Protocoles cryptographiques

Dr. Y. Challal

Contexte et motivation

- **Les algorithmes de cryptographie ne suffisent pas pour garantir le secret ou l'authenticité d'un message**

Hyp: Kab est une clef secrète entre A et B

Comment le garantir?

Protocole de cryptographie

- **Un protocole cryptographique est une séquence d'étapes spécifiant les actions respectives devant être réalisée par deux entités (ou plus) pour remplir un objectif de sécurité donné (exemple: Diffie Hellman pour l'échange de clé).**

- **Un protocole cryptographique utilise des algorithmes de cryptographie.**

Types de protocoles cryptographiques

- **Protocoles d'échange de clefs (key exchange, key establishment)**
  - Création d'un secret partagé (exemple: protocole de DH)
- **Protocoles d'authentification**
  - Authentification de l'origine des données
  - Authentification de l'entité homologue (ou identification)
- **Protocoles combinant authentification et échange de clés**
- **Autres protocoles**
  - Vote électronique
  - Partage de clé de groupe (key agreement)
  - Horodatage et estampillage
  - Partage de secret
  - …

Notations pour algorithmes cryptographiques

- **Kab désignera une clé secrète (utilisée dans un algorithme symétrique) partagée entre a et b**
- **PKa désignera une clé publique (utilisée dans un algorithme asymétrique) de a**
- **Ska désignera une clé privée (utilisée dans un algorithme asymétrique) de a**
- **Si m désigne un message**
  - {m}Kab désignera m chiffré avec Kab
  - {m}PKa désignera m chiffré avec PKa
  - {m}Ska désignera m chiffré avec Ska
  - H(m) désignera un condensé calculé sur m avec la fonction de hachage h
  - Hk(m) désignera un MAC calculé sur m avec la fonction de hachage h paramétrée avec la clé h, Hk.

Notation : protocoles cryptographiques

- **Protocole :=** **Message**
  ...
  Message
- **Message := Mk, Entité -> Entité : Data**
- **Entité := a|b|s|c/a|c/b|c/s (a ali, b bachir, c attaquant, s serveur)**
- **Data :=** **a|b|s**
  K tel que K ds{Ksa,Ksb,Kab,PKa,PKb,PKs,Ska,SKb,SKs}
  ra|rb (Nonces générés resp. par a et b)
  ta|tb (éstampilles générées resp. par a et b)
  Data.Data (concaténation)
  {Data}k donnée chiffrée avec k
  Data* (donnée optionnelle)
- **Nonce (ou aléas) : nombre unique et imprévisible**
- **Estampille : marqueur de temps, sert à calculer la fraîcheur**

Hypothèses sur l'attaquant

- **C peut écouter les messages échangés**
- **C peut bloquer les messages**
- **C peut rediriger les messages**
- **C peut enregistrer les messages**
- **C peut rejouer les messages**
- **C peut fabriquer (chiffrer déchiffrer) de nouveaux messages à partir d'anciens**
- **C ne sait pas déchiffrer (sans avoir la clé) dans le temps d'une session**

Protocoles d'échange de clé (1)

- **Echange de clé en utilisant un système asymétrique**
  - M1: b->a : PKb
  - M2: a->b : {Kab}PKb

- **Ne garantie pas l'authentification: attaque « man in the middle »**
  - M1: b->c/a: PKb
  - M2: c/a->b: {Kcb}PKb

Protocoles d'échange de clé (2) : Otway-Rees

- M1: a->b: m.a.b.{na.m.a.b}Ksa (m identifiant de transaction
- M2: b->s: m.a.b.{na.m.a.b}Ksa.{nb.m.a.b}Ksb
- M3: s->b: m.{na.Kab}Ksa.{nb.Kab}Ksb
- M4: b->a: m.{na.Kab}Ksa
- Supposons que m fait 32 bits, a et b 16 bits, et Kab 64 bits

➢ **Attaque par confusion de type**

- il suffit que c rejoue la partie chiffrée de M1
- M1: a->c/b: m.a.b.{na.m.a.b}Ksa
- M4': c/b->a: m.{na.m.a.b}Ksa

➢ **Conclusion**

- a accepte m.a.b comme nouvelle clé (envoyée en claire !!!)

Protocoles d'authentification de l'origine

- **Objectif: b doit être assuré que le message a été créé par a**
  - Utilisation d'un algorithme asymétrique de signature
    - ✓ M1: a->b : a.{m}Ska
  - Utilisation d'un MAC
    - ✓ M1: a->b a.m.Hk(m)

- **Ne protège pas contre le rejeu**
  - Ajout de paramètre de temps pour assurer la fraîcheur : estampille, numéro de séquence, nombre aléatoire envoyé dans un message précédent

Protocoles d'authentification d'entité

- **Objectif:**
  - A doit être authentifié auprès de b à la fin du déroulement du protocole

- **Types d'identification**
  - Authentification faible : mots de passe fixes, mots de passe à usage unique
  - Authentification forte : protocoles défi-réponse basés sur la cryptographie symétrique ou asymétrique

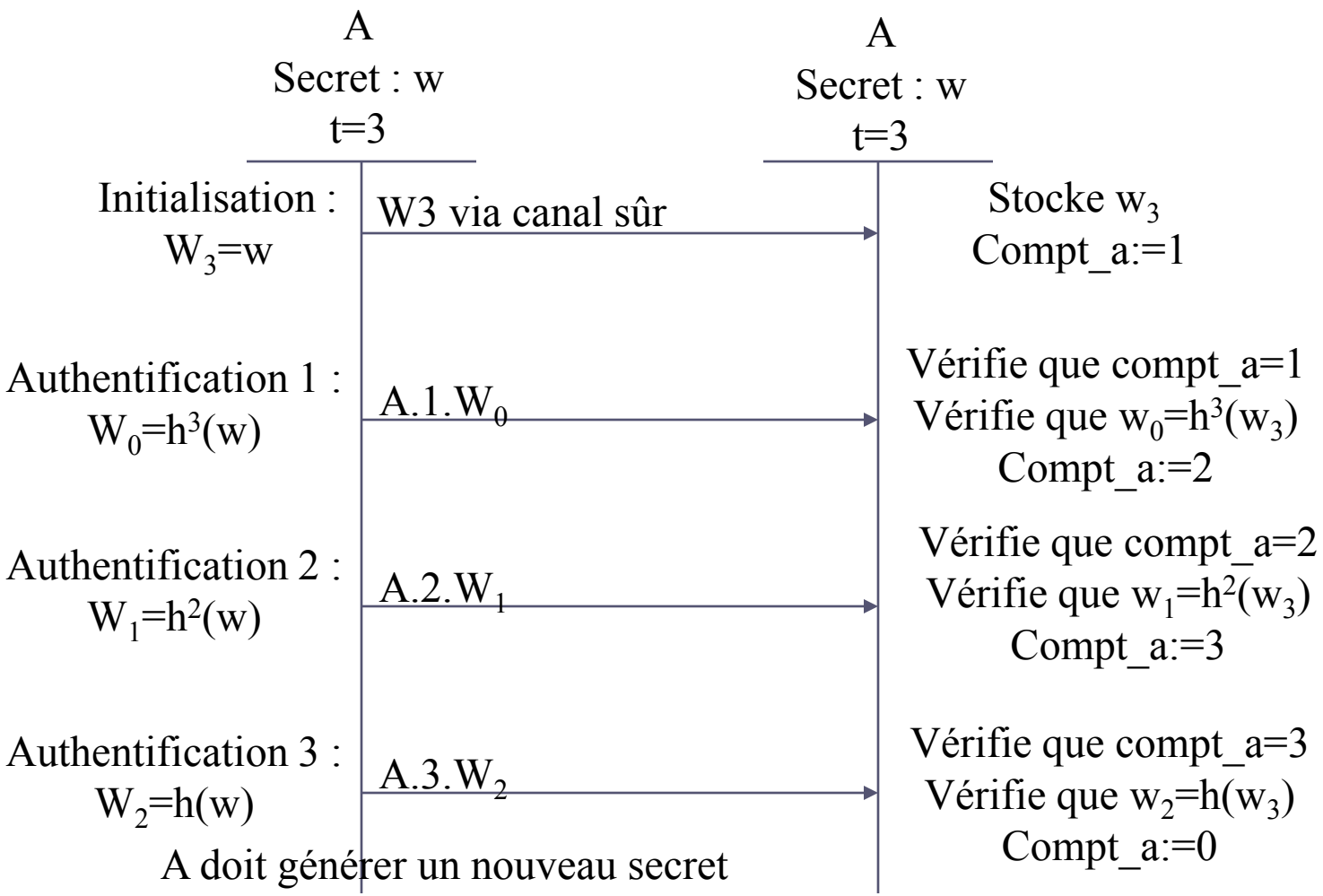
Authentification d'entité faible par mot de passe (1)

- **Mots de passe fixe**
  - Possibilité d'intercepter le mot de passe
  - Mots de passes stockés dans un fichier protégé en lecture et écriture, ou
  - Utilisation de fonction à sens unique avant le stockage (exemple: login sous unix)
  - Attaque par dictionnaire

Authentification d'entité faible par mot de passe (2)

- **Mots de passe à usage unique : vers l'authentification forte**
  - Protège contre les interceptions de message
  - Basé sur une liste pré-partagée
  - Basé sur un incrément
  - Basé sur une fonction à sens unique : exemple protocole de Lamport
    - ✓ A veut s'identifier à b
    - ✓ A possède un secret w, h fct à sens unique connue par a et b
    - ✓ t : constante définissant le nombre d'authentifications autorisées (exple si t=100, après 100 authentification, a génère un nouveau secret w)
    - ✓ On définit par récurrence : $h^0(w)=w$, $h^i(w)=h^{i-1}(h(w))$ pour $1<=i<=t$
    - ✓ On note $w_i=h^{t-i}(w)$, i-ième mot de passe
    - ✓ A transfère à b par un canal sûr $w_0=h^t(w)$, b initialise son compteur compt_a à 1
    - ✓ À la ième identification : a calcule w_i, et envoie à b M1 : a->b : a.i.w_i, b vérifie que compt_a=i et que $h(w_i)=w_{i-1}$

Authentification d'entité faible par mot de passe (2)



Authentification forte par défi réponse

- **Principe**
  - Une entité (le prouveur) prouve son identité à une autre entité (le vérificateur) en démontrant au vérificateur qu'il possède un secret sans révéler ce secret grâce à une réponse à un challenge variant dans le temps
- **Challenge : utilisation de nonce garantissant une notion de fraîcheur / unicité**
  - Nombre pseudo-aléatoire (noté r dans la suite)
    - ✓ Nombre non prévisible par un attaquant
  - Numéro de séquence (noté n dans la suite)
    - ✓ Nécessite de mémoriser les numéros de séquence déjà utilisés
  - Estampille (horodateur, time stamp) + horloges synchronisées (noté t)
    - ✓ Les horloges doivent être sécurisées
  - Combinaison de nonces : nombre aléatoire concaténé à un numéro de séquence ou à une estampille
    - ✓ Permet de garantir qu'un nombre aléatoire n'a pas été dupliqué

Authentification forte par défi réponse

- **Critères de classification**
  - Système cryptographique à clés publiques ou symétriques
  - Nombre de messages
  - Authentification unilatérale ou mutuelle

- **Types de protocoles par défi-réponse**
  - Protocoles utilisant une clé secrète (chiffrement symétrique ou MAC)
  - Protocoles basés sur un chiffrement avec clé publique
  - Protocoles basés sur la vérification d'une signature

Authentification forte par défi réponse
Clé partagée (1)

- **Variante 1**
  - M1: a->b : I'm A
  - M2: b->a : nb
  - M3: a->b : {nb}Kab
- **L'authentification est non mutuelle**

- **Variante 2**
  - M1: a->b : I'm A
  - M2: b->a : {nb}Kab
  - M3: a->b : nb

Authentification forte par défi réponse
Clé partagée (1)

- **Variante 3 : une passe**
  - M1: a->b : a.{ta}Kab
- **Nécessite que A et B aient des horloges synchronisées**
- **Bachir déchiffre le message et s'assure que ta est dans un intervalle de temps raisonnable.**

- **Clé partagée:**
  - Si la base de donnée du côté serveur (B) est corrompue, Ali peut être usurpé par un intrus.
  - Solution: usage des clés publiques

Authentification forte par défi réponse
Clé publique (1)

- **Variante 1**
  - M1: a->b : I'm A
  - M2: b->a : nb
  - M3: a->b : {nb}Ska (A signe le nonce nb avec sa clé privée)
- **Variante 2**
  - M1: a->b : I'm A
  - M2: b->a : {nb}PKa (b chiffre le nonce nb avec la clé publique de A)
  - M3: a->b : nb
- **Authentification unilatérale => un intrus peut faire signer à A un message, ou intercepter un message qui lui est destiné et le lui faire déchiffré !!!**

Authentification forte par défi réponse
Authentification mutuelle : clé partagée (1)

- **Version 1**
  - M1: a->b : I'm A
  - M2: b->a : nb
  - M3: a->b : {nb}Kab
  - M4: a->b : na
  - M5: b->a : {na}Kab
- **Version 2 : réduire le nombre de messages**
  - M1: a->b : I'm A . na
  - M2: b->a : nb.{na}Kab
  - M3: a->b : {nb}Kab
- **Attaque réflexive avec entrelacement de sessions**

Authentification forte par défi réponse
Authentification mutuelle : clé partagée (1)

- **Version 2 : réduire le nombre de messages**
  - M1: a->b : I'm A . na
  - M2: b->a : nb.{na}Kab
  - M3: a->b : {nb}Kab

Session 1	Session 2
M1: c/a->b : I'm A . na M2: b->c/a : nb.{na}Kab	
	M1: c/a->b : I'm A . nb M2: b->c/a : nb'.{nb}Kab
M3: c/a->b : {nb}Kab	

- **Solution:**
  - M1: a->b : I'm A . na
  - M2: b->a : nb.{na.b}Kab
  - M3: a->b : {nb.a}Kab

Authentification forte par défi réponse
Authentification mutuelle : clé partagée (2)

- **Réduire le nombre de messages en utilisant des estampilles à la place de nonces**
- **Nécessite la synchronisation des horloges de A et B**
  - M1: a->b : I'm A . {a.ta}Kab
  - M2 : b->a : {b.ta}Kab

Authentification forte par défi réponse
Authentification mutuelle : clé publique

- **Version 1**
  - M1: a->b : I'm A . {na}PKb
  - M2: b->a : na.{nb}PKa
  - M3: a->b : nb

- **Version 2**
  - M1: a->b : I'm A . na
  - M2: b->a : nb . {na}SKb
  - M3: a->b : {nb}Ska

- **Problèmes:**
  - Comment A connaît la clé publique de B ?
    - ✓ Voir PKI

AVISPA: Automated Validation of Internet Security Protocols and Applications

HLPSL: High Level Protocol Specification Language

Contexte

- **Environnements de plus en plus distribués**
- **Nécessité de développer des protocoles de communication**
- **Un problème majeur dans la conception de protocoles est les erreurs de sécurité :**
  - Coût : une erreur de sécurité peut causer des millions $ de perte
  - Temps: le déploiement de protocoles est retardé
  - Confiance : difficulté d'attirer la confiance des clients potentiels des applications développées

Motivation

- **Le nombre et l'ampleur des protocoles de sécurité en développement dépasse les capacités humaines à les valider et les analyser rigoureusement**

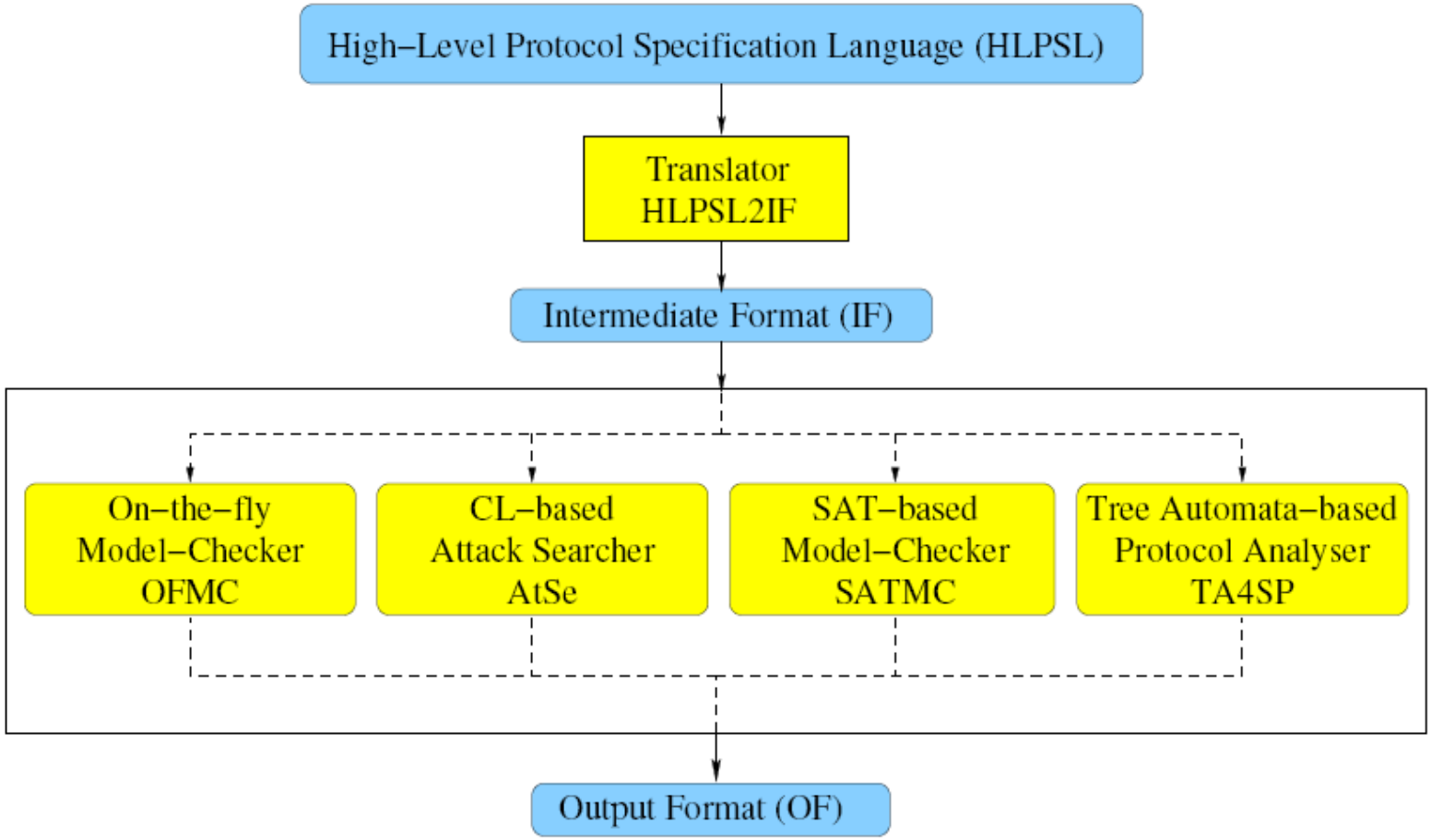
- **Pour accélérer le développement de nouveau protocoles de sécurité et améliorer leur niveau de sécurité, il est nécessaire de disposer de:**
  - Outils qui supportent l'analyse rigoureuse des protocoles cryptographiques, en
  - établissant leur correction, ou en démontrant leur failles

- **Ces outils doivent être complètement automatisés, robustes, expressifs, et facilement utilisable.**

Objectifs du projet AVISPA

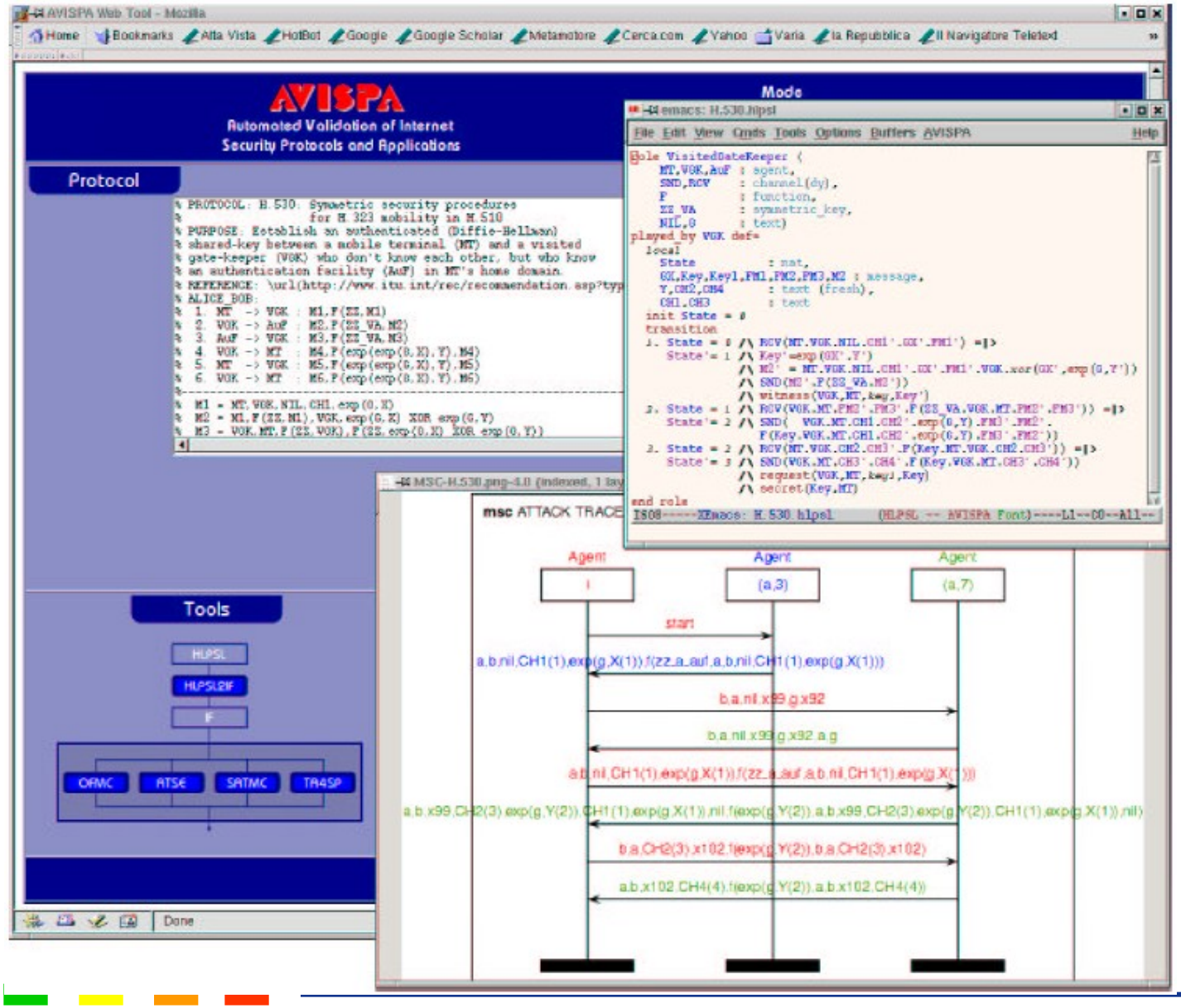
- **Définition d'un langage de haut niveau pour la spécification de protocoles cryptographiques: HLPSL**
- **Développement de techniques d'analyse de protocoles cryptographiques**
- **Développer un outil (AVISPA) en se basant sur ces techniques**
- **Développement d'une bibliothèque de spécifications de protocoles avec HLPSL, et les analyser avec l'outil AVISPA**

L'outil AVISPA



AVISPA Tool: the Back-ends

- The On-the-fly Model-Checker (OFMC)
  - **employs several symbolic techniques to explore the state space in a demand-driven way.**
- CL-AtSe (Constraint-Logic-based Attack Searcher)
  - **applies constraint solving with simplification heuristics and redundancy elimination techniques.**
- The SAT-based Model-Checker (SATMC)
  - **builds a propositional formula encoding all the possible attacks (of bounded length) on the protocol and feeds the result to a state-of-the-art SAT solver.**
- TA4SP (Tree Automata based on Automatic Approximations for the Analysis of Security Protocols)
  - **approximates the intruder knowledge by using regular tree languages and rewriting to produce under and over approximations.**


AVISPA Web Tool - Mozilla
AVISPA
Automated Validation of Internet
Security Protocols and Applications
Protocol
Mode
emacs: H.530.hlpsl
File Edit View Cmds Tools Options Buffers AVISPA Help
role VisitedDateKeeper (
MT,VGK,AuF : agent,
SND,RCV : channel(dy),
P : function,
ZZ_VA : symmetric_key,
NIL,0 : text)
played_by VGK def=
local
State : nat,
GX,Key,Key1,FM1,FM2,FM3,M2 : message,
Y,CH2,CH4 : text (fresh),
CH1,CH3 : text
init State = 0
transition
1. State = 0 /\ RCV(MT.VGK.NIL.CH1'.GX'.FM1') =|>
State' = 1 /\ Key'=exp(GX',Y')
/\ M2' = MT.VGK.NIL.CH1'.GX'.FM1'.VGK.xor(GX',exp(0,Y'))
/\ SND(M2'.P(ZZ_VA.M2'))
/\ witness(VGK,MT,key,Key')
2. State = 1 /\ RCV(VGK.MT.FM2'.FM3'.P(ZZ_VA.VGK.MT.FM2'.FM3')) =|>
State' = 2 /\ SND(VGK.MT.CH1.CH2'.exp(0,Y).FM3'.FM2'.
P(Key.VGK.MT.CH1.CH2'.exp(0,Y).FM3'.FM2'))
3. State = 2 /\ RCV(MT.VGK.CH2.CH3'.P(Key.MT.VGK.CH2.CH3')) =|>
State' = 3 /\ SND(VGK.MT.CH3'.CH4'.P(Key.VGK.MT.CH3'.CH4'))
/\ request(VGK,MT,key1,Key)
/\ secret(Key,MT)
end role
I808-----XEmacs: H.530.hlpsl (HLPSL -- AVISPA Font)----L1--C0--All--
Tools
HLPSL
HLPSL2IF
IF
OFMC
ATSE
SATMC
TA4SP
msc ATTACK TRACE
Agent
i
Agent
(a,3)
Agent
(a,7)
start
a.b.nil.CH1(1).exp(g,X(1)).f(zz_a_auf.a.b.nil.CH1(1).exp(g,X(1)))
b.a.nil.x99.g.x92
b.a.nil.x99.g.x92.a.g
a.b.nil.CH1(1).exp(g,X(1)).f(zz_a_auf.a.b.nil.CH1(1).exp(g,X(1)))
a.b.x99.CH2(3).exp(g,Y(2)).CH1(1).exp(g,X(1)).nil.f(exp(g,Y(2)).a.b.x99.CH2(3).exp(g,Y(2)).CH1(1).exp(g,X(1)).nil)
b.a.CH2(3).x102.f(exp(g,Y(2)).b.a.CH2(3).x102)
a.b.x102.CH4(4).f(exp(g,Y(2)).a.b.x102.CH4(4))
Done

HLPSL

- **Langage à base de rôles**
  - Un rôle pour chaque agent honnête
  - Les composition séquentielles et parallèles relient les différents rôles
- **Supporte les clés symétriques, asymétriques, fonction de hachage, fonctions algébriques, données typées et non typées, etc.**
- **L'intrus est modélisé par les canaux de communication**
  - Dolev-Yao (dy), et autres modèles,
- **Propriétés de sécurité : différentes formes d'authentification et de confidentialité**

Exemple

- NSPK Key Server Protocol:

$$
\begin{aligned}
&\textit{if } A \textit{ does not know } K_B, \\
&\qquad A \rightarrow S: \quad A, B \\
&\qquad S \rightarrow A: \quad \{B, K_B\}_{K_S^{-1}} \\
&A \rightarrow B: \quad \{N_A, A\}_{K_B} \\
&\textit{if } B \textit{ does not know } K_A, \\
&\qquad B \rightarrow S: \quad B, A \\
&\qquad S \rightarrow B: \quad \{A, K_A\}_{K_S^{-1}} \\
&B \rightarrow A: \quad \{N_A, N_B\}_{K_A} \\
&A \rightarrow B: \quad \{N_B\}_{K_B}
\end{aligned}
$$

- Structures de données complexes : liste de clés
- Contrôle de flux non supporté par d'autres outils de validation

Spécification modulaire à base de rôles

- **Rôles basiques**
  - Alice (initiateur)
  - Bob (répondeur)
  - Un serveur central
- **Rôles composés**
  - Définition d'une session : une Alice et un Bob
  - Instanciations de sessions : un serveur, et plusieurs sessions
- **Chaque rôle a un environnement local**

Entête d'un rôle basique Bob

```
role bob(A,B: agent,
         Kb,Ks: public_key,
         KeyRing: (agent.public_key) set,
         SND,RCV: channel(dy))
   played_by B def=
   local
      State: nat,
      Na,Nb: text,
      Ka: public_key
   init
      State:=1
   transition
      ...
end role
```

Transitions de Bob

```
1a.   State  =1 /\ RCV({Na'.A}_Kb) /\ in(A.Ka',KeyRing)
 =|> State':=3 /\ Nb':=new() /\ SND({Na'.Nb'}_Ka')

1b.   State  =1 /\ RCV({Na'.A}_Kb) /\ not(in(A.Ka',KeyRing))
 =|> State':=2 /\ SND(B.A)

2.    State  =2 /\ RCV({A.Ka'}_inv(Ks))
 =|> State':=3 /\ Nb':=new() /\ SND({Na.Nb'}_Ka')
                /\ KeyRing':=cons(A.Ka',KeyRing)

3.    State  =3 /\ RCV({Nb}_inv(Kb))
 =|> State':=4
```

Rôles composés

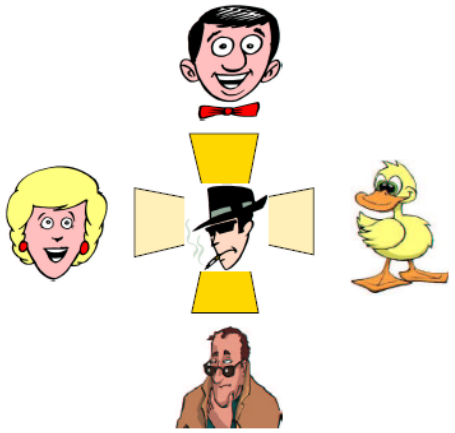
```
role session(A,B: agent,
             Ka,Kb,Ks: public_key,
             KeyRings: agent -> (agent.public_key) set) def=

   local SND,RCV: channel(dy)

   composition
         alice(A,B,Ka,Ks,KeyRings(A),SND,RCV)
      /\   bob(A,B,Kb,Ks,KeyRings(B),SND,RCV)

end role
```

Environnement global

```
role environment() def=
   local KeyRings: agent -> (agent.public_key) set,
         KeyRing: (agent.public_key) set,
         SND,RCV: channel(dy)
   const a,b,s,i: agent,
         ka,kb,ks,ki: public_key
   init  KeyRings:={(a.{}), (b.{a.ka}), (i.{a.ka,b.kb})}
      /\ KeyRing :={a.ka,b.kb,s.ks,i.ki}
   intruder_knowledge={a,b,s,i,ka,kb,ks,ki,inv(ki)}
   composition
         server(s,ks,KeyRing,SND,RCV)
      /\ session(a,b,ka,kb,ks,KeyRings)
      /\ session(i,b,ki,kb,ks,KeyRings)
      /\ session(a,i,ka,ki,ks,KeyRings)
end role
```

Le modèle d'intrus Dolev-Yao

- **L'intrus a le contrôle complet du réseau, c'est le réseau**
- **Tous les messages passent par l'intrus**
- **Les opérations de l'intrus sur les messages sont:**
  - Transfert, rejeu, suppression
  - Décomposition et analyse (si clés connues)
  - Modifier synthétiser
  - Envoyer à n'importe qui



- **L'intrus ne peut pas avoir accès aux messages chiffrés sans connaissance des clés**
- **L'intrus peut jouer le rôle de n'importe quel participant**
- **L'intrus récupère les connaissances d'un participant compromis**

Le système de typage HLPSL

- Types de base pour la spécification de protocoles:
  - **Agant, channel, boolean, integer, text, message, public key, symmetric key**
  - **On peut affecter une valeur fraîche à une variable en utilisant la fonction new()**

- Types constructeurs
  - **Function, tuples, sets**

- Types composés comme {text.bool}_public_key
  - **Décrit la structure des termes**
  - **Facilite l'optimisation de l'espace de recherche**

Déclaration des objectifs de sécurité (Goals)

- **Trois propriétés peuvent être considérées**
  - Confidentialité
  - Authentification faible
  - Authentification forte (avec protection contre le rejeu)

```
goal
    secrecy_of na, nb
    authentication_on alice_bob_nb
    authentication_on bob_alice_na
end goal
```

Spécification des « goal facts »

```
role bob...
   1a. State  =1 /\ RCV({Na'.A}_Kb) /\ in(A.Ka',KeyRing)
   =|> State':=3 /\ Nb':=new() /\ SND({Na'.Nb'}_Ka')
       /\ secret(Nb',nb,{A,B})
       /\ witness(B,A,alice_bob_nb,Nb')
       ...
end role

role alice...
   3.  State  =3 /\ RCV({Na.Nb'}_Ka)
   =|> State':=4 /\ SND({Nb'}_Kb)
       /\ request(A,B,alice_bob_nb,Nb')
end role
```

Propriétés de HLPSL

- Facile à apprendre et lire
- Sémantique non ambiguë
- Fortement typé
- Expressif, supportant:
  - ✓ **Modularité: composition,…**
  - ✓ **Contrôle du flux**
  - ✓ **Connaissance explicite de l'intrus**
  - ✓ **Primitive cryptographique: nonces, haches, signatures, ..**
  - ✓ **Propriétés algébriques: comme xor, et exp**