \mathrm{A}) \triangleq \mathbf{in}(\mathrm{c}, (Y, \mathrm{aenc}(KY, pkB))).\ \mathbf{out}(\mathrm{c}, (\mathrm{B}, \mathrm{aenc}(KY, pkY))).0$

$Protocol \triangleq PA(A, B) \mid PB(B, A)$

Infinité de transitions

$Protocol \quad (1) \xrightarrow{\mathbf{out}(\mathrm{c},(\mathrm{A},\mathrm{aenc}(K,pkB)))} \mathbf{in}(\mathrm{c}, (X, \mathrm{aenc}(KX, pkA))).0 \mid PB(B, A)$

$(2) \xrightarrow{\mathbf{in}(\mathrm{c},(\mathrm{I},\mathrm{aenc}(K_I,pkB)))} \mathbf{in}(\mathrm{c}, (X, \mathrm{aenc}(KX, pkA))).0 \mid \mathbf{out}(\mathrm{c}, (\mathrm{B}, \mathrm{aenc}(K_I, pkI)$

Modèle formel : modélisation de protocoles

Version en Spi Calcul simplifiée :

$PA(\mathrm{A}, \mathrm{B}) \triangleq \mathbf{out}(\mathrm{c}, (\mathrm{A}, \mathrm{aenc}(K, pkB))).\ \mathbf{in}(\mathrm{c}, (X, \mathrm{aenc}(KX, pkA))).0$

$PB(\mathrm{B}, \mathrm{A}) \triangleq \mathbf{in}(\mathrm{c}, (Y, \mathrm{aenc}(KY, pkB))).\ \mathbf{out}(\mathrm{c}, (\mathrm{B}, \mathrm{aenc}(KY, pkY))).0$

$Protocol \triangleq PA(A, B) \mid PB(B, A)$

Infinité de transitions

$$Protocol \quad (1) \xrightarrow{\mathbf{out}(\mathrm{c},(\mathrm{A},\mathrm{aenc}(K,pkB)))} \mathbf{in}(\mathrm{c}, (X, \mathrm{aenc}(KX, pkA))).0 \mid PB(B, A)$$

$$(2) \xrightarrow{\mathbf{in}(\mathrm{c},(\mathrm{I},\mathrm{aenc}(K_I,pkB)))} \mathbf{in}(\mathrm{c}, (X, \mathrm{aenc}(KX, pkA))).0 \mid \mathbf{out}(\mathrm{c}, (\mathrm{B}, \mathrm{aenc}(K_I, pkI)$$

$$(2') \xrightarrow{\mathbf{in}(\mathrm{c},(\mathrm{I},\mathrm{aenc}(<K_I,K_I>,pkB)))} \mathbf{in}(\mathrm{c}, (X, \mathrm{aenc}(KX, pkA))).0 \mid \mathbf{out}(\mathrm{c}, (\mathrm{B}, \mathrm{aenc}(< K_I, K$$

Modèle formel : modélisation de protocoles

Version en Spi Calcul simplifiée :

$PA(\mathrm{A},\mathrm{B}) \triangleq \mathbf{out}(\mathrm{c},(\mathrm{A},\mathrm{aenc}(K,pkB))).\ \mathbf{in}(\mathrm{c},(X,\mathrm{aenc}(KX,pkA))).0$

$PB(\mathrm{B},\mathrm{A}) \triangleq \mathbf{in}(\mathrm{c},(Y,\mathrm{aenc}(KY,pkB))).\ \mathbf{out}(\mathrm{c},(\mathrm{B},\mathrm{aenc}(KY,pkY))).0$

$Protocol \triangleq PA(A,B) \mid PB(B,A)$

Infinité de transitions

$$
\begin{array}{lll}
Protocol & (1) \xrightarrow{\mathbf{out}(\mathrm{c},(\mathrm{A},\mathrm{aenc}(K,pkB)))} & \mathbf{in}(\mathrm{c},(X,\mathrm{aenc}(KX,pkA))).0 \mid PB(B,A) \\
& (2) \xrightarrow{\mathbf{in}(\mathrm{c},(\mathrm{I},\mathrm{aenc}(K_I,pkB)))} & \mathbf{in}(\mathrm{c},(X,\mathrm{aenc}(KX,pkA))).0 \mid \mathbf{out}(\mathrm{c},(\mathrm{B},\mathrm{aenc}(K_I,pkI)) \\
& (2') \xrightarrow{\mathbf{in}(\mathrm{c},(\mathrm{I},\mathrm{aenc}(<K_I,K_I>,pkB)))} & \mathbf{in}(\mathrm{c},(X,\mathrm{aenc}(KX,pkA))).0 \mid \mathbf{out}(\mathrm{c},(\mathrm{B},\mathrm{aenc}(<K_I,K \\
& (2'') \xrightarrow{\mathbf{in}(\mathrm{c},(\mathrm{I},\mathrm{aenc}(<K_I,aenc(K_I,pK_I)>,pkB)))} & \mathbf{in}(\mathrm{c},(X,\mathrm{aenc}(KX,pkA))).0 \mid \mathbf{out}(\mathrm{c},(\mathrm{B},\mathrm{aenc}(<K_I,a
\end{array}
$$

Modèle formel : modélisation de protocoles

Version en Spi Calcul simplifiée :

$PA(\mathrm{A}, \mathrm{B}) \triangleq \mathbf{out}(\mathrm{c}, (\mathrm{A}, \mathrm{aenc}(K, pkB))).\ \mathbf{in}(\mathrm{c}, (X, \mathrm{aenc}(KX, pkA))).0$

$PB(\mathrm{B}, \mathrm{A}) \triangleq \mathbf{in}(\mathrm{c}, (Y, \mathrm{aenc}(KY, pkB))).\ \mathbf{out}(\mathrm{c}, (\mathrm{B}, \mathrm{aenc}(KY, pkY))).0$

$Protocol \triangleq PA(A, B) \mid PB(B, A)$

Infinité de transitions

$$Protocol \quad (1) \xrightarrow{\mathbf{out}(\mathrm{c},(\mathrm{A},\mathrm{aenc}(K,pkB)))} \mathbf{in}(\mathrm{c}, (X, \mathrm{aenc}(KX, pkA))).0 \mid PB(B, A)$$

$$(2) \xrightarrow{\mathbf{in}(\mathrm{c},(\mathrm{I},\mathrm{aenc}(K_I,pkB)))} \mathbf{in}(\mathrm{c}, (X, \mathrm{aenc}(KX, pkA))).0 \mid \mathbf{out}(\mathrm{c}, (\mathrm{B}, \mathrm{aenc}(K_I, pkI)$$

$$(2') \xrightarrow{\mathbf{in}(\mathrm{c},(\mathrm{I},\mathrm{aenc}(<K_I,K_I>,pkB)))} \mathbf{in}(\mathrm{c}, (X, \mathrm{aenc}(KX, pkA))).0 \mid \mathbf{out}(\mathrm{c}, (\mathrm{B}, \mathrm{aenc}(< K_I, K$$

$$(2'') \xrightarrow{\mathbf{in}(\mathrm{c},(\mathrm{I},\mathrm{aenc}(<K_I,aenc(K_I,pK_I)>,pkB)))} \mathbf{in}(\mathrm{c}, (X, \mathrm{aenc}(KX, pkA))).0 \mid \mathbf{out}(\mathrm{c}, (\mathrm{B}, \mathrm{aenc}(< K_I, a$$

Taille non bornée des termes $\Rightarrow$ nombre infini de transitions possibles

Modèle formel : modélisation de protocoles

Version en Spi Calcul simplifiée :

$PA(\mathrm{A},\mathrm{B}) \triangleq \mathbf{out}(\mathrm{c},(\mathrm{A},\mathrm{aenc}(K,pkB))).\ \mathbf{in}(\mathrm{c},(X,\mathrm{aenc}(KX,pkA))).0$

$PB(\mathrm{B},\mathrm{A}) \triangleq \mathbf{in}(\mathrm{c},(Y,\mathrm{aenc}(KY,pkB))).\ \mathbf{out}(\mathrm{c},(\mathrm{B},\mathrm{aenc}(KY,pkY))).0$

$Protocol \triangleq PA(A,B) \mid PB(B,A)$

Infinité de transitions

$Protocol \quad (1) \xrightarrow{\mathbf{out}(\mathrm{c},(\mathrm{A},\mathrm{aenc}(K,pkB)))} \mathbf{in}(\mathrm{c},(X,\mathrm{aenc}(KX,pkA))).0 \mid PB(B,A)$

$(2) \xrightarrow{\mathbf{in}(\mathrm{c},(\mathrm{I},\mathrm{aenc}(K_I,pkB)))} \mathbf{in}(\mathrm{c},(X,\mathrm{aenc}(KX,pkA))).0 \mid \mathbf{out}(\mathrm{c},(\mathrm{B},\mathrm{aenc}(K_I,pkI)$

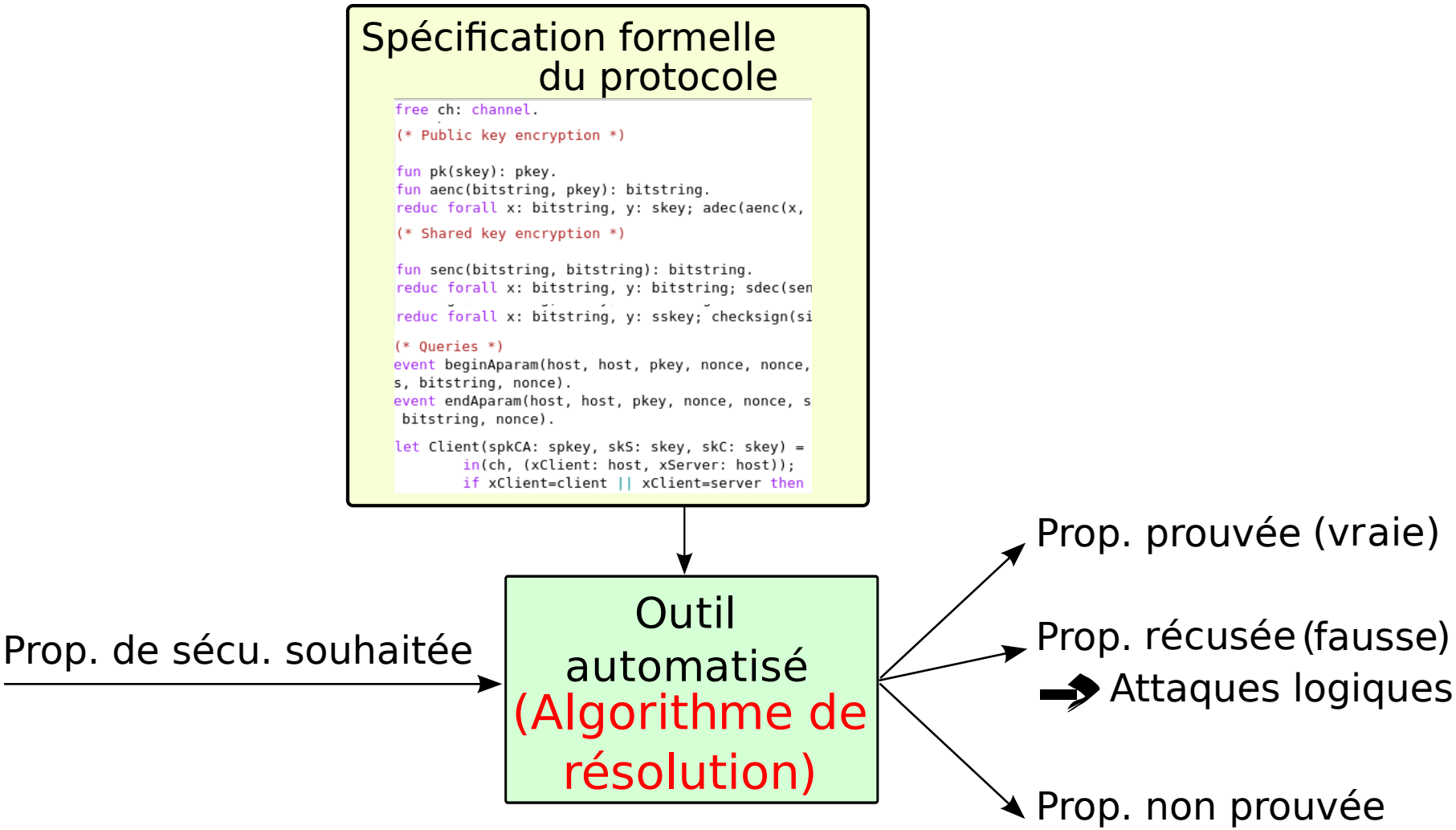
$(2') \xrightarrow{\mathbf{in}(\mathrm{c},(\mathrm{I},\mathrm{aenc}(<K_I,K_I>,pkB)))} \mathbf{in}(\mathrm{c},(X,\mathrm{aenc}(KX,pkA))).0 \mid \mathbf{out}(\mathrm{c},(\mathrm{B},\mathrm{aenc}(<K_I,K$

$(2'') \xrightarrow{\mathbf{in}(\mathrm{c},(\mathrm{I},\mathrm{aenc}(<K_I,aenc(K_I,pK_I)>,pkB)))} \mathbf{in}(\mathrm{c},(X,\mathrm{aenc}(KX,pkA))).0 \mid \mathbf{out}(\mathrm{c},(\mathrm{B},\mathrm{aenc}(<K_I,a$

Taille non bornée des termes $\Rightarrow$ nombre infini de transitions possibles

Idem nombre de sessions et nombre de participants non bornés

Modèle formel : comment établir une preuve ?



Modèle formel : indécidabilité et approches de résolution

Indécidabilité

La vérification d'un protocole cryptographique est un problème indécidable, car :

1. le nombre de sessions est non borné,
2. le nombre de participants est non borné,
3. la taille des termes est non bornée

Modèle formel : indécidabilité et approches de résolution

Indécidabilité

La vérification d'un protocole cryptographique est un problème indécidable, car :

1. le nombre de sessions est non borné,
2. le nombre de participants est non borné,
3. la taille des termes est non bornée

Plusieurs approches de résolution :

1. Borner le nombre de sessions $\rightarrow$ le problème devient NP-complet
   Mais la preuve n'est valide que pour le nombre de sessions considéré

Modèle formel : indécidabilité et approches de résolution

Indécidabilité

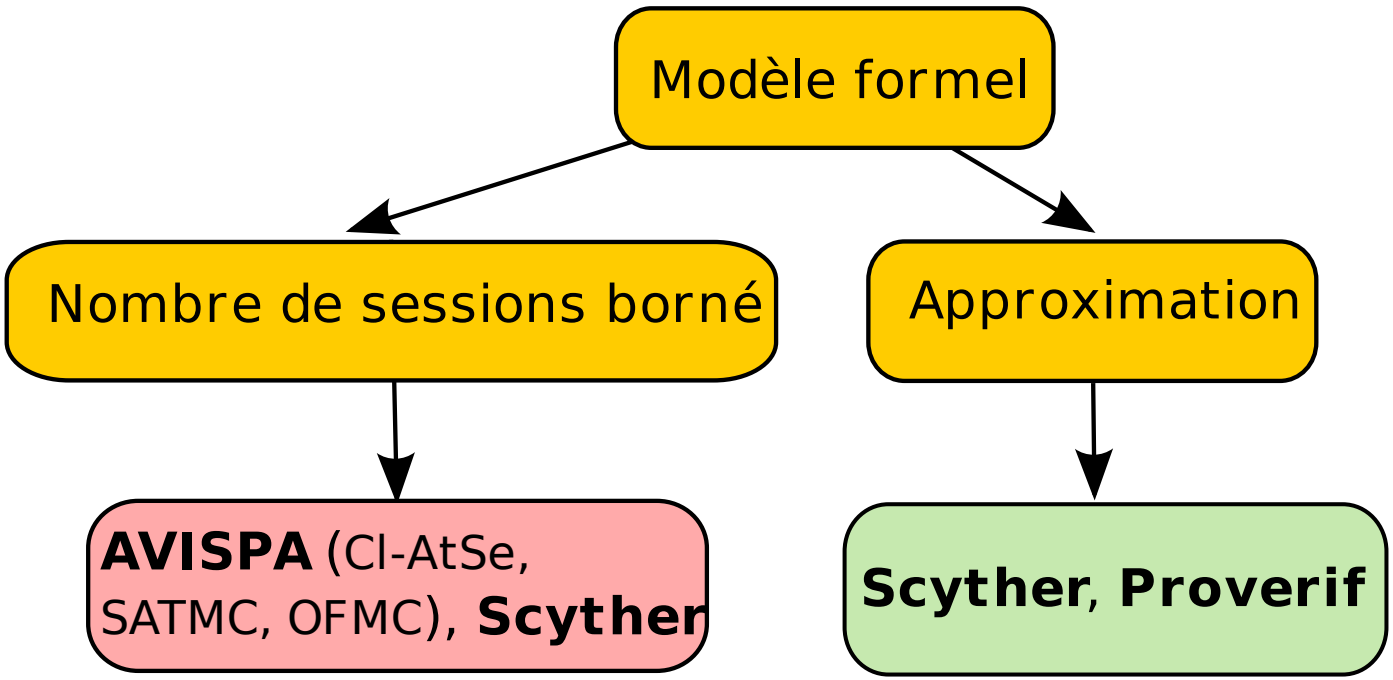
La vérification d'un protocole cryptographique est un problème indécidable, car :

1. le nombre de sessions est non borné,
2. le nombre de participants est non borné,
3. la taille des termes est non bornée

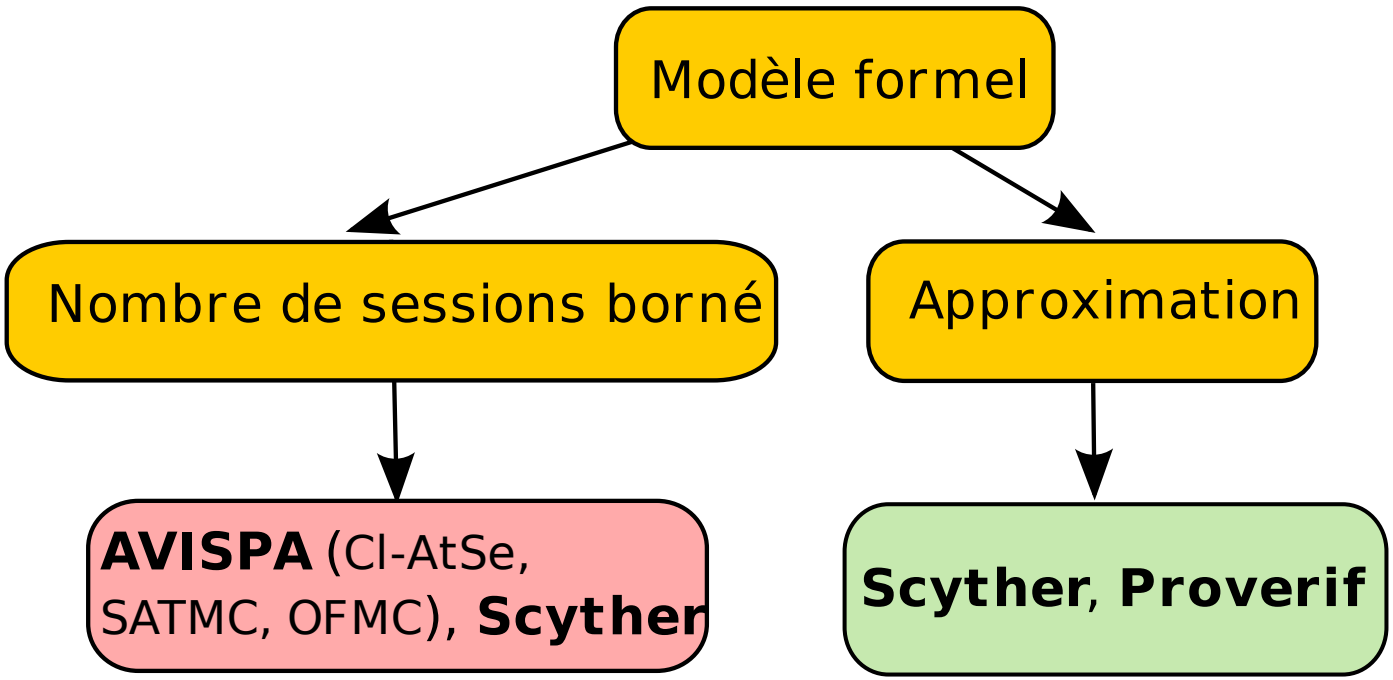
Plusieurs approches de résolution :

1. Borner le nombre de sessions $\rightarrow$ le problème devient NP-complet
   Mais la preuve n'est valide que pour le nombre de sessions considéré
2. Utiliser des approximations $\rightarrow$ le problème reste indécidable :
   - soit on prouve la propriété,
   - soit on trouve une attaque,
   - soit la propriété ne peut pas être prouvée

Modèle formel : approches utilisées par les outils

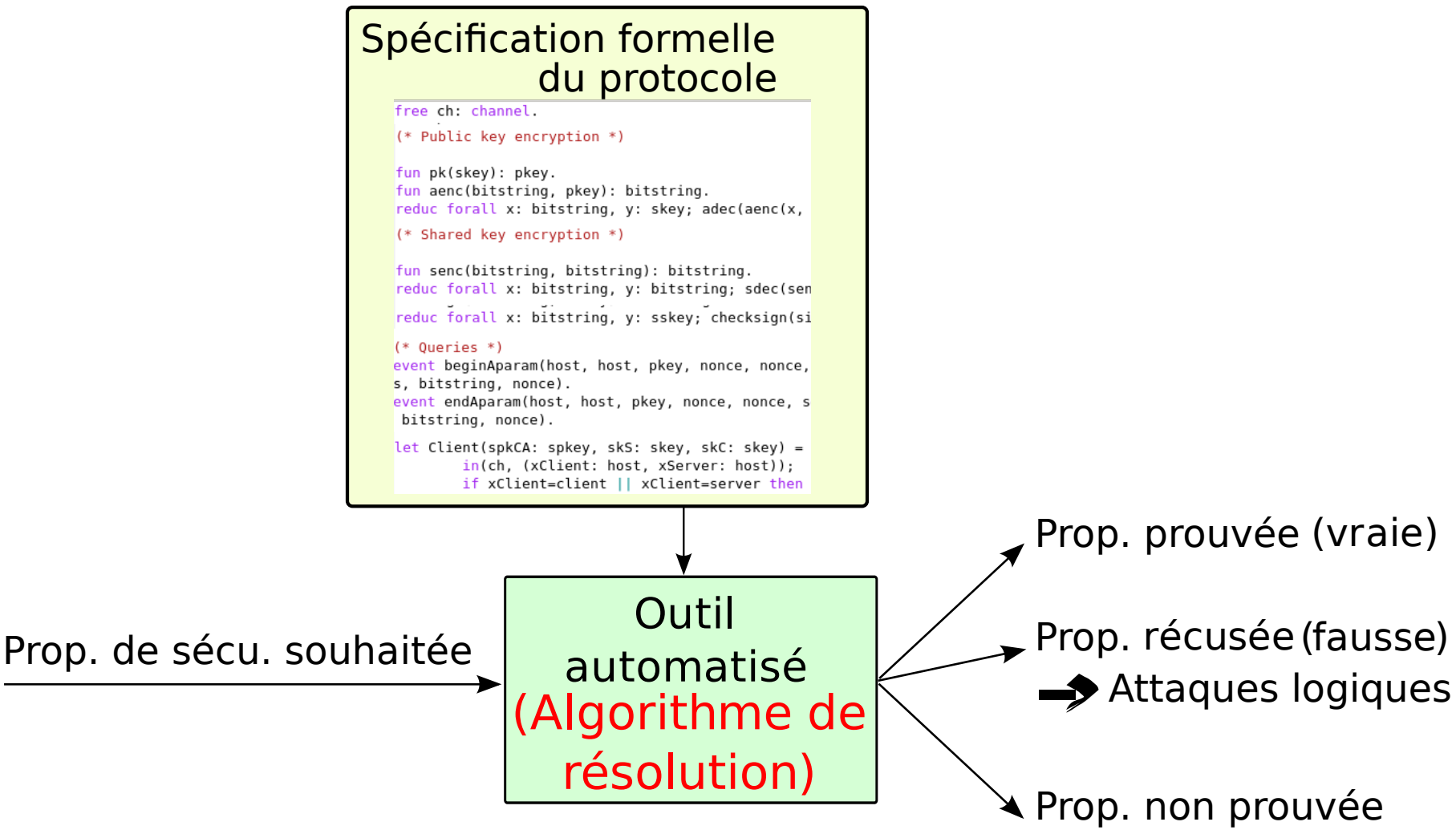


Modèle formel : approches utilisées par les outils



⇒ Nécessité de croiser les vérificateurs utilisés pour donner un avis précis

Modèle formel : résumé



Plan

Propriétés de sécurité : généralités

- Dire qu'un protocole satisfait une propriété de sécurité, c'est énoncer un *théorème*.
- Si une propriété ne peut pas être prouvée, on peut parfois trouver un *contre-exemple* = attaque logique

Propriétés de sécurité : le secret

Secret

On dit que le protocole P ne préserve pas le secret K si :
$\exists \{\alpha_i\}_i,\ P \xrightarrow{\alpha_1} \cdots \xrightarrow{\alpha_n} 0 : \alpha_1 \cdots \alpha_n \vdash K$
où $\{\alpha_i\}_i$ sont les sorties du protocole lors des transitions.

Propriétés de sécurité : le secret

Secret

On dit que le protocole P ne préserve pas le secret K si :
$\exists \{\alpha_i\}_i,\ P \xrightarrow{\alpha_1} \cdots \xrightarrow{\alpha_n} 0 : \alpha_1 \cdots \alpha_n \vdash K$
où $\{\alpha_i\}_i$ sont les sorties du protocole lors des transitions.

Intuitivement :

- un protocole préserve le secret de K si un adversaire ne peut jamais obtenir K en le construisant à partir des sorties du protocole.

Propriétés de sécurité : le secret

Secret

On dit que le protocole P ne préserve pas le secret K si :
$\exists \{\alpha_i\}_i,\ P \xrightarrow{\alpha_1} \cdots \xrightarrow{\alpha_n} 0 : \alpha_1 \cdots \alpha_n \vdash K$
où $\{\alpha_i\}_i$ sont les sorties du protocole lors des transitions.

Intuitivement :

- un protocole préserve le secret de K si un adversaire ne peut jamais obtenir K en le construisant à partir des sorties du protocole.

Dans la spécification, on écrit :

- dans ProVerif : query attacker(K).
- dans AVISPA : secret(K,sec_K,{A,B})
- dans Scyther : claim_I1(I,Secret,k);

Propriétés de sécurité : l'authentification, plusieurs nuances

Authentification

« L'authentification a pour but de vérifier l'identité dont se réclame une personne ou une machine » RGS, ANSSI

Propriétés de sécurité : l'authentification, plusieurs nuances

Authentification

« L'authentification a pour but de vérifier l'identité dont se réclame une personne ou une machine » RGS, ANSSI

Hiérarchie de Gavin Lowe

De la plus faible à la plus forte :

- vitalité
- accord faible
- accord (sur une liste de termes S)

Propriétés d'authentification : définitions

Vitalité

Si un participant A exécute le protocole en pensant qu'il l'effectuait avec un participant B, alors B a effectivement réalisé une action.

Propriétés d'authentification : définitions

Vitalité

Si un participant A exécute le protocole en pensant qu'il l'effectuait avec un participant B, alors B a effectivement réalisé une action.

Accord faible

En plus de la vitalité, B admet qu'il est en train de communiquer avec A.

Propriétés d'authentification : définitions

Vitalité

Si un participant A exécute le protocole en pensant qu'il l'effectuait avec un participant B, alors B a effectivement réalisé une action.

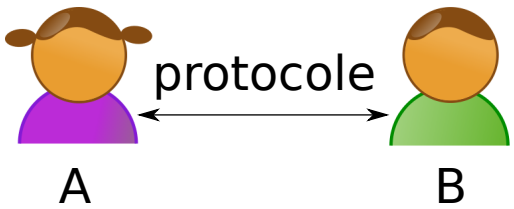
Accord faible

En plus de la vitalité, B admet qu'il est en train de communiquer avec A.

Accord (sur une liste de termes S)

En plus de l'accord faible, A et B sont d'accord sur les valeurs incluses dans S.

Propriétés d'authentification : en pratique



Propriétés d'authentification : en pratique

[Docs] [txt|pdf] [draft-ietf-tls-rf...] [Diff1] [Diff2] [IPR] [Errata]

Updated by: 5746, 5878, 6176 — PROPOSED STANDARD — Errata Exist

Network Working Group — T. Dierks
Request for Comments: 5246 — Independent
Obsoletes: 3268, 4346, 4366 — E. Rescorla
Updates: 4492 — RTFM, Inc.
Category: Standards Track — August 2008

The Transport Layer Security (TLS) Protocol Version 1.2

Status of This Memo

This document specifies an Internet standards track protocol for the Internet community, and requests discussion and suggestions for improvements. Please refer to the current edition of the "Internet Official Protocol Standards" (STD 1) for the standardization state and status of this protocol. Distribution of this memo is unlimited.

Abstract

This document specifies Version 1.2 of the Transport Layer Security (TLS) protocol. The TLS protocol provides communications security over the Internet. The protocol allows client/server applications to communicate in a way that is designed to prevent eavesdropping, tampering, or message forgery.

Table of Contents

1. Introduction 4
   1.1. Requirements Terminology 5
   1.2. Major Differences from TLS 1.1 5
2. Goals 6
3. Goals of This Document 7
4. Presentation Language 7

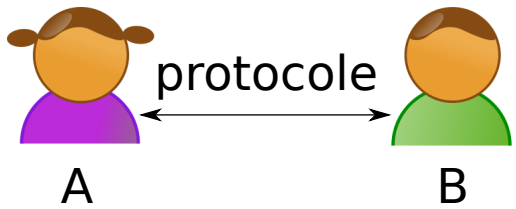
Spécification en langage naturel

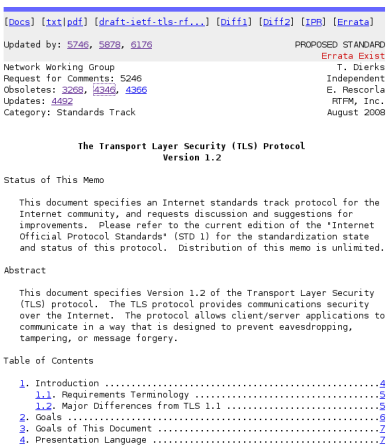
objectifs de sécurité

```
free ch: channel.

(* Public key encryption *)

fun pk(skey): pkey.
fun aenc(bitstring, pkey): bitstring.
reduc forall x: bitstring, y: skey; adec(aenc(x,

(* Shared key encryption *)

fun senc(bitstring, bitstring): bitstring.
reduc forall x: bitstring, y: bitstring; sdec(sen

reduc forall x: bitstring, y: sskey; checksign(si

(* Queries *)
event beginAparam(host, host, pkey, nonce, nonce,
s, bitstring, nonce).
event endAparam(host, host, pkey, nonce, nonce, s
 bitstring, nonce).

let Client(spkCA: spkey, skS: skey, skC: skey) =
        in(ch, (xClient: host, xServer: host));
        if xClient=client || xClient=server then
```

Spécification formelle

Propriétés d'authentification : en pratique





[Docs] [txt|pdf] [draft-ietf-tls-rf...] [Diff1] [Diff2] [IPR] [Errata]

Updated by: 5746, 5878, 6176 — PROPOSED STANDARD — Errata Exist

Network Working Group — T. Dierks
Request for Comments: 5246 — Independent
Obsoletes: 3268, 4346, 4366 — E. Rescorla
Updates: 4492 — RTFM, Inc.
Category: Standards Track — August 2008

The Transport Layer Security (TLS) Protocol Version 1.2

Status of This Memo

This document specifies an Internet standards track protocol for the Internet community, and requests discussion and suggestions for improvements. Please refer to the current edition of the "Internet Official Protocol Standards" (STD 1) for the standardization state and status of this protocol. Distribution of this memo is unlimited.

Abstract

This document specifies Version 1.2 of the Transport Layer Security (TLS) protocol. The TLS protocol provides communications security over the Internet. The protocol allows client/server applications to communicate in a way that is designed to prevent eavesdropping, tampering, or message forgery.

Table of Contents

1. Introduction 4
1.1. Requirements Terminology 5
1.2. Major Differences from TLS 1.1 5
2. Goals 6
3. Goals of This Document 7
4. Presentation Language 7

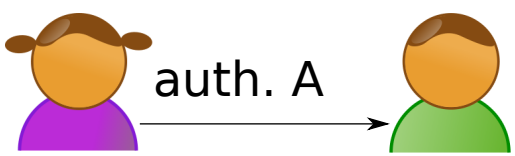
Spécification en langage naturel

objectifs de sécurité

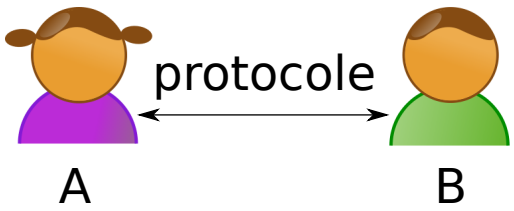
```
free ch: channel.
(* Public key encryption *)

fun pk(skey): pkey.
fun aenc(bitstring, pkey): bitstring.
reduc forall x: bitstring, y: skey; adec(aenc(x,
(* Shared key encryption *)

fun senc(bitstring, bitstring): bitstring.
reduc forall x: bitstring, y: bitstring; sdec(sen
reduc forall x: bitstring, y: sskey; checksign(si
(* Queries *)
event beginAparam(host, host, pkey, nonce, nonce,
s, bitstring, nonce).
event endAparam(host, host, pkey, nonce, nonce, s
 bitstring, nonce).
let Client(spkCA: spkey, skS: skey, skC: skey) =
        in(ch, (xClient: host, xServer: host));
        if xClient=client || xClient=server then
```

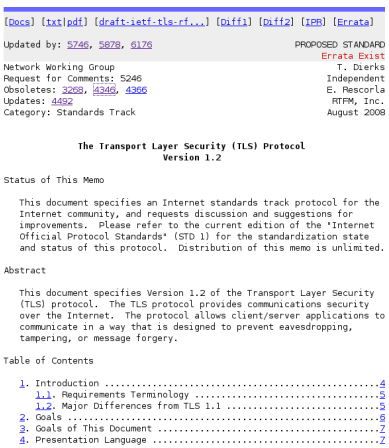
Spécification formelle

- Au plus 2 authentifications à vérifier :



Propriétés d'authentification : en pratique





[Docs] [txt|pdf] [draft-ietf-tls-rf...] [Diff1] [Diff2] [IPR] [Errata]

Updated by: 5746, 5878, 6176 — PROPOSED STANDARD — Errata Exist

Network Working Group — T. Dierks
Request for Comments: 5246 — Independent
Obsoletes: 3268, 4346, 4366 — E. Rescorla
Updates: 4492 — RTFM, Inc.
Category: Standards Track — August 2008

The Transport Layer Security (TLS) Protocol Version 1.2

Status of This Memo

This document specifies an Internet standards track protocol for the Internet community, and requests discussion and suggestions for improvements. Please refer to the current edition of the "Internet Official Protocol Standards" (STD 1) for the standardization state and status of this protocol. Distribution of this memo is unlimited.

Abstract

This document specifies Version 1.2 of the Transport Layer Security (TLS) protocol. The TLS protocol provides communications security over the Internet. The protocol allows client/server applications to communicate in a way that is designed to prevent eavesdropping, tampering, or message forgery.

Table of Contents

1. Introduction 4
1.1. Requirements Terminology 5
1.2. Major Differences from TLS 1.1 5
2. Goals 6
3. Goals of This Document 7
4. Presentation Language 7

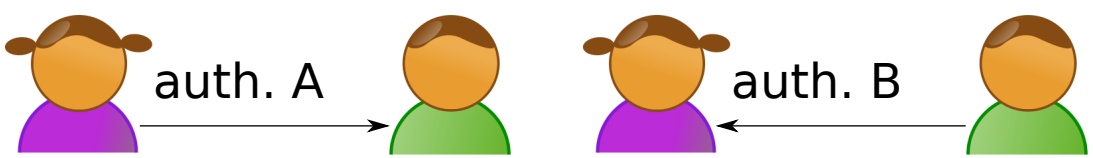
Spécification en langage naturel

objectifs de sécurité

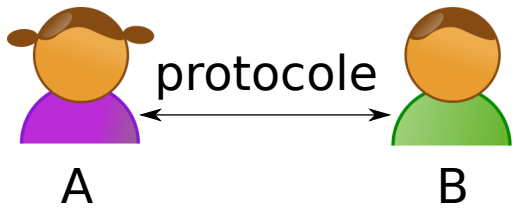
```
free ch: channel.
(* Public key encryption *)

fun pk(skey): pkey.
fun aenc(bitstring, pkey): bitstring.
reduc forall x: bitstring, y: skey; adec(aenc(x,
(* Shared key encryption *)

fun senc(bitstring, bitstring): bitstring.
reduc forall x: bitstring, y: bitstring; sdec(sen
reduc forall x: bitstring, y: sskey; checksign(si
(* Queries *)
event beginAparam(host, host, pkey, nonce, nonce,
s, bitstring, nonce).
event endAparam(host, host, pkey, nonce, nonce, s
 bitstring, nonce).
let Client(spkCA: spkey, skS: skey, skC: skey) =
        in(ch, (xClient: host, xServer: host));
        if xClient=client || xClient=server then
```

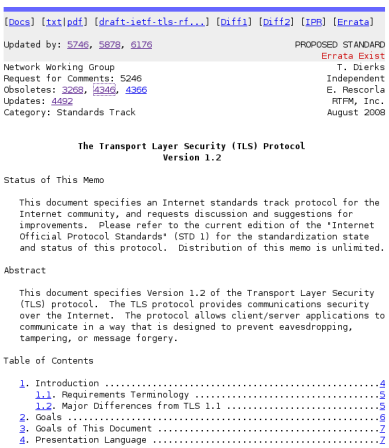
Spécification formelle

- Au plus 2 authentifications à vérifier :



Propriétés d'authentification : en pratique





[Docs] [txt|pdf] [draft-ietf-tls-rf...] [Diff1] [Diff2] [IPR] [Errata]

Updated by: 5746, 5878, 6176 — PROPOSED STANDARD — Errata Exist

Network Working Group — T. Dierks
Request for Comments: 5246 — Independent
Obsoletes: 3268, 4346, 4366 — E. Rescorla
Updates: 4492 — RTFM, Inc.
Category: Standards Track — August 2008

The Transport Layer Security (TLS) Protocol Version 1.2

Status of This Memo

This document specifies an Internet standards track protocol for the Internet community, and requests discussion and suggestions for improvements. Please refer to the current edition of the "Internet Official Protocol Standards" (STD 1) for the standardization state and status of this protocol. Distribution of this memo is unlimited.

Abstract

This document specifies Version 1.2 of the Transport Layer Security (TLS) protocol. The TLS protocol provides communications security over the Internet. The protocol allows client/server applications to communicate in a way that is designed to prevent eavesdropping, tampering, or message forgery.

Table of Contents

1. Introduction 4
1.1. Requirements Terminology 5
1.2. Major Differences from TLS 1.1 5
2. Goals 6
3. Goals of This Document 7
4. Presentation Language 7

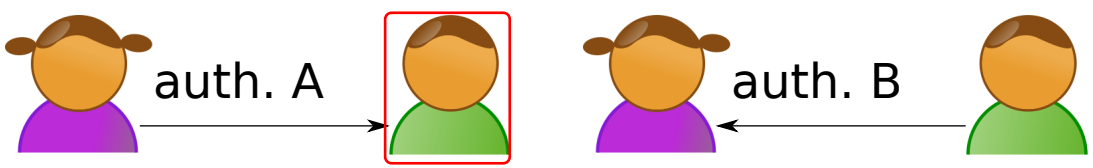
Spécification en langage naturel

objectifs de sécurité

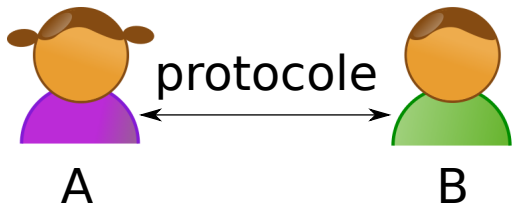
```
free ch: channel.
(* Public key encryption *)

fun pk(skey): pkey.
fun aenc(bitstring, pkey): bitstring.
reduc forall x: bitstring, y: skey; adec(aenc(x,
(* Shared key encryption *)

fun senc(bitstring, bitstring): bitstring.
reduc forall x: bitstring, y: bitstring; sdec(sen
reduc forall x: bitstring, y: sskey; checksign(si
(* Queries *)
event beginAparam(host, host, pkey, nonce, nonce,
s, bitstring, nonce).
event endAparam(host, host, pkey, nonce, nonce, s
 bitstring, nonce).
let Client(spkCA: spkey, skS: skey, skC: skey) =
        in(ch, (xClient: host, xServer: host));
        if xClient=client || xClient=server then
```

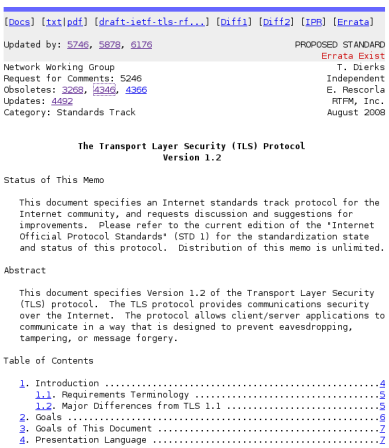
Spécification formelle

- Au plus 2 authentifications à vérifier :



Propriétés d'authentification : en pratique





[Docs] [txt|pdf] [draft-ietf-tls-rf...] [Diff1] [Diff2] [IPR] [Errata]

Updated by: 5746, 5878, 6176 — PROPOSED STANDARD — Errata Exist

Network Working Group — T. Dierks
Request for Comments: 5246 — Independent
Obsoletes: 3268, 4346, 4366 — E. Rescorla
Updates: 4492 — RTFM, Inc.
Category: Standards Track — August 2008

The Transport Layer Security (TLS) Protocol Version 1.2

Status of This Memo

This document specifies an Internet standards track protocol for the Internet community, and requests discussion and suggestions for improvements. Please refer to the current edition of the "Internet Official Protocol Standards" (STD 1) for the standardization state and status of this protocol. Distribution of this memo is unlimited.

Abstract

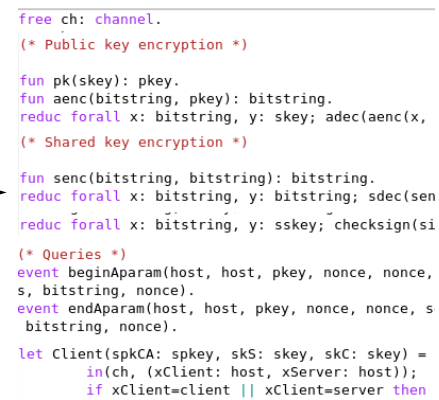
This document specifies Version 1.2 of the Transport Layer Security (TLS) protocol. The TLS protocol provides communications security over the Internet. The protocol allows client/server applications to communicate in a way that is designed to prevent eavesdropping, tampering, or message forgery.

Table of Contents

1. Introduction 4
1.1. Requirements Terminology 5
1.2. Major Differences from TLS 1.1 5
2. Goals 6
3. Goals of This Document 7
4. Presentation Language 7

Spécification en langage naturel

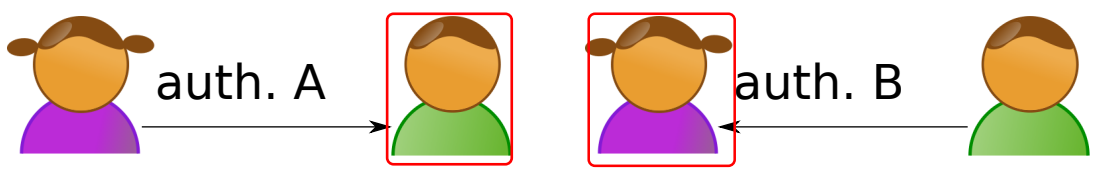
objectifs de sécurité



```
free ch: channel.
(* Public key encryption *)

fun pk(skey): pkey.
fun aenc(bitstring, pkey): bitstring.
reduc forall x: bitstring, y: skey; adec(aenc(x,
(* Shared key encryption *)

fun senc(bitstring, bitstring): bitstring.
reduc forall x: bitstring, y: bitstring; sdec(sen
reduc forall x: bitstring, y: sskey; checksign(si
(* Queries *)
event beginAparam(host, host, pkey, nonce, nonce,
s, bitstring, nonce).
event endAparam(host, host, pkey, nonce, nonce, s
 bitstring, nonce).
let Client(spkCA: spkey, skS: skey, skC: skey) =
        in(ch, (xClient: host, xServer: host));
        if xClient=client || xClient=server then
```

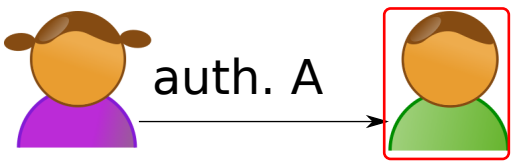
Spécification formelle

- Au plus 2 authentifications à vérifier :



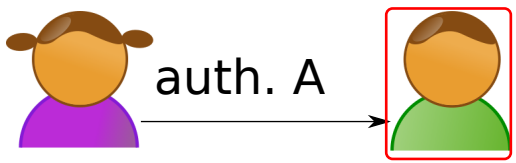
Propriétés d'authentification
Exemple 1 : Y a-t-il vitalité ?

$A \rightarrow B : \{Na,A\}pk(B)$
$B \rightarrow A : Na,Nb$
$A \rightarrow B : \{Nb\}pk(B)$



Propriétés d'authentification
Exemple 1 : Y a-t-il vitalité ?

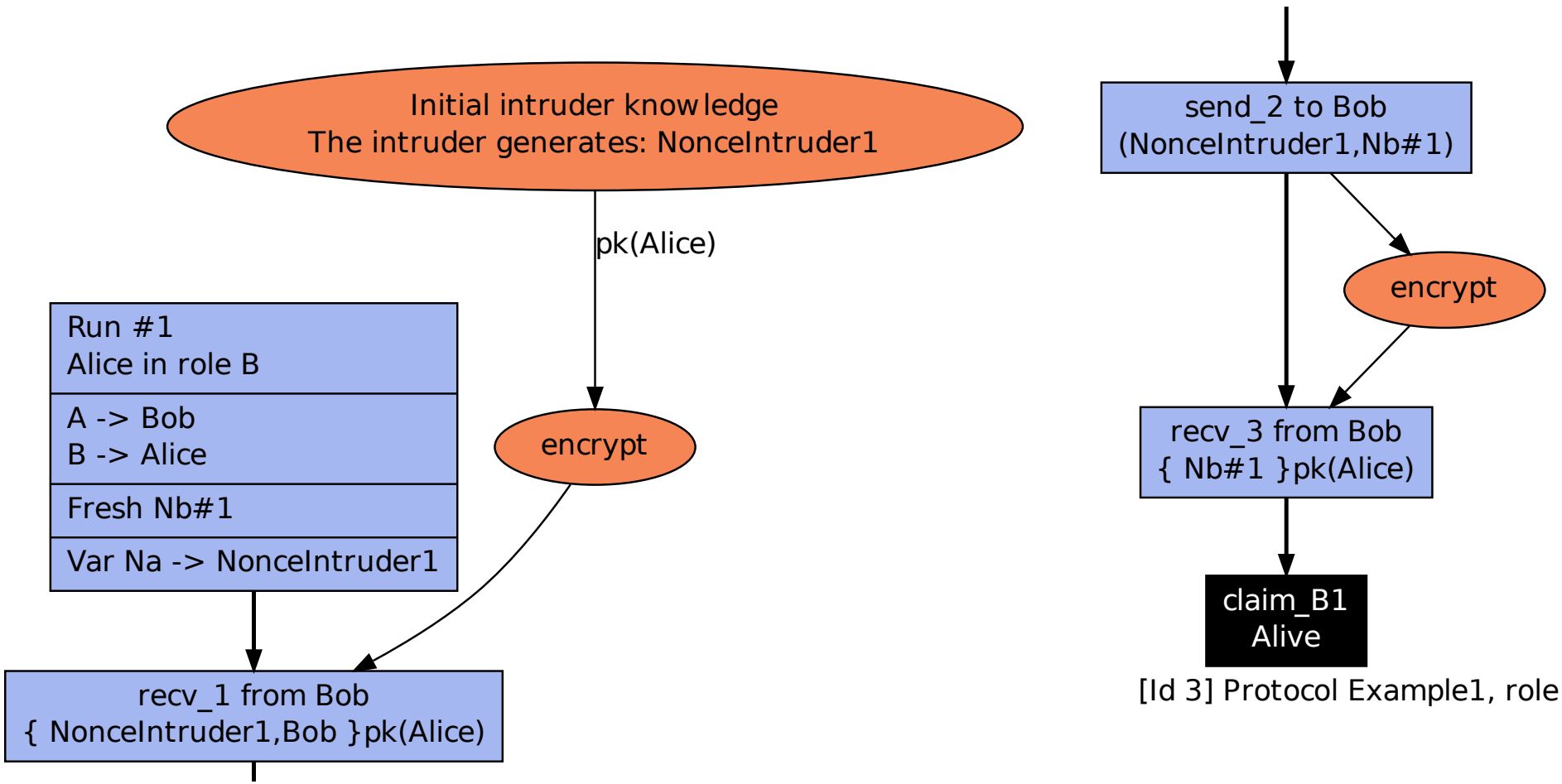
A → B : {Na,A}pk(B)
B → A : Na,Nb
A → B : {Nb}pk(B)



Scyther results : verify

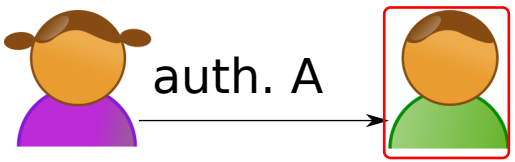
			Status		Comments	Patterns
B	Example1,B1	Alive	Fail	Falsified	At least 1 attack.	1 attack
	Example1,B2	Weakagree	Fail	Falsified	At least 1 attack.	1 attack
	Example1,B3	Niagree	Fail	Falsified	At least 1 attack.	1 attack

Propriétés d'authentification
Exemple 1 : absence de vitalité



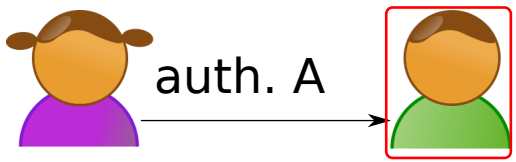
Propriétés d'authentification
Exemple 2 : Y a-t-il vitalité ?

$A \rightarrow B : \{Na,A\}pk(B)$
$B \rightarrow A : \{Na,Nb\}pk(A)$
$A \rightarrow B : \{Nb\}pk(B)$



Propriétés d'authentification
Exemple 2 : Y a-t-il vitalité ?

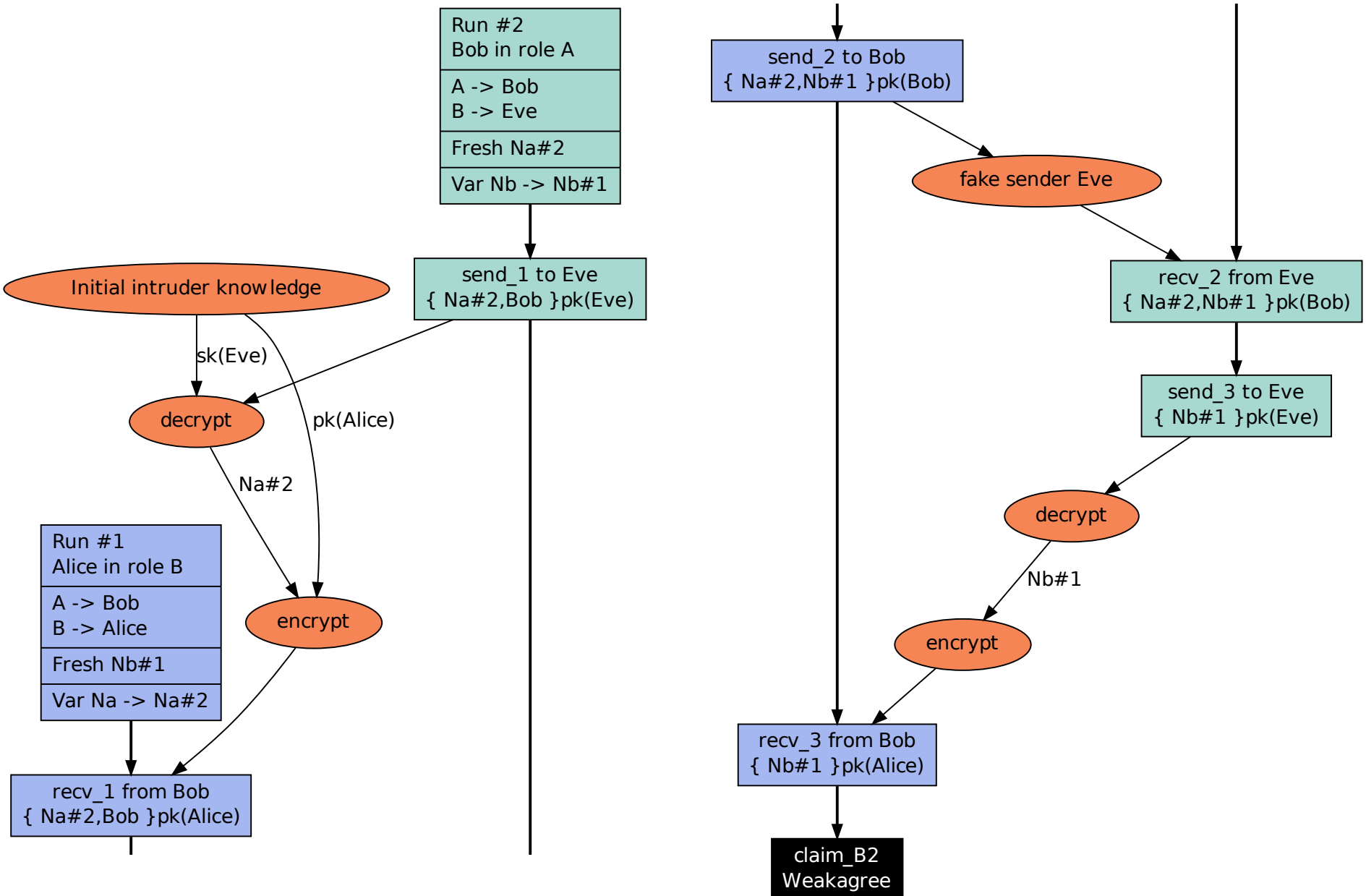
$A \rightarrow B : \{Na,A\}pk(B)$
$B \rightarrow A : \{Na,Nb\}pk(A)$
$A \rightarrow B : \{Nb\}pk(B)$



Scyther results : verify

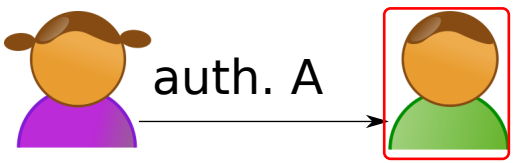
			Status		Comments	Patterns
B	Example2,B1	Alive	Ok	Verified	No attacks.	
	Example2,B2	Weakagree	Fail	Falsified	At least 1 attack.	1 attack
	Example2,B3	Niagree	Fail	Falsified	At least 1 attack.	1 attack

Propriétés d'authentification
Exemple 2 : vitalité oui, mais absence d'accord faible



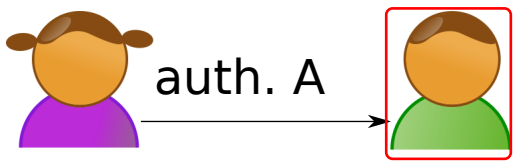
Propriétés d'authentification
Exemple 3 : Y a-t-il accord faible ?

$A \rightarrow B : \{Na,A\}pk(B)$
$B \rightarrow A : \{Na,Nb,B\}pk(A))$
$A \rightarrow B : \{Nb\}pk(B)$



Propriétés d'authentification
Exemple 3 : Y a-t-il accord faible ?

$A \rightarrow B : \{Na,A\}pk(B)$
$B \rightarrow A : \{Na,Nb,B\}pk(A))$
$A \rightarrow B : \{Nb\}pk(B)$



Scyther results : verify

			Status		Comments
B	Exemple3,B1	Alive	Ok	Verified	No attacks.
	Exemple3,B2	Weakagree	Ok	Verified	No attacks.
	Exemple3,B3	Niagree	Ok	Verified	No attacks.

Propriétés de sécurité : persistance de confidentialité

Persistance de confidentialité (PFS)

Un protocole garantit la persistance de confidentialité si, en cas de compromission des clés long-termes, l'intrus ne peut découvrir rétrospectivement aucune clé de session utilisée précédemment.

Propriétés de sécurité : persistance de confidentialité

Persistance de confidentialité (PFS)

Un protocole garantit la persistance de confidentialité si, en cas de compromission des clés long-termes, l'intrus ne peut découvrir rétrospectivement aucune clé de session utilisée précédemment.

Remarque :

- Les protocoles utilisant Diffie-Hellman ont cette propriété.

Propriétés de sécurité : persistance de confidentialité

Persistance de confidentialité (PFS)

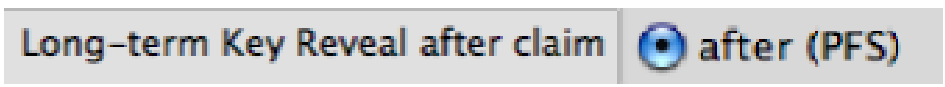
Un protocole garantit la persistance de confidentialité si, en cas de compromission des clés long-termes, l'intrus ne peut découvrir rétrospectivement aucune clé de session utilisée précédemment.

Remarque :

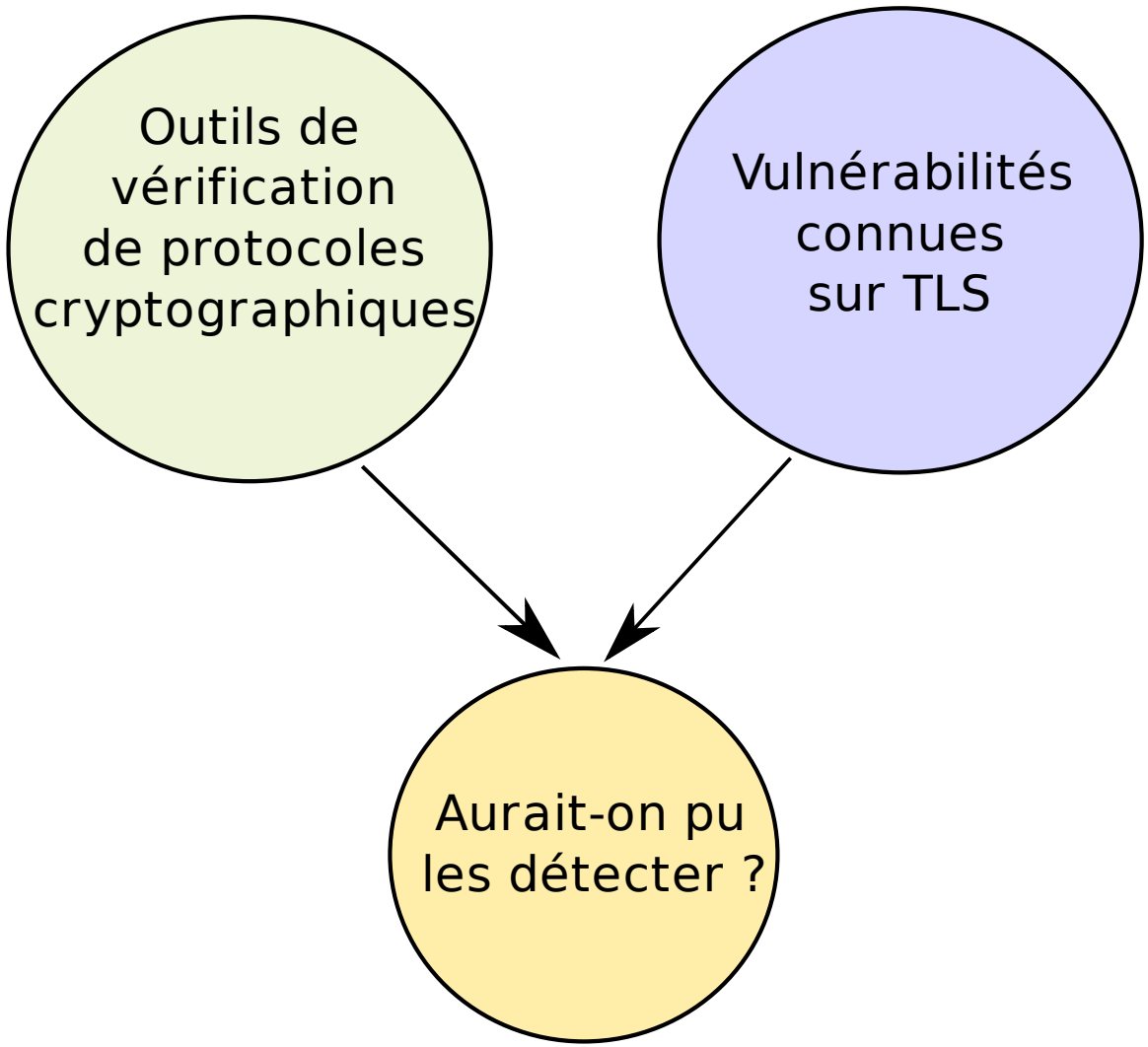
- Les protocoles utilisant Diffie-Hellman ont cette propriété.

En pratique :

- la clé long-terme est donnée à l'attaquant dans une phase suivant une première exécution du protocole sans connaissance de cette clé.
- dans ProVerif : phase 1; out(c, skA)
- dans Scyther : sélectionner l'option PFS :

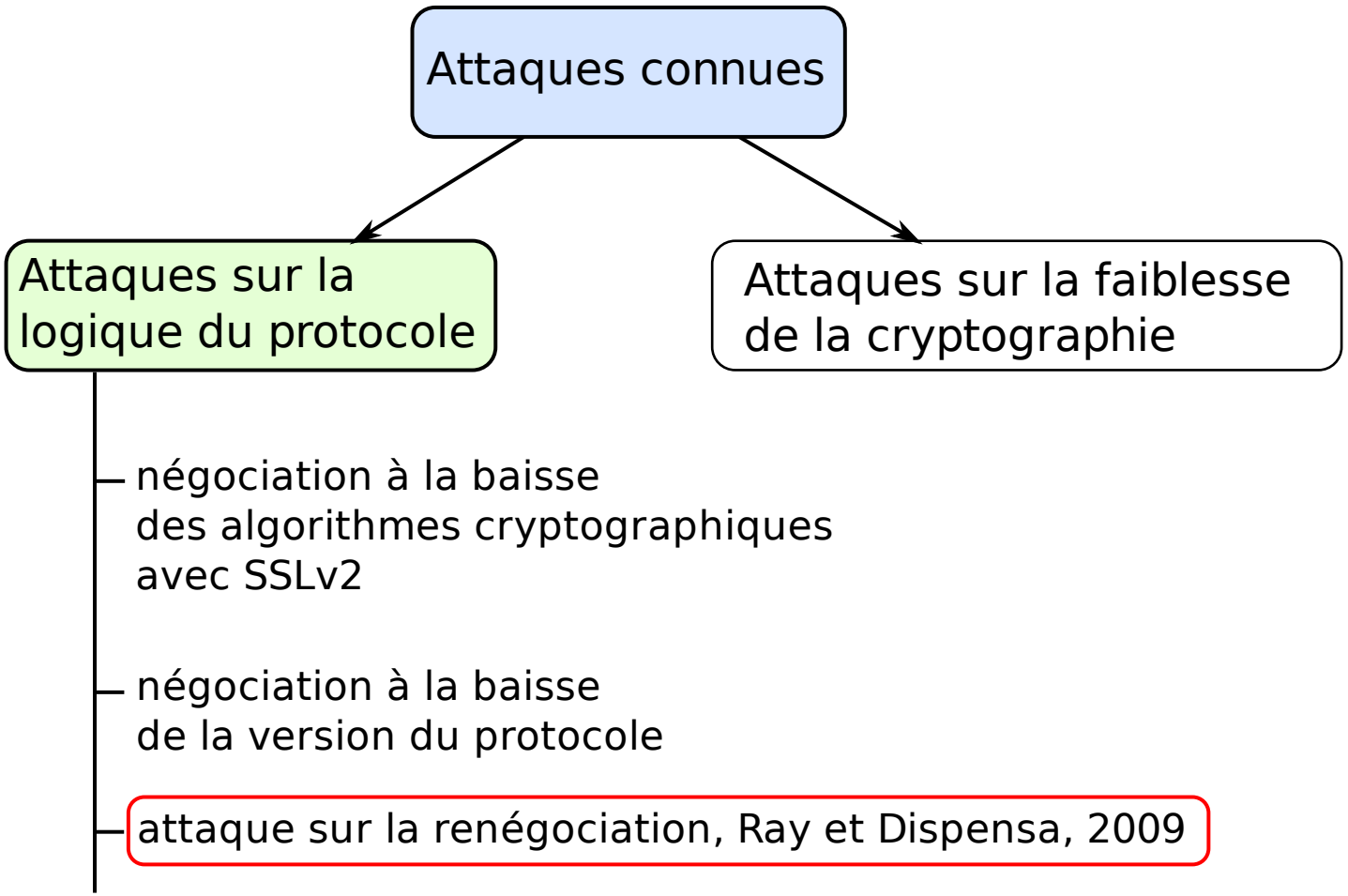


Vers un exemple plus proche de la réalité

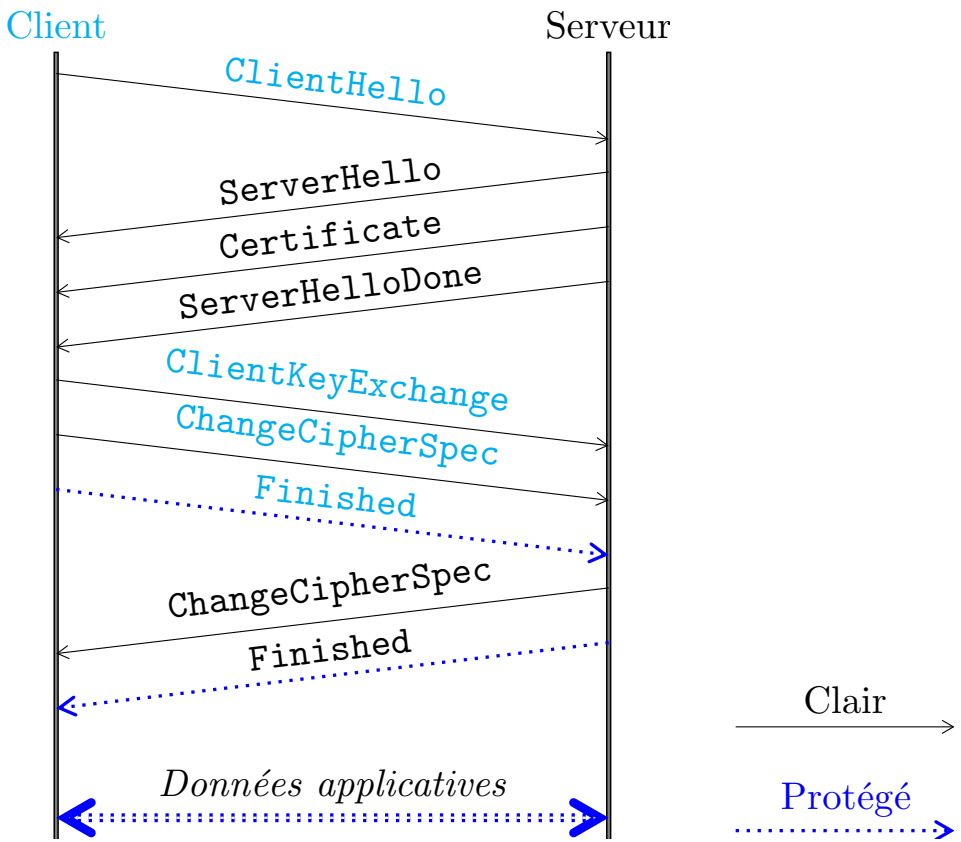


Plan

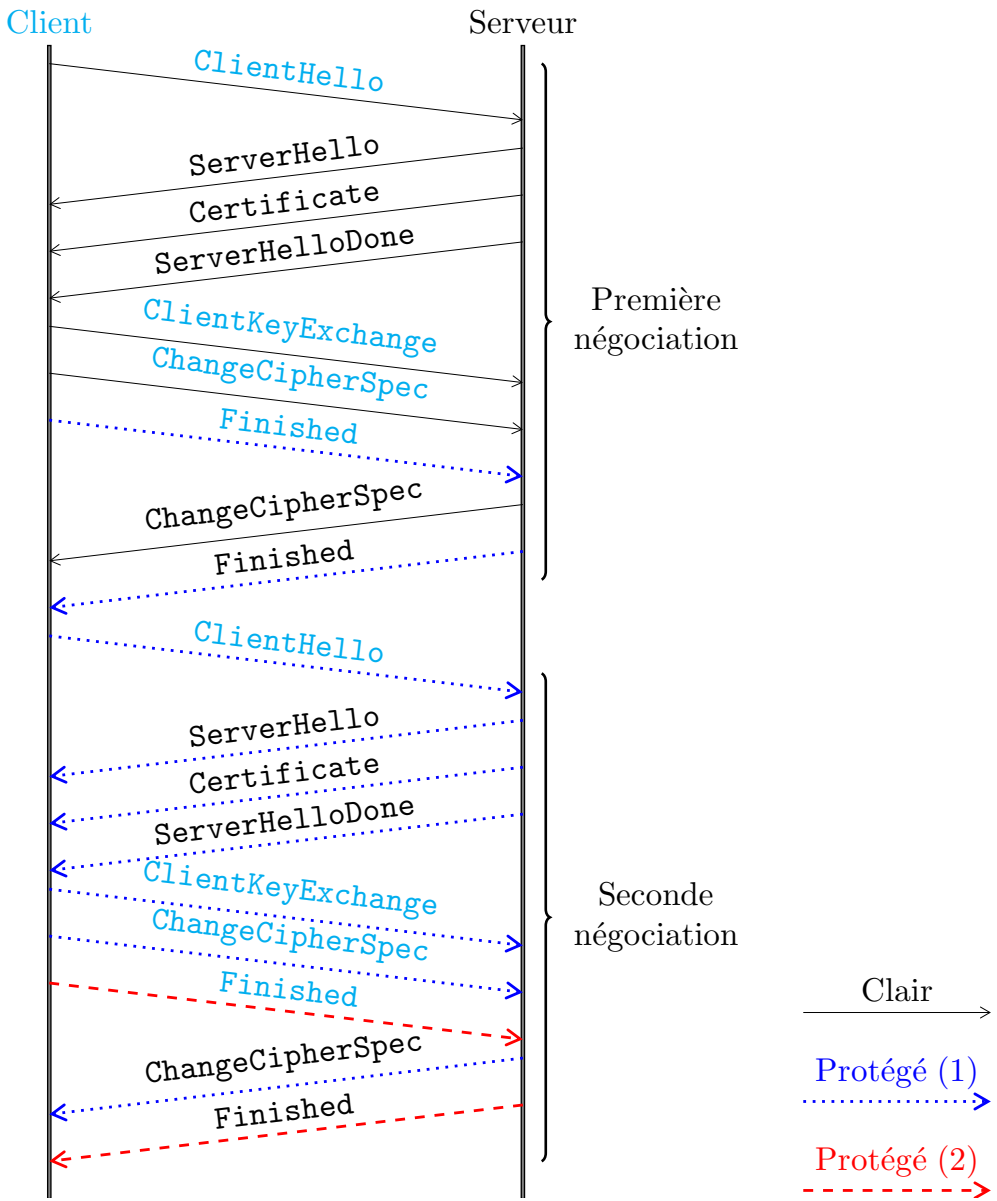
TLS (*Transport Layer Security*) : brève présentation



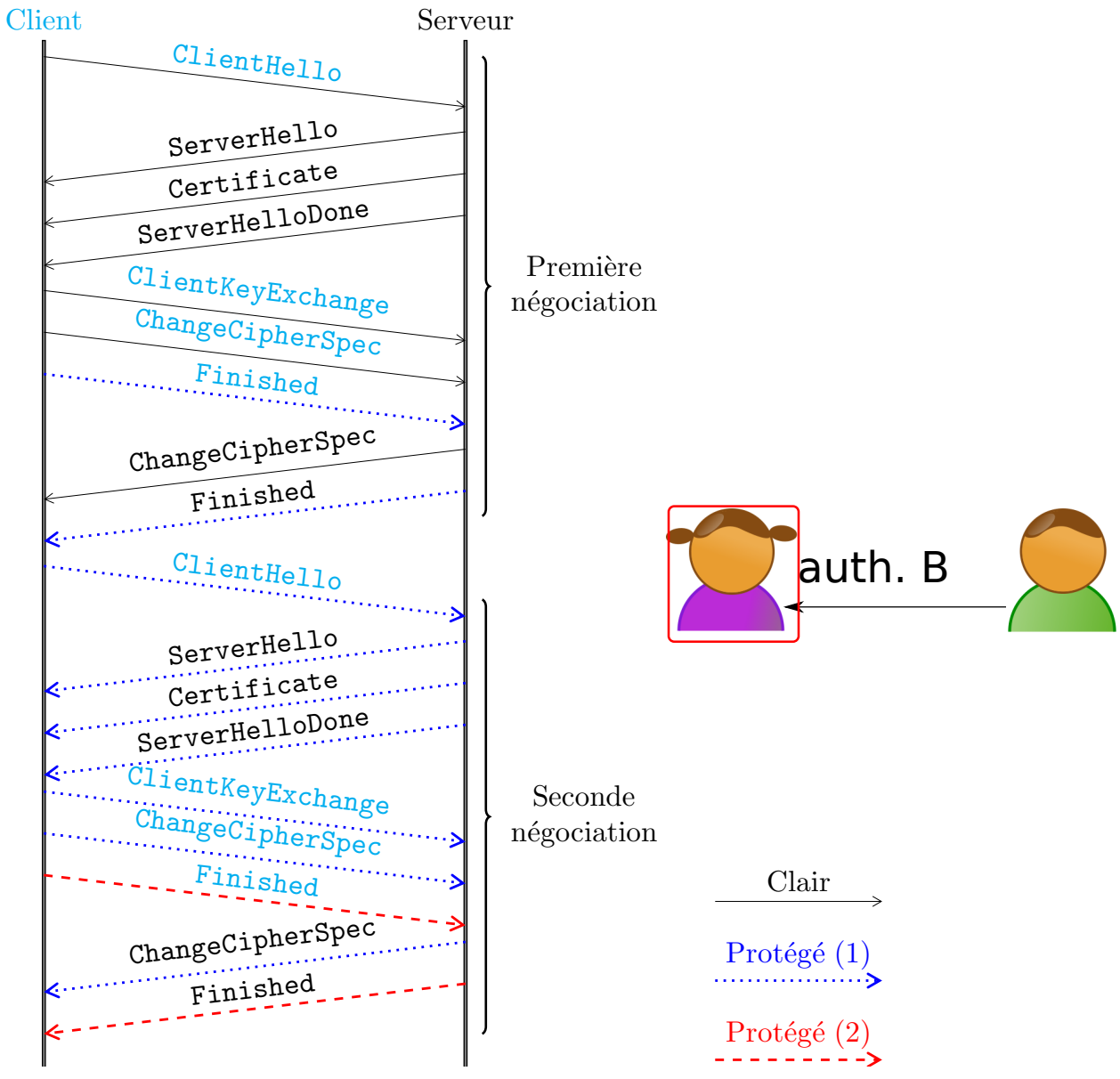
TLS : déroulement de la négociation



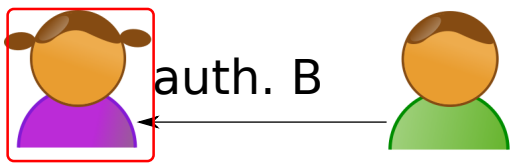
TLS : déroulement de la renégociation



TLS : déroulement de la renégociation

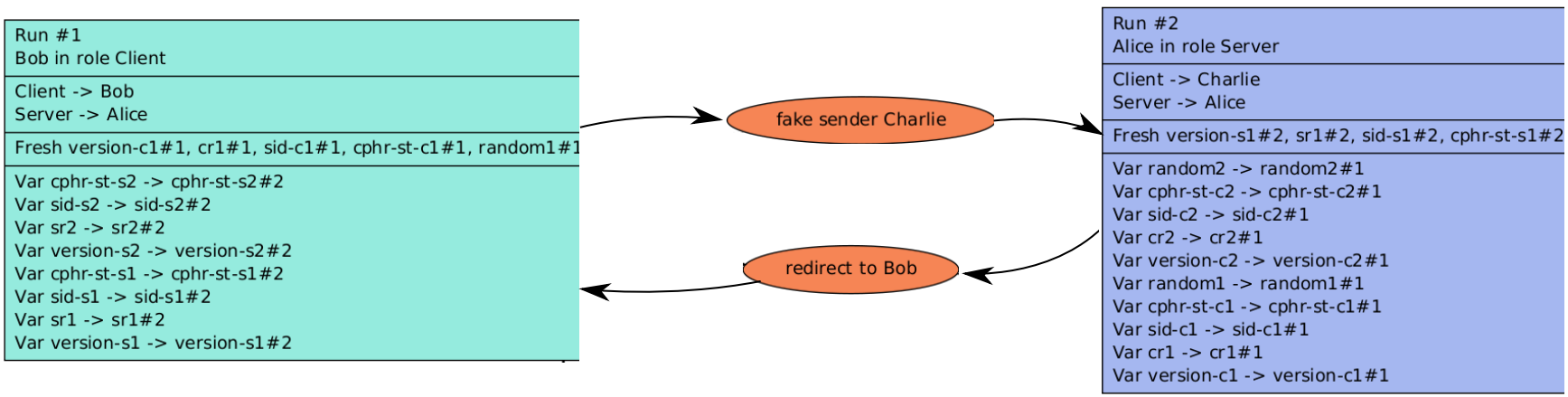


Renégociation TLS : vérification des propriétés d'authentification du serveur avec Scyther et ProVerif

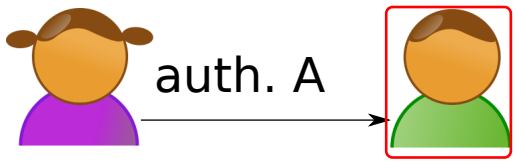


	Scyther	ProVerif
Vitalité	prop. prouvée	prop. prouvée
Accord faible	prop. fausse	prop. fausse
Accord sur tous les termes	prop. fausse	prop. fausse

Renégociation TLS : contre-exemple à l'accord faible du serveur (Scyther et ProVerif)

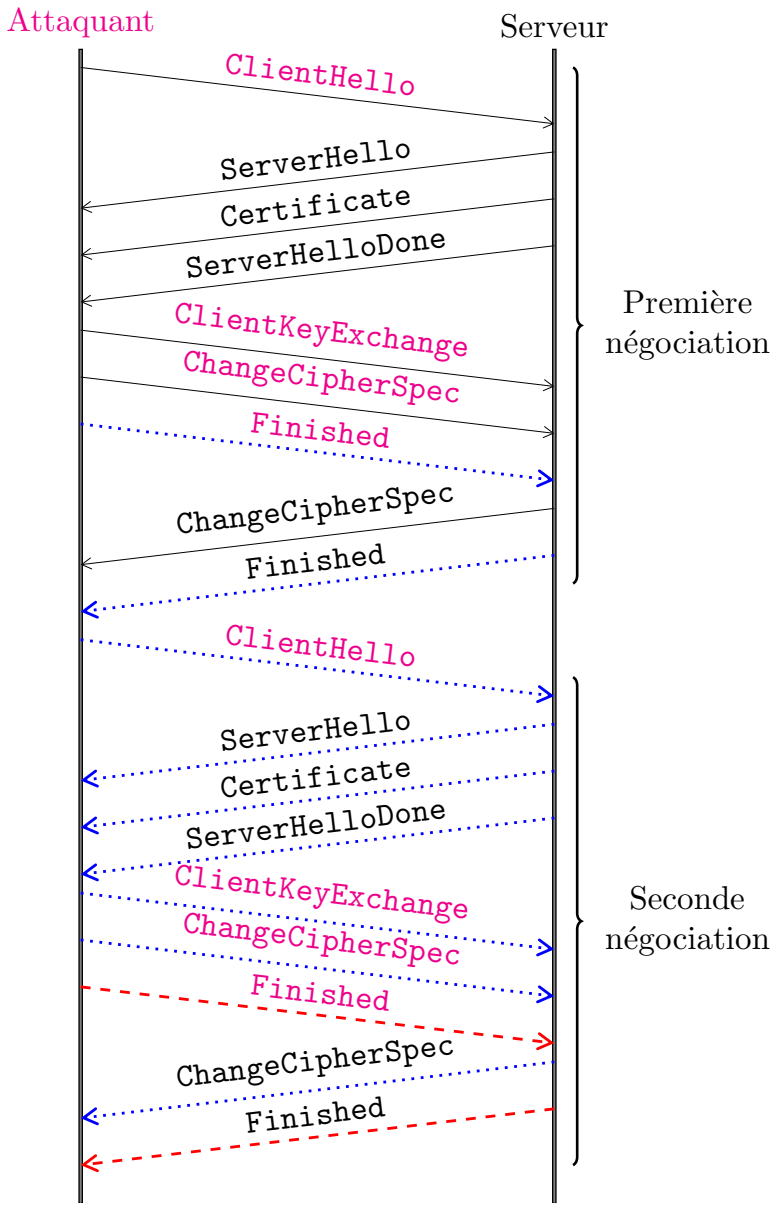


Renégociation TLS : vérification des propriétés d'authentification du client avec Scyther et ProVerif

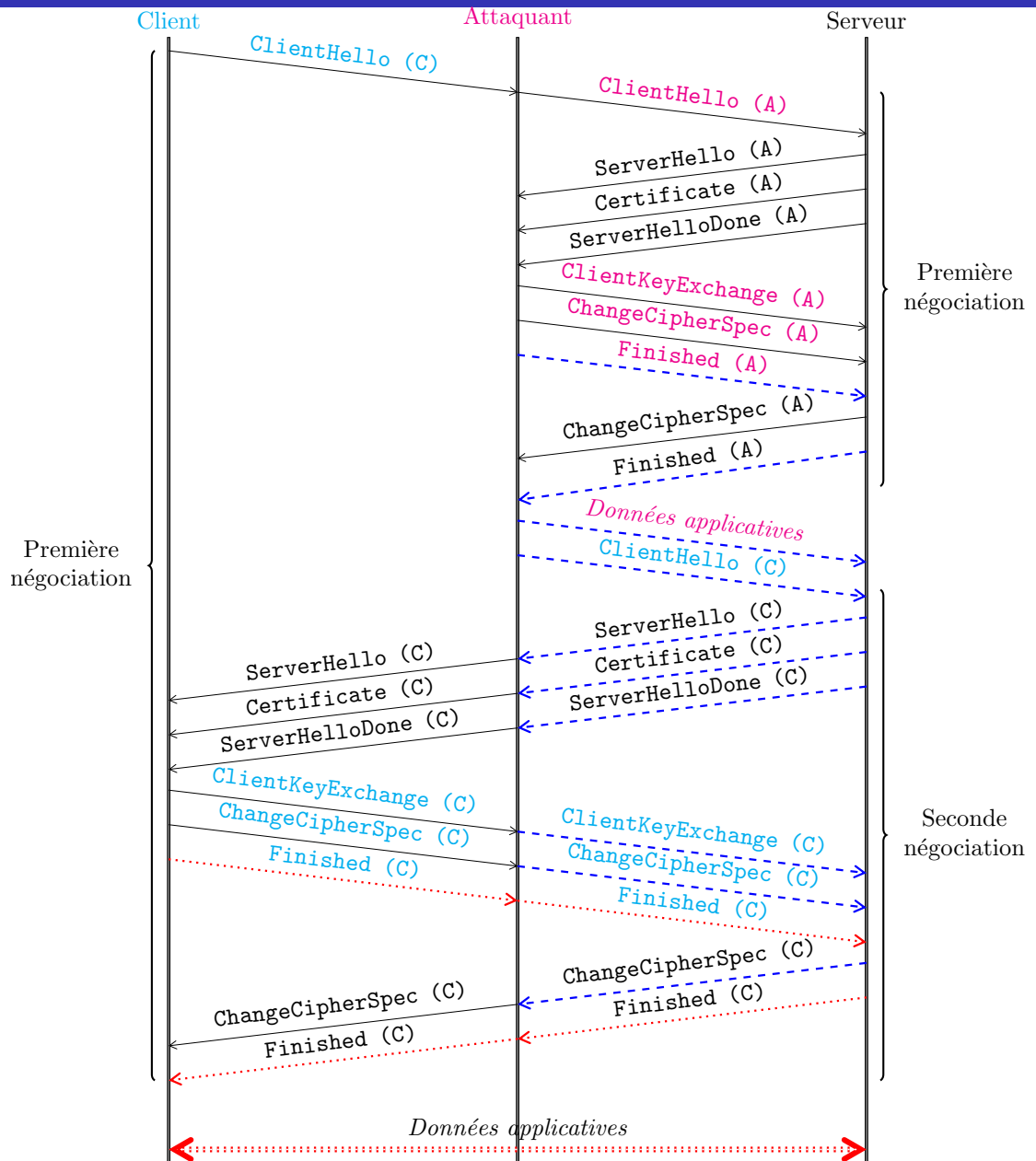


	Scyther	ProVerif
Vitalité	prop. fausse	prop. fausse
Accord faible	prop. fausse	prop. fausse
Accord sur tous les termes	prop. fausse	prop. fausse

Renégociation TLS : contre-exemple à la vitalité du client (ProVerif)



Renégociation TLS : contre-exemple à l'accord faible du client (ProVerif)



Renégociation TLS : contre-exemple à l'accord faible du client. Extrait du résultat donné par ProVerif

Client

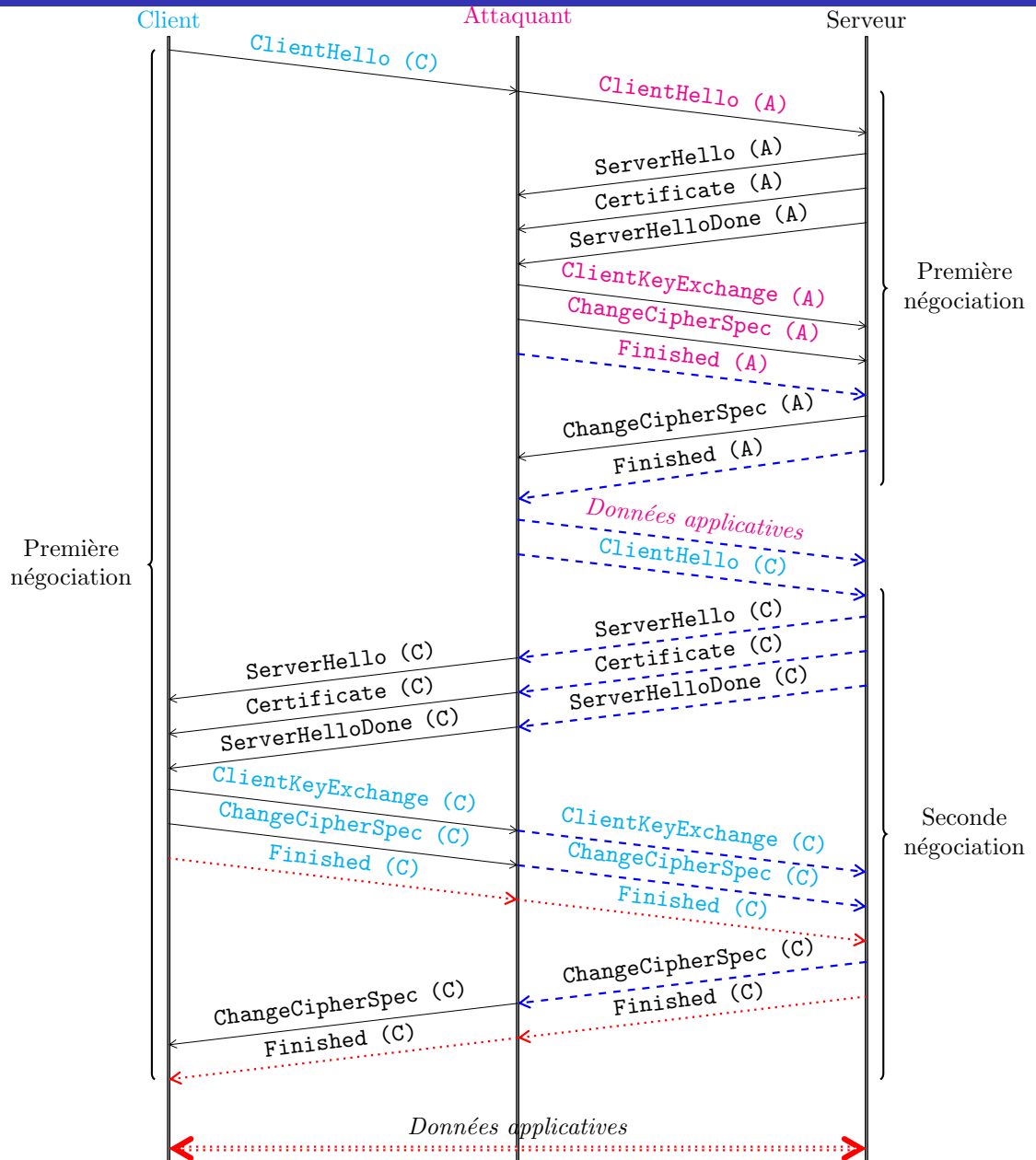
- out(ch, (verc1_1,cr1_2,sidc1_3,cyph_suite_c1_4)) at {17}
  in copy a_11

Attaquant

- in(ch, (a_5,a_6,a_7,a_8)) at {66}
  in copy a

- in(ch, senc((verc1_1,cr1_2,sidc1_3,cyph_suite_c1_4),
  kdf((prf((prf(((a_5,a_9),a_6,sr3_10)),sr3_10,a_6))))))
  at {85} in copy a

Renégociation TLS : contre-exemple contredisant l'accord faible du client



Renégociation TLS : contre-mesures et recommandations

- RFC 5746
- « Recommandations pour la sécurisation des sites web » ANSSI
- Olivier Levillain, « SSL/TLS: état des lieux et recommandations », SSTIC 2012

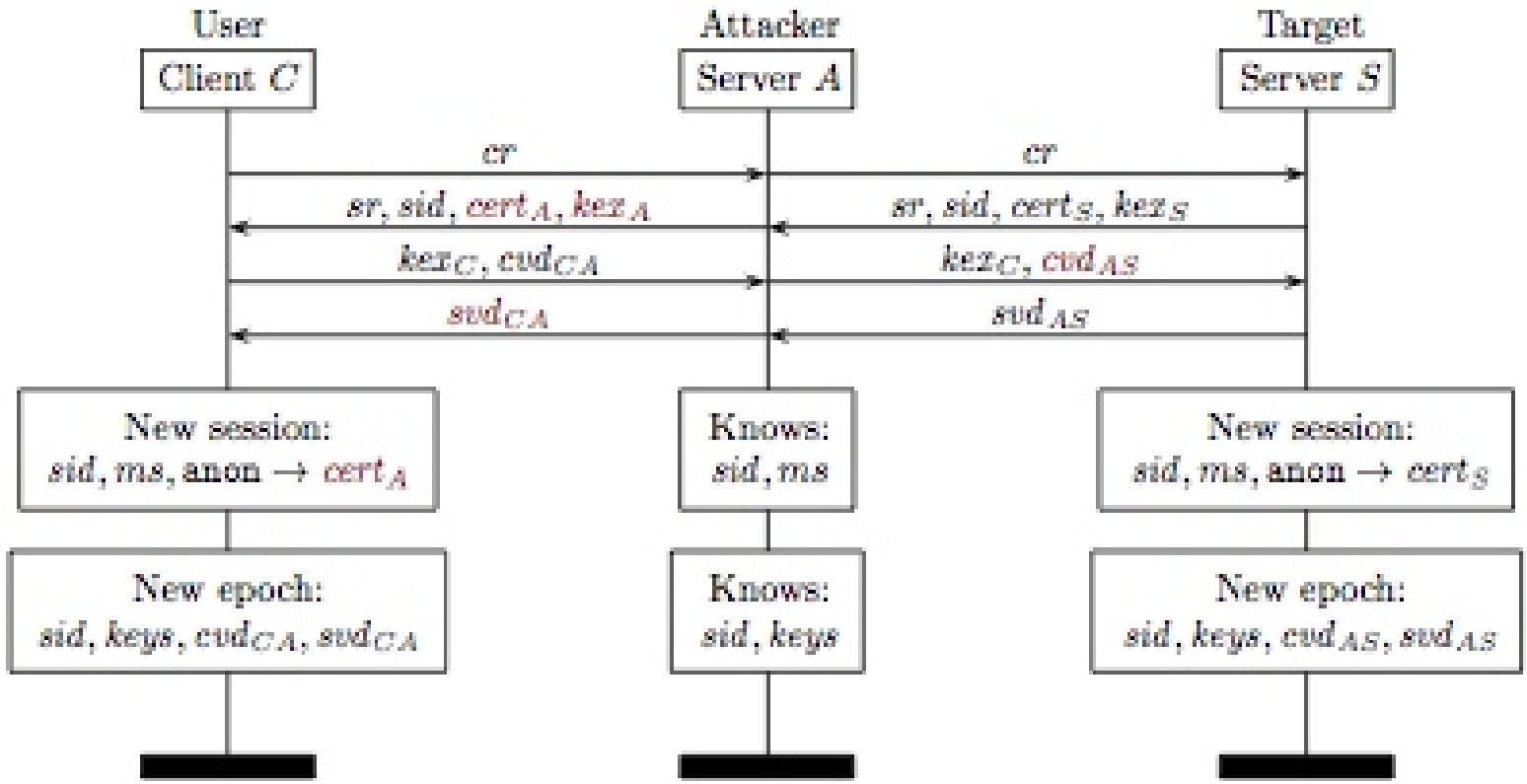
`www.ssi.gouv.fr`

Plan

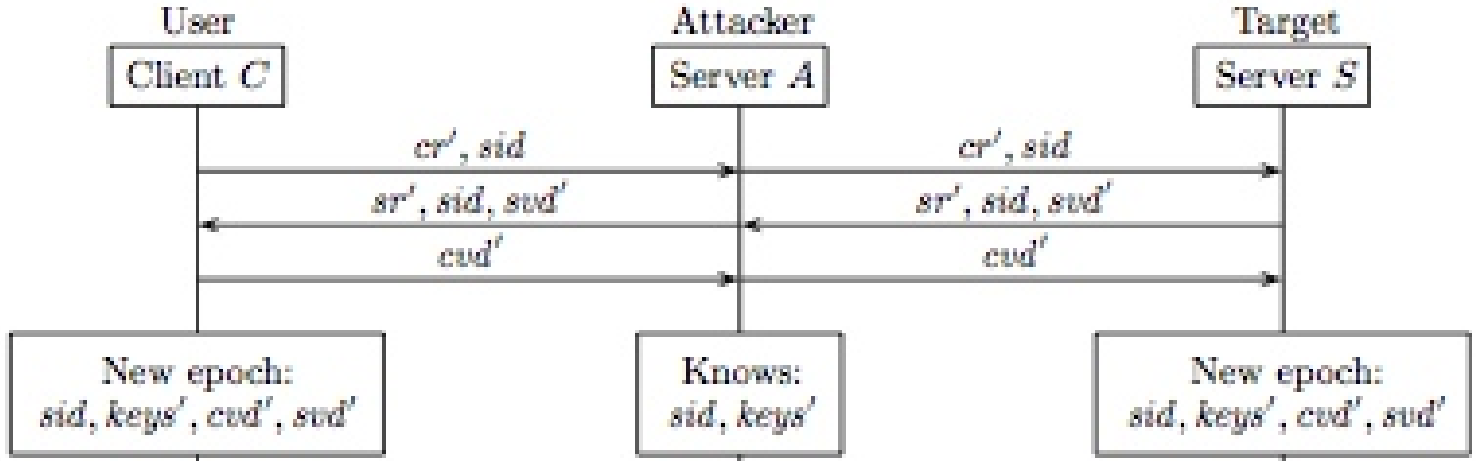
Conclusion

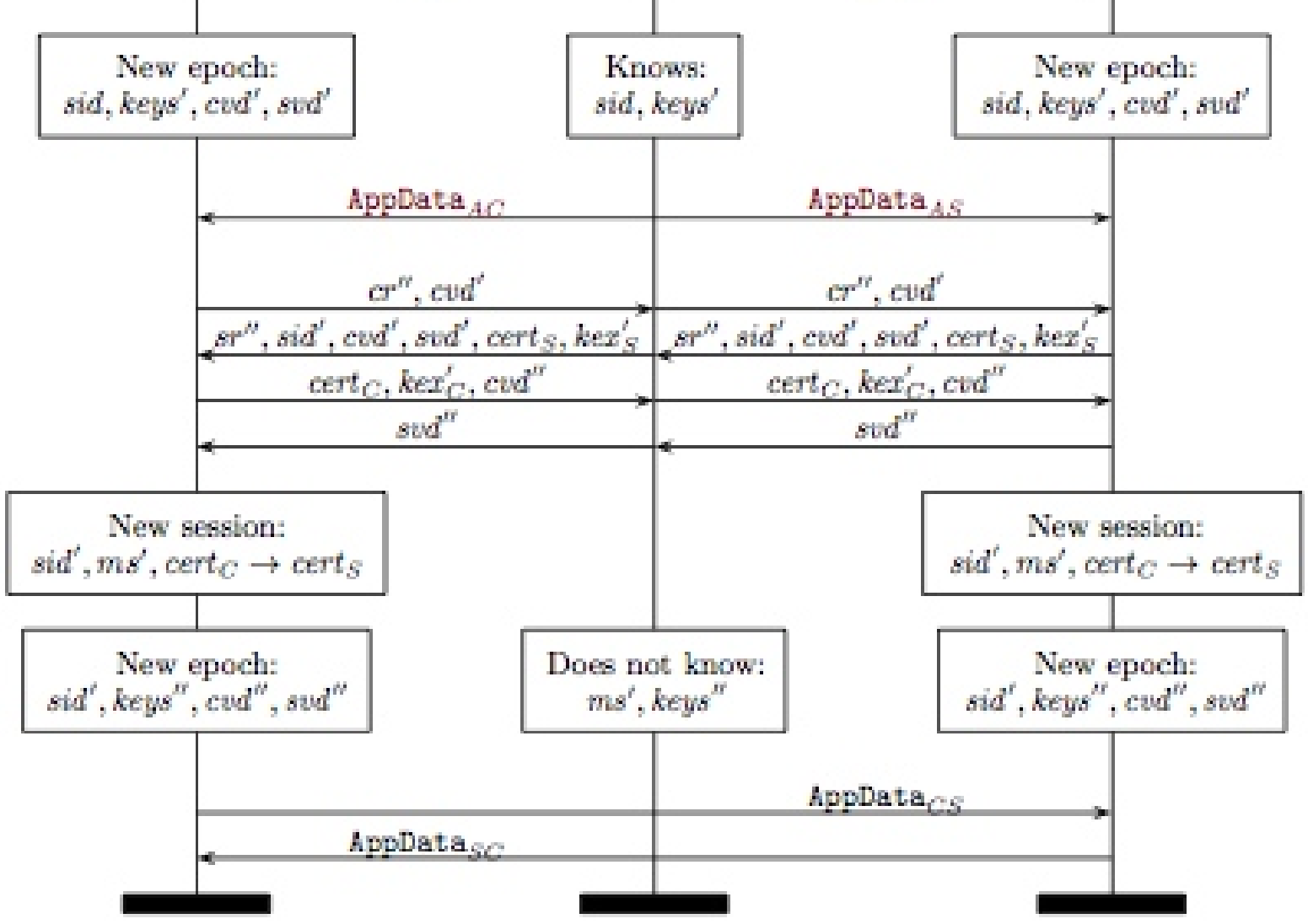
- Le moins qu'on puisse attendre de tout outil, actuel ou à venir, est sa capacité à trouver les attaques connues.
- Dans l'état actuel, les outils permettent de détecter des faiblesses de protocoles contemporains.
- L'efficacité (et le temps de calcul) des outils disponibles impose par prudence d'en croiser les résultats.
- Le savoir faire du modélisateur a une réelle importance
- Une preuve n'est valide que sous certaines hypothèses.

Connection 1: Full Handshake



Connection 2: Session Resumption and Renegotiation





Karthikeyan Bhargavan, Antoine Delignat-Lavaud, Cédric Fournet, Alfredo Pironti et Pierre-Yves Strub. *Triple Handshakes and Cookie Cutters: Breaking and Fixing Authentication over TLS*. 35th IEEE Symposium on Security and Privacy. Mai 2014. (à paraître)

Et l'avenir ?

- L'intégration des nouveaux modèles d'attaquants
- Des propriétés de sécurité plus précises, plus réalistes et peut être plus utiles
- Un nouveau domaine : *computational soundness*
- Une diversification des outils utilisant le modèle calculatoire
- Génération automatique d'implémentation de protocoles cryptographiques vérifiés.

Quelques références

1. Cas Cremers et Sjouke Mauw. *Operational Semantics and Verification of Security Protocols*. Information Security and Cryptography. Springer, 2012.
2. Bruno Blanchet. *ProVerif : Cryptographic Protocol Verifier in the Formal Model*. 2001-2013. `http://prosecco.gforge.inria.fr/personal/bblanche/proverif`
3. Gavin Lowe. *A Hierarchy of Authentication Specifications*. Pages 31-43. IEEE Computer Society Press, 1997.
4. David Basin and Cas Cremers. *Modeling and Analyzing Security in the Presence of Compromising Adversaries*. Pages 340-356. ESORICS'10 Proceedings of the 15th European conference on Research in computer security
5. Cas Cremers. *Key Exchange in IPsec revisited: Formal Analysis of IKEv1 and IKEv2*. Pages 315-334. ESORICS'11 Proceedings of the 16th European conference on Research in computer security, 2011
6. Larafa, Claire Sondès, « Propriétés d'authentification dans ProVerif et Scyther : comparaison et application à TLS », SARSSI 2013
7. Olivier Levillain, « SSL/TLS: état des lieux et recommandations », SSTIC 2012
8. Karthikeyan Bhargavan, Antoine Delignat-Lavaud, Cédric Fournet, Alfredo Pironti et Pierre-Yves Strub. *Triple Handshakes and Cookie Cutters: Breaking and Fixing Authentication over TLS*. 35th IEEE Symposium on Security and Privacy. Mai 2014 (à paraître).

Annexe : étude de IKEv2 (cf. [5])

IKEv2 est composé de 3 échanges :

- phase 1 :

1er échange IKE_SA_INIT :

$$A \to B : HDR, SAa1, KEa, Na$$
$$B \to A : HDR, SAb1, KEb, Nb, [CERTREQ]$$

2ème échange IKE_AUTH :

$$A \to B : HDR, SK\{IDa, [IDb], [CERT], [CERTREQ], AUTH, SAa2, TSa, TSb\}$$
$$B \to A : HDR, SK\{IDb, [CERT], AUTH, SAb2, TSa, TSb\}$$

- phase 2 :

3ème échange CREATE_CHILD_SA :

$$A \to B : HDR, SK\{SA, Na, [KEa], TSa, TSb\}$$
$$B \to A : HDR, SK\{SA, Nb, [KEb], TSa, TSb\}$$

Annexe : étude de IKEv2

La phase 1 utilise l'une des méthodes d'authentification suivantes :

- MAC
- signature numérique

La phase 2 admet deux versions : avec DH et sans DH

Les propriétés de sécurité examinées :

1. secret des valeurs DH privées (sec. DH)
2. secret des clés de session (sec. clés session)
3. confidentialité des identités des participants (sec. id)
   - authentification mutuelle :
     4. vitalité
     5. accord faible
     6. accord
7. PFS

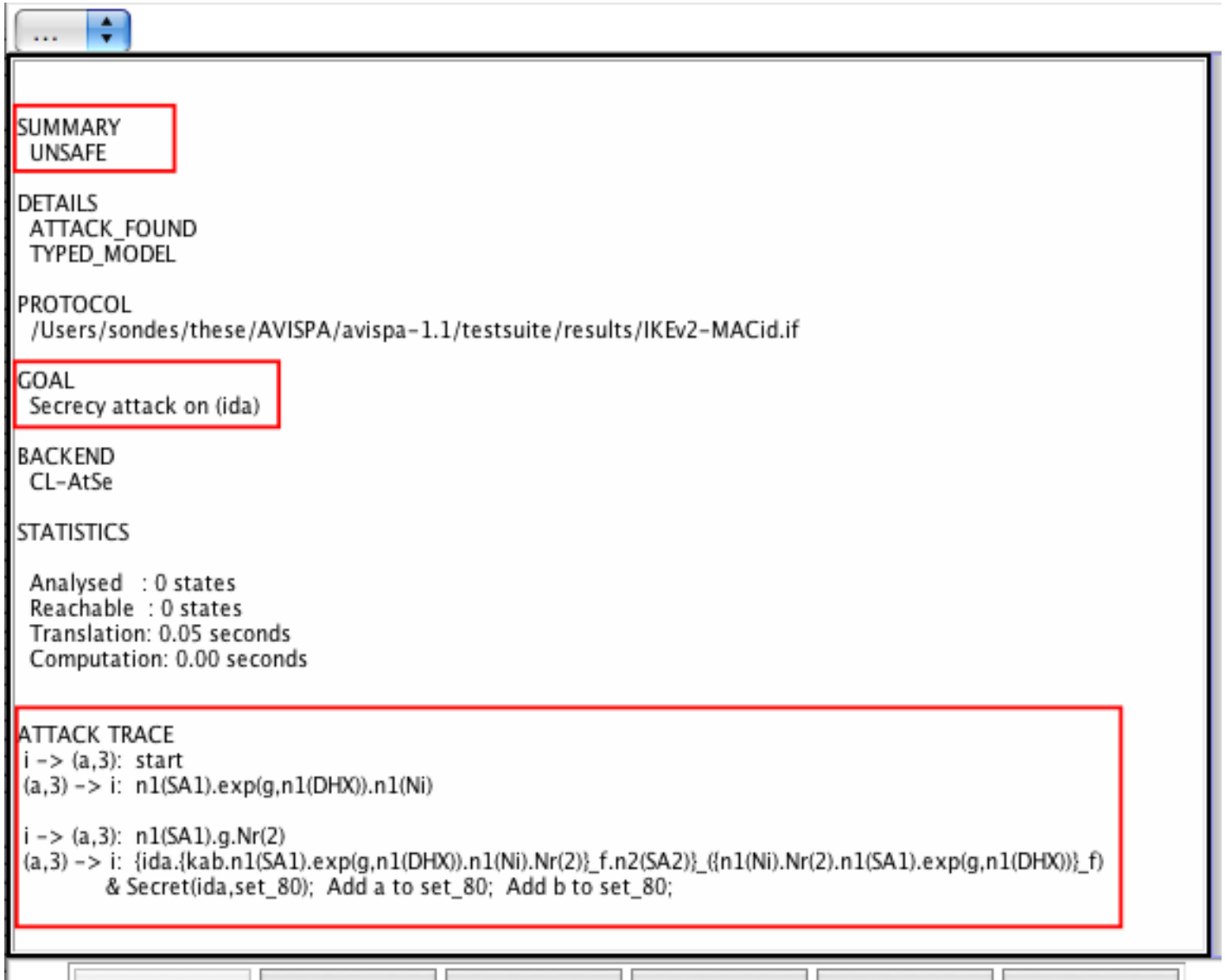
Annexe : étude de IKEv2

Résultats de la vérification avec AVISPA et Scyther :

Sous-protocole	**Propriétés vérifiées**	**Propriétés violées**
IKEv2 ph1 avec MAC	1. sec. DH, 2. sec. clés session, 4. vitalité, 5. accord faible, 6. accord, 7. PFS	3. sec. id
IKEv2 ph1 avec signature	1. sec. DH, 2. sec. clés session, 4. vitalité, 7. PFS	3. sec. id, 5. accord faible, 6. accord
IKEv2 ph2 avec DH	1. sec. DH, 2. sec. clés session, 7. PFS	4. vitalité, 5. accord faible, 6. accord
IKEv2 ph2 sans DH	$\emptyset$	4. vitalité, 5. accord faible, 6. accord, 7. PFS

Annexe : étude de IKEv2

Violation de la confidentialité de l'identité de l'initiateur dans IKEv2 ph1 :



Annexe : étude de IKEv2

Violation de la confidentialité de l'identité de l'initiateur dans IKEv2 ph1 :

- Attaque sur le secret de la clé DH :

$$A \rightarrow I(B) : HDR, SAa1, KEa, Na$$
$$I(B) \rightarrow A : HDR, SAb1, KEi, Ni$$
$$A \rightarrow I(B) : HDR, SK\{IDa, AUTH, SAa2, TSa, TSb\}$$

$\Rightarrow$ L'attaquant se fait passer pour B au près de A :

- Il bloque le 1^{er} message
- Il génère une valeur DH publique $KEi \rightarrow$ La clé de chiffrement du $3^{ème}$ message est partagée entre A et l'attaquant

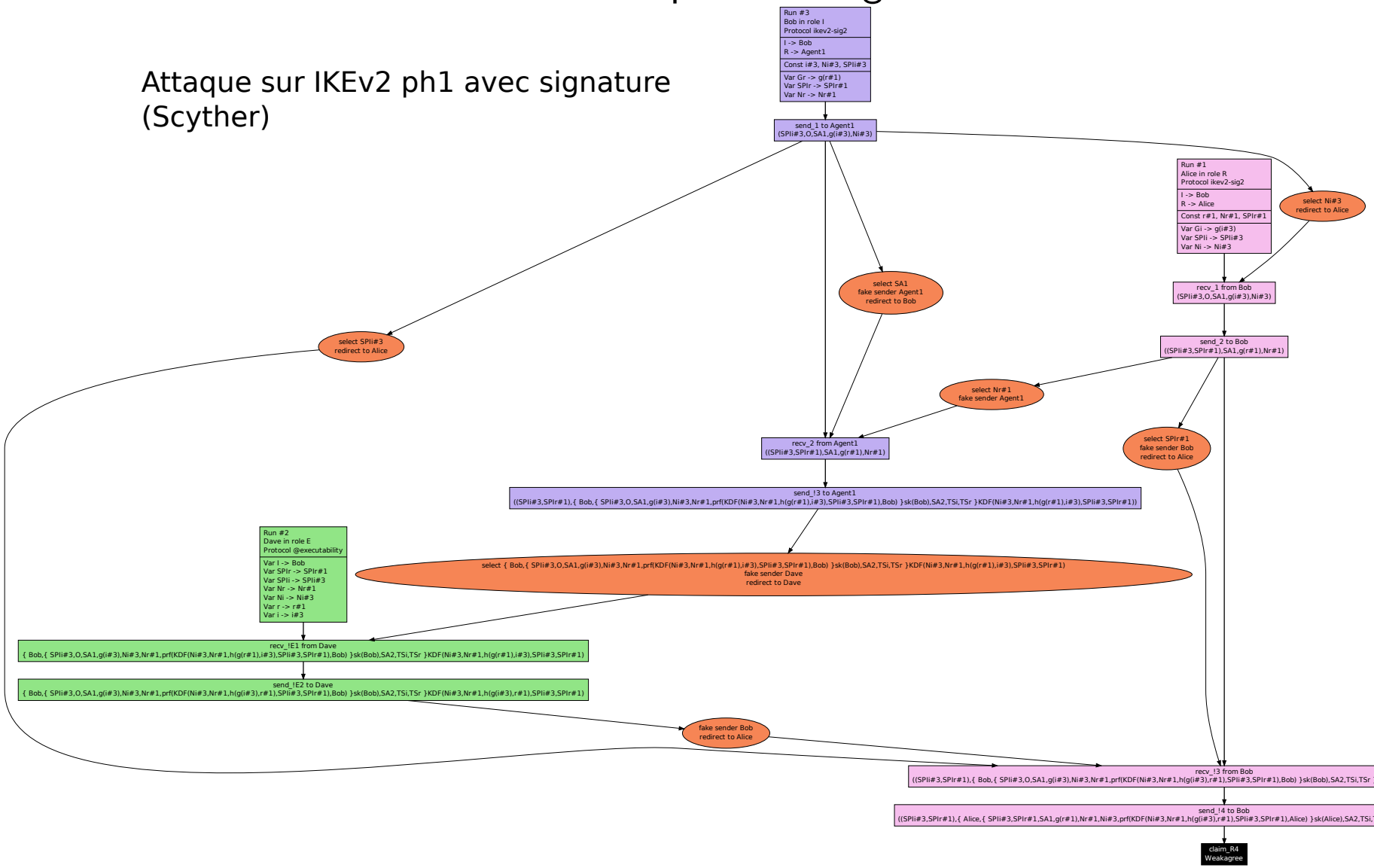
$\Rightarrow$ L'attaquant peut déchiffrer et obtenir IDa

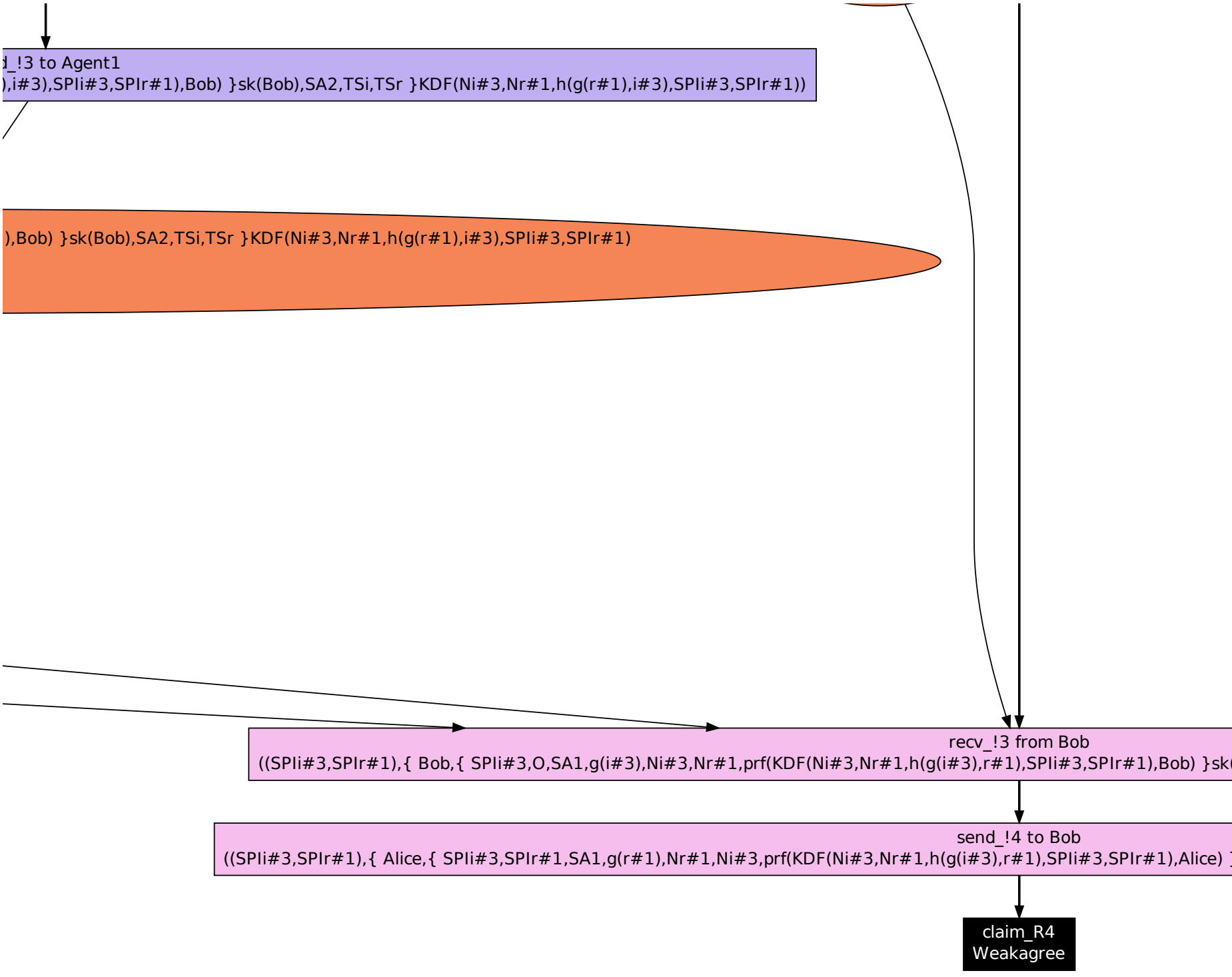
- Si le champ optionnel IDb est présent dans le $3^{ème}$ message, I peut aussi le découvrir.

Annexe : étude de IKEv2

Violation de l'accord faible dans IKEv2 ph1 avec signature :

Attaque sur IKEv2 ph1 avec signature (Scyther)





ype Weakagree, cost 65

Annexe : étude de IKEv2

Violation de l'accord faible dans IKEv2 ph1 avec signature, explications :

- Attaque de l'homme du milieu :

$$
\begin{array}{lll}
A \to I : HDR, SAa1, KEa, Na & & \\
 & I(A) \to B : HDR, SAa1, KEa, Na & \\
 & B \to I(A) : HDR, SAb1, KEb, Nb & \\
I \to A : HDR, SAb1, KEb, Nb & & \\
A \to I : HDR, SK\{IDa, AUTH, SAa2, TSa, TSb\} & & \\
 & I(A) \to B : HDR, SK\{IDa, AUTH, SAa2, TSa, TSb\} & \\
 & B \to I(A) : HDR, SK\{IDb, AUTH, SAb2, TSa, TSb\} &
\end{array}
$$

$\Rightarrow$ L'attaquant se fait passer pour A au près de B.

- Il bloque le dernier message.
- Du point de vue de B, A a bien exécuté le protocole $\to$ vitalité.
- Mais, A et B ne sont pas d'accord sur les identités de l'initiateur et du répondant :

  pour A : initiateur = A et répondant = I
  pour B : initiateur = A et répondant = B

$\to$ violation de l'accord faible.

Annexe : étude de IKEv2

Violation de l'accord faible dans IKEv2 ph1 avec signature, les remèdes possibles :

- Ajouter le champ optionnel IDb dans le 3$^{\text{ème}}$ message :

$A \rightarrow B$: HDR, SAa1, KEa, Na
$B \rightarrow A$: HDR, SAb1, KEb, Nb
$A \rightarrow B$: HDR, SK{IDa, **IDb**, AUTH, SAa2, TSa, TSb}
$B \rightarrow A$: HDR, SK{IDb, AUTH, SAb2, TSa, TSb}

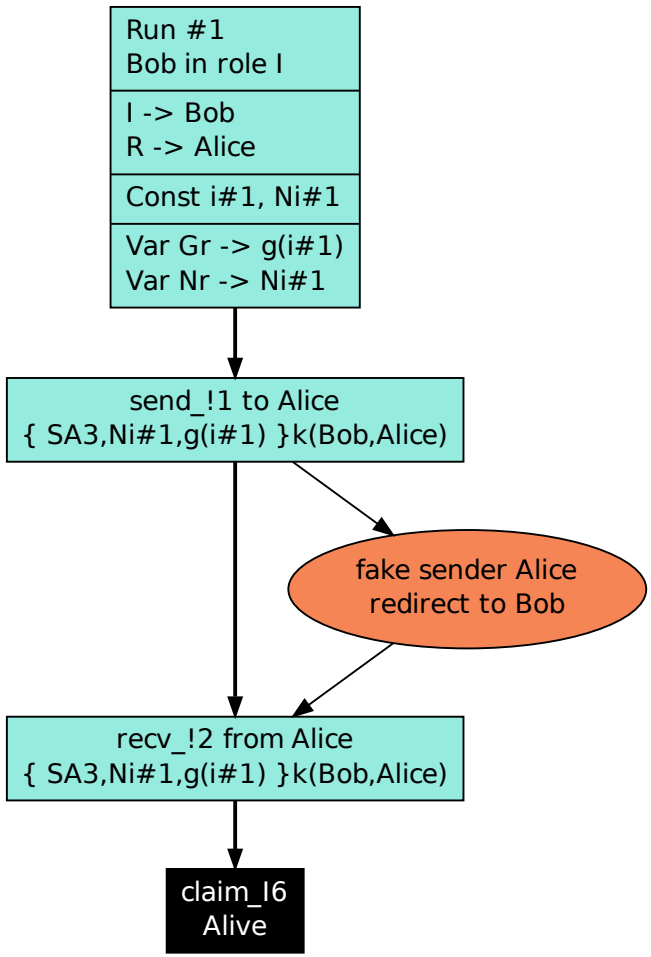
- Confirmer la valeur de la nouvelle clé grâce à un échange ultérieur de données chiffrées.

Annexe : étude de IKEv2

Violation de la vitalité dans IKEv2 ph2 :

Attaque sur IKEv2 ph2

(Scyther)



[Id 5] Protocol ikev2-child, role I, claim type Alive, cost 19

L'attaquant renvoie le message de l'initiateur en guise de réponse → attaque par réflexion.

Le répondant n'a pas exécuté le protocole → violation de la vitalité

Annexe : étude de IKEv2

Violation de la vitalité dans IKEv2 ph2, les remèdes :

- Confirmer la valeur de la nouvelle clé grâce à un échange ultérieur de données chiffrées.

Annexe : étude de IKEv2

Violation de la vitalité dans IKEv2 ph2, les remèdes :

- Confirmer la valeur de la nouvelle clé grâce à un échange ultérieur de données chiffrées.

Que conclure sur les faiblesses d'IKEv2 :

- confidentialité des identités : problème réél.
- violation de l'accord faible dans la ph1 avec signature :

→ attaques plutôt théoriques ?...

→ par prudence, ajouter *IDb* et/ou confirmer la clé de session établie.

- violation de la vitalité dans la ph2 :

→ attaques plutôt théoriques ?...

→ par prudence, confirmer la clé de session établie.

- privilégier la version avec DH de la ph2.