Mémoire de stage de DEA
effectué au LSV :
Laboratoire Spécification et Vérification
Ecole Normale Supérieure de Cachan
61, avenue du Président Wilson
94235 CACHAN Cedex.

Vérification de protocoles de sécurité dans un modèle de l'intrus étendu

Mémoire présenté par Stéphanie Delaune
Sous la direction de
Hubert Comon-Lundh et Florent Jacquemard

Année 2003

Table des matières

Chapitre 1

Présentation

1.1 Présentation du laboratoire

Le Laboratoire Spécification et Vérification (LSV) est le laboratoire de recherche en informatique de l'École Normale Supérieure de Cachan (ENS de Cachan). Il a été créé en octobre 1996 et est associé au Centre Nationale de Recherche Scientifique (CNRS).

Le LSV est un laboratoire reconnu de recherche en informatique, sur une thématique clairement définie et de grande importance pour le monde industriel : l'informatique de la vérification, c'est à dire l'ensemble des méthodes formelles permettant d'accroître la confiance dans les logiciels critiques.

La notion de logiciel critique, dont les défaillances peuvent avoir des conséquences désastreuses en termes humains ou économiques, n'est plus cantonnée à des secteurs d'activité spécifiques et délimités (transports, télécommunications, énergie). Aujourd'hui, le logiciel est souvent le maillon faible de bien d'autres secteurs d'activité, comme le commerce électronique, l'industrie des loisirs, la santé ... C'est pourquoi jamais les enjeux scientifiques, techniques et économiques de l'informatique de la vérification n'ont été aussi cruciaux.

Une des forces du LSV est de rassembler des chercheurs travaillant tous dans le domaine de la vérification, mais chacun avec des approches et des techniques spécifiques. Aussi, pour exploiter de façon optimale cette diversité, le LSV n'est pas structuré en équipes, mais la recherche est organisée autour de trois axes principaux :
- INFINI : Vérification symbolique des systèmes complexes
- SECSI : Sécurité des systèmes d'information
- Vérification de systèmes temporisés

Les travaux des membres du LSV se répartissent naturellement entre les différents axes et sont menés en partenariat étroit avec des industriels qui expriment des besoins en matière de vérification de logiciels, comme Électricité de France, France Télécom, ...

Ce stage porte sur la vérification des protocoles cryptographiques et trouve donc sa place dans l'axe de recherche SECSI.

1.2 Présentation de l'axe de recherche : SECSI

Le LSV a développé depuis trois ans un axe de recherche sur la sécurité des systèmes informatiques auquel participent environ la moitié de l'effectif de recherche (soit une quinzaine de personnes).

Traditionnellement, le premier problème de sécurité est la confidentialité : certaines données doivent rester confinées à un cercle d'initiés. Vu sous un autre angle, il s'agit de contrôler l'accès à certaines ressources. Ce problème n'est pas nouveau et se pose depuis très longtemps dans les systèmes d'exploitation. Ce qui change essentiellement est qu'on s'intéreresse à ces problèmes dans un environnement distribué dont les canaux de communication ne sont pas fiables.

Au sein de cet axe, les recherches sont organisées autour de trois thèmes principaux :

- **Vérification automatique de protocoles cryptographiques**
  C'est l'activité dominante de cet axe : un professeur Hubert Comon-Lundh, un chargé de recherche F. Jacquemard, un maître de conférence R. Treinen et quatre doctorants travaillent à plein temps sur ce sujet, les autres partageant leur temps avec d'autres aspects de la sécurité. La première étape de la vérification d'un protocole cryptographique étant sa spécification formelle, nous nous intéressons à des modèles formels décrivant les protocoles et leurs propriétés de sécurité, ainsi qu'à une définition rigoureuse de la relation de satisfaction entre protocoles et propriétés. Puis, nous travaillons à la mise au point de techniques automatiques pour décider cette relation de satisfaction.

  Securify est un exemple d'outil permettant de vérifier automatiquement le secret pour les protocoles cryptographiques. Il a été développé au LSV dans le cadre du projet RNTL EVA et l'étude de Securify m'a permis de contribuer à son amélioration. D'autre part, ce stage a permis d'étudier un aspect de l'affaiblissement de l'*hypothèse dite du chiffrement parfait* et constitue donc une première étape dans la réalisation du projet RNTL PROUVÉ, qui débute en septembre 2003, en partenariat, notamment avec France Télécom R&D

  France Télécom R&D, centre de recherche de France Télécom, travaille depuis une vingtaine d'années sur les techniques formelles (spécification, preuve et test) tant sur le plan théorique que pratique. Les résultats présentés dans ce mémoire devraient permettre de contribuer au développement de techniques pertinentes par rapport aux problèmes soulevés en pratique et ont fait l'objet d'un exposé au sein du laboratoire TAL (Technique Avancées pour la construction de Logiciels) de FTR&D.
- **Détection d'intrusions**
  Le travail porte sur la réalisation d'outils automatiques permettant de détecter en temps réel des intrusions dans des systèmes informatiques.
- **Analyse statique de programmes**
  Il s'agit ici de l'analyse des implémentations concrètes des protocoles sécuritaires. L'analyse d'un code est beaucoup plus délicate que la simple vérification de protocoles cryptographiques. La raison principale est que la granularité est différente : ce qui est vu au

niveau abstrait comme un seul pas dans un protocole cryptographique devient au niveau code une suite de plusieurs instructions qui manipulent des structures de données plus primitives.

Chapitre 2

Introduction

2.1 Enjeux

La communication par des canaux publics (par exemple internet) s'est beaucoup développée ces dernières années. Les transactions utilisant ce médium sont de plus en plus nombreuses : communication client-fournisseur, services audiovisuels (vidéo à la demande), services bancaires, porte-monnaie électronique, protocoles d'enchères, vote électronique... . Les aspects sécuritaires occupent une place importante dans toutes ces applications. Pour assurer ces propriétés de sécurité, des moyens algorithmiques, tels que les chiffrements et les fonctions à sens unique ont été mis au point ; ils permettent d'assurer certaines propriétés, par exemple qu'il est très improbable qu'un individu puisse obtenir un message en clair à partir d'un chiffré sans connaître la clef de déchiffrement.

Cependant, même si ces moyens algorithmiques remplissent parfaitement leur spécification, les propriétés sécuritaires ne sont pas pour autant toujours satisfaites. Les communications "sécurisées" sont en effet assurées par des *protocoles* dits *cryptographiques* qui utilisent ces moyens algorithmiques mais sont constitués de plusieurs messages. Plusieurs exemples célèbres ont montré qu'ils peuvent être attaqués ("man in the middle attack", "replay attack",...) même en présence d'un chiffrement parfait. (Voir l'article de synthèse [9] : presque tous les protocoles d'authentification comportent des failles).

Les protocoles cryptographiques qui sont censés assurer des propriétés de sécurité ont ainsi des caractéristiques spécifiques :

- on les trouve en très grand nombre, souvent sous la forme de petites variantes d'un protocole connu,
- les failles de sécurité dans l'un de ces protocoles peuvent avoir des conséquences économiques catastrophiques, à cause de leur déploiement à grande échelle,
- leur description est souvent très courte, sous forme d'un échange d'une dizaine de messages tout au plus.

Il est donc crucial de pouvoir vérifier les propriétés de ces protocoles.

2.2 Contexte

Le problème de la vérification des protocoles cryptographiques n'est pas nouveau : il remonte aux années 1980. Cependant, d'une part son importance n'a fait que croître et d'autre part, ce n'est que depuis 1998 environ que des cadres formels (sémantique) ont été proposés pour effectuer ces vérifications, les travaux précédents se bornant à la recherche d'attaques.

2.2.1 Difficultés de la vérification

La vérification des protocoles cryptographiques est un cas particulier de model-checking où les systèmes considérés sont des protocoles cryptographiques dans un réseau hostile. Le problème de la vérification automatique des protocoles cryptographiques vient de la complexité du modèle :

1. le nombre de sessions d'un protocole (en parallèle ou en séquence) n'est pas borné,
2. les capacités mémoire d'un attaquant ne sont pas supposées bornées,
3. la taille des messages que peut fabriquer un attaquant n'est pas a priori bornée,
4. la plupart des protocoles utilisent la possibilité d'engendrer aléatoirement un nombre (*nonce*).

Même sans l'une des trois dernières hypothèses, le problème de confidentialité le plus simple est indécidable ([2, 17, 10] par exemple). Il est par contre remarquable qu'en fixant le nombre de sessions, le problème devient NP-complet [27]. Mais fixer le nombre de sessions à un nombre peu élevé ne correspond à aucune hypothèse réaliste ; le seul résultat permettant de borner (à 2) le nombre de sessions nécessaire à une attaque [22] concerne une classe de protocoles extrêmement restreinte. C'est pourquoi les démonstrateurs automatiques procèdent soit par abstraction [6, 4], soit ne sont pas garantis de s'arrêter [15, 8].

Quelle que soit la méthode employée, un problème algorithmique de base est de savoir si, étant donné un ensemble fini de messages T et un secret s, l'intrus peut déduire s à partir de T. Ce problème appelée *problème de déduction de l'intrus* correspond à un intrus *passif* qui récupère un ensemble de messages T en écoutant ce qui circule sur le réseau, et qui utilise son pouvoir de déduction pour essayer de dériver le secret s.

2.2.2 Outils consacrés à la preuve

Plusieurs outils de démonstration automatique ont été mis au point ces dernières années. Ils se sont développés un peu partout en France, en Europe et dans le monde. Citons tout particulièrement les outils de preuve automatique CASPER [20], CASPL [24], CASRUL [3], EVA [5, 13].

Mais ces outils comportent de nombreuses faiblesses. En effet, les démonstrateurs existants se concentrent sur un unique modèle d'attaquant (dit de *Dolev-Yao*) et une seule propriété de sécurité : le secret. Autrement dit, tous les modèles de protocoles cryptographiques utilisés par les démonstrateurs font l'*hypothèse du chiffrement parfait* : sans avoir la clef de déchiffrement, on ne peut obtenir aucune information sur un texte chiffré. Cette hypothèse est excessive dans beaucoup de cas et ne correspond pas toujours à la réalité.

2.2.3 Hypothèse du chiffrement parfait

Cette hypothèse simplificatrice a été prise à cause de la compléxité des problèmes de vérification. Cependant les algorithmes de chiffrement et les autres primitives cryptographiques possèdent des propriétés algébriques qui sont parfois exploitées par les protocoles eux-mêmes. Certains protocoles sont alors sûrs si l'on fait l'*hypothèse du chiffrement parfait*, mais vulnérables dans la réalité, les attaques s'appuyant sur les propriétés algébriques [11].

Certains travaux théoriques récents ont étudié l'ajout d'une théorie équationnelle au modèle d'intrus standard dit de *Dolev-Yao* afin de prendre en compte certaines propriétés algébriques des primitives cryptographiques [11, 7, 12]. Mais ces travaux sont insuffisants pour prendre en compte certaines (autres) faiblesses du chiffrement.

Donnons ici deux exemples de protocoles pour lesquels l'*hypothèse du chiffrement parfait* est excessive. Les attaques ne s'appuient pas ici sur les propriétés algébriques des primitives cryptographiques, mais sur le fait que certains types de données, dits *faibles* prennent leurs valeurs dans un domaine fini connu de l'intrus.

Exemple 2.1
Considérons le protocole (naïf) de vote suivant :

$$A \;\rightarrow S : \;\; \{m\}_{pub(S)}$$

L'agent A chiffre son vote m avec la clef publique du serveur S. Lorsque le serveur reçoit le message chiffré, il le déchiffre avec sa clef privée $priv(S)$. On demande que m soit un secret partagé uniquement entre A et S.

Le message $\{m\}_{pub(S)}$ qui circule sur des canaux de communication non fiables est connu de tous, mais seul S peut récupérer le vote de A, car il est le seul à connaître la clef de déchiffrement. Cependant, m est le nom d'un candidat parmi une liste de candidats que l'on peut supposer connue de l'intrus. Dans ce cas, l'intrus peut construire les messages de la forme $\{m'\}_{pub(S)}$ où m' est le nom d'un candidat, et par simple comparaison avec le message intercepté en déduire le vote de A.

Dans le modèle standard de *Dolev-Yao*, ce protocole est donc sûr puisque m reste secret, mais il existe une attaque très simple dans la réalité, ce qui met en évidence un des points faibles du modèle.

Exemple 2.2
Considérons le challenge suivant :

$$\begin{aligned} A \;&\rightarrow B : \;\; \{n\}_k \\ B \;&\rightarrow A : \;\; \{n+1\}_k \end{aligned}$$

L'agent A engendre une donnée n aléatoire, appelée *nonce*, la chiffre avec la clef symétrique k qu'il partage avec l'agent B. Lorsque B reçoit $\{n\}_k$, il le déchiffre, calcule $n+1$, chiffre le résultat avec la clef k et envoie le message à A. Comme k n'est connue que de A et de B, la réception par A de $\{n+1\}_k$ est un certificat que B a bien reçu $\{n\}_k$. Donc, par exemple, n pourrait être utilisé par la suite comme nouvelle clef de chiffrement. Un tel échange (challenge) est utilisé dans de nombreux protocoles cryptographiques comme Needham-Schroeder à clef symétrique [26], Denning-Sacco modifié par Lowe [21] ...

De ces deux messages, et sous l'*hypothèse du chiffrement parfait* un intrus ne peut déduire ni n, ni k. En revanche, si l'on suppose que la clef k partagée entre les agents A et B a été dérivée à partir d'un mot de passe faible (typiquement un mot du dictionnaire). L'intrus dispose alors d'une information supplémentaire qu'il peut exploiter pour déduire n et k. Il lui suffit de déchiffrer les deux messages interceptés avec toutes les valeurs possibles pour la clef k, d'incrémenter la première et de comparer les deux résultats. Si les deux messages obtenus sont identiques, l'intrus a alors «deviné» la valeur de k et de n.

On voit encore sur ce deuxième exemple que le modèle de *Dolev-Yao* est insuffisant pour rendre compte du fait que certaines données prennent leurs valeurs dans un ensemble fini connu de l'intrus. L'objet de ce mémoire est d'étendre le modèle de *Dolev-Yao* pour prendre en compte les données de type faible.

L'étude de ces protocoles est une première étape dans la réalisation du projet RNTL PROUVÉ. Elle va permettre de prendre en compte ces données de type faible et contribuer ainsi à l'affaiblissement de l'*hypothèse du chiffrement parfait* qui est un des objectifs majeurs de ce projet.

2.3 Contenu et plan du mémoire

Ce mémoire se présente en deux parties. D'une part, la formalisation et l'étude d'un nouveau modèle d'intrus (chapitre 3), et d'autre part une présentation de résultats utiles pour la vérification pratique des protocoles (chapitre 4).

Sur le plan de la modélisation, nous introduisons un nouveau modèle d'attaquant (section 3.3) pour prendre en compte la capacité de l'intrus à deviner la valeur d'une donnée faible et à vérifier son choix ensuite. Ce nouveau modèle est en fait une extension du modèle standard dit de *Dolev-Yao* par l'ajout, en particulier, d'une règle Comparaison qui permet de comparer des données obtenues de deux manières différentes et d'en déduire qu'une valeur devinée est correcte.

Dans le modèle de *Dolev-Yao*, les techniques de recherche de preuves ou d'attaques s'appuient en général sur un résultat (qui est une conséquence de [1]) : le *problème de déduction de l'intrus* est complet pour PTIME (section 3.2). Notre premier résultat consiste à généraliser ce théorème au modèle étendu incluant la règle de comparaison : le *problème de déduction de l'intrus* est encore PTIME complet (section 3.4).

Sur le plan de la vérification pratique des protocoles cryptographiques, un des objectifs du projet RNTL PROUVÉ est de développer les différents modèles d'intrus qui seront apparus pertinents compte tenu des études de cas traités. L'objectif étant la vérification automatique des protocoles, il faut que l'algorithme de recherche utilisé pour la vérification (que celui-ci soit une recherche de preuve ou d'attaque) soit modulaire, c'est à dire paramétré par la théorie de l'intrus.

Dans le cadre du projet RNTL EVA, plusieurs outils prouvant le secret ont été implémentés, dont Securify développé au LSV par Véronique Cortier (section 4.1). Cet outil [13] cherche à montrer inductivement que l'application d'une règle du protocole ne compromet aucun secret. Pour ce faire trois conditions suffisantes ont été mis en évidence et une procédure de recherche en arrière (section 4.1.2). Cet outil permet de vérifier automatiquement les protocoles cryptographiques, mais il comporte (comme les démonstrateurs existants actuellement) de nombreuses faiblesses. En particulier, il se concentre sur un seul modèle d'attaquant, celui de *Dolev-Yao*.

L'étude de l'outil Securify m'a permis d'apporter quelques améliorations (section 4.1.3) qui se sont traduites par l'ajout de nouveaux tests élémentaires, permettant à l'outil de conclure au secret plus souvent et d'éviter un certain nombre d'échecs qui étaient dû à de fausses attaques. Cependant, il est apparu que dans cet outil le modèle d'intrus et l'algorithme de recherche étaient intimement liés et ne pouvaient pas être dissociés facilement.

Ce travail a donc donné lieu à la réalisation d'un prototype (section 4.2) : une implémentation simple, mais modulaire d'un algorithme de recherche d'attaque. L'algorithme est paramétré par un module représentant la théorie de l'intrus et permet donc la recherche d'attaque pour différents types d'intrus. Les théories de l'intrus qui ont été implémentées sont celles correspondant au modèle de *Dolev-Yao* standard et au modèle de *Dolev-Yao* étendu par la règle de comparaison aux données de type faible. Cette procédure a été implémentée en *Ocaml*, puis testée sur plusieurs protocoles. Nous exposons les résultats obtenus en section 4.2.4.

Chapitre 3

Extension du modèle de Dolev-Yao

Bien que la vérification automatique de protocoles cryptographiques soit indécidable même avec certaines restrictions [10, 17], elle a reçu beaucoup d'attention ces dernières années. En particulier, le *problème de déduction de l'intrus* qui correspond au problème du secret en présence d'un attaquant passif est une question importante tant pour la vérification des protocoles que pour la recherche d'attaques.

Nous allons présenter un résultat de décidabilité dans le cadre de la vérification de protocoles cryptographiques en présence de données prenant leurs valeurs dans un domaine fini connu de l'intrus. Puisque l'*hypothèse du chiffrement parfait* n'est pas réaliste pour des protocoles qui utilisent des données faibles, nous allons étendre le modèle de *Dolev-Yao* pour prendre en compte les « attaques par prédictions », lorqu'un intrus devine la valeur d'une donnée faible et vérifie ensuite la pertinence de son choix. Nous allons montrer que le *problème de déduction de l'intrus* reste décidable en temps polynomial pour ce nouveau modèle d'attaquant.

Comme dans les cas des travaux qui étendent le modèle d'intrus par une théorie équationnelle, [11, 7, 12], la difficulté réside dans le résultat de décidabilité et de complexité du *problème de déduction de l'intrus*. Cependant une difficulté supplémentaire est de trouver un modèle convaincant pour modéliser le pouvoir de l'attaquant afin de prendre en compte les « attaques par prédictions » induites par l'utilisation de données de type faible. Le modèle présenté sera illustré par de nombreux exemples afin de justifier les différents choix.

3.1 Première approche, des exemples

Nous commençons ici par présenter un certain nombre d'exemples pour mettre en évidence l'insuffisance des modèles existants pour rendre compte des attaques en présence de données faibles.

Exemple 3.1
Reprenons l'exemple donné en introduction :

$$A \ \to S : \ \{m\}_{pub(S)}$$

L'agent A chiffre son vote m avec la clef publique du serveur S. Lorsque le serveur reçoit le message chiffré, il le déchiffre avec sa clef privée $priv(S)$. On demande que m soit un secret partagé uniquement entre A et S.

Le message $\{m\}_{pub(S)}$ qui circule sur des canaux de communication non fiables est connu de tous, mais seul S peut récupérer le vote de A, car il est le seul à connaître la clef de déchiffrement, $priv(S)$. Cependant, m est le nom d'un candidat parmi une liste de candidats que l'on peut supposer connue de l'intrus. Dans ce cas, l'intrus peut construire les messages de la forme $\{m'\}_{pub(S)}$ où m' est le nom d'un candidat, et par simple comparaison avec le message intercepté en déduire le vote de A.

Exemple 3.2
Considérons l´échange suivant (dit challenge-réponse) qui est utilisé dans de nombreux protocoles : Needham Schroeder à clefs symétriques (78) [26], Denning-Sacco, Wide-Mouthed Frog modifiés par Lowe (97) [21].

$$\begin{array}{lll} A & \to B : & \{n\}_k \\ B & \to A : & \{n+1\}_k \end{array}$$

A engendre un nombre aléatoirement, *un nonce* n, et le chiffre avec une clef symétrique k partagée entre A et B. Lorsque B reçoit le message, il le déchiffre, calcule $n+1$ et retourne le résultat chiffré avec k, pour prouver qu'il possède la clef k. Dans le modèle de *Dolev-Yao* standard, un intrus ne peut déduire (à partir de ces deux messages) ni k, ni n.

Cependant, si on suppose que la clef k utilisée pour le chiffrement a été dérivée à partir d'un mot de passe faible, autrement dit, si on suppose que k est une donnée faible, l'intrus va pouvoir utiliser cette hypothèse supplémentaire pour en déduire k et n.

Il lui suffit en effet de déchiffrer les deux messages interceptés avec une valeur possible pour la clef k, et d'ajouter 1 à la première de ces deux valeurs. Si les valeurs obtenues sont identiques, c'est que l'attaquant a utilisé la « bonne » clef pour le déchiffrement. Autrement dit, il a trouvé la valeur de la clef k et donc de n.

Exemple 3.3
Considérons un dernier exemple, dans lequel nous supposerons que la clef privée de l'agent A est faible :

Puisque l'intrus connaît la clef publique $pub(A)$, il peut choisir une valeur pour $priv(A)$ - parmi toutes les valeurs possibles - et vérifier si son choix est correct en chiffrant un message arbitraire et en le déchiffrant avec la clef publique de A.

Si le message obtenu après ces différentes opérations est le même qu'initialement, c'est que l'intrus avait fait le bon choix pour le clef privée de A.

Ces exemples ont pour but de mettre en évidence l'intérêt de considérer les données à valeurs dans un domaine fini. Nous allons par la suite, formaliser par un système de déduction le pouvoir de ce nouveau type d'intrus.

3.2 Le problème de déduction de l'intrus dans le modèle de Dolev-Yao

Avant de nous intéresser au *problème de déduction de l'intrus* en présence de données faibles, nous allons rappeler le problème ainsi que les résultats obtenus dans le cadre du modèle de *Dolev-Yao* standard.

3.2.1 Préliminaires

Les messages sont représentés par des termes construits sur l'alphabet :

$$\mathcal{F}_{\mathsf{msg}} = \{< \_, \_ >, \{\_\}_{\_}, \_^{-1}\}$$

et à partir de constantes.

Le symbole $\langle \_, \_ \rangle$ est le symbole classique de la paire.

Le symbole $\{\_\}_{\_}$ représente le chiffrement : le terme $\{m_1\}_{m_2}$ représente le message m_1 chiffré avec la clef m_2. On peut noter ici que tout message peut servir de clef.

Le symbole $\_^{-1}$ appliqué à un terme représente le terme inverse.

Nous considérons les clefs symétriques et asymétriques.

Définition 3.1 *(clef symétrique)*
Une clef k est symétrique si $k^{-1} = k$ et nous supposerons que les clefs composées sont symétriques.

Nous considérons la théorie équationnelle engendrée par l'équation ${x^{-1}}^{-1} = x$. Si on oriente de gauche à droite cette équation, on obtient un système de réécriture convergent. Autrement dit, chaque terme t admet une unique forme normale notée $t\downarrow$.

Définition 3.2 *(ensemble des sous-termes)*
Soit T un ensemble fini de termes, on note $St(T)$ l'ensemble des sous-termes de T, i.e. l'ensemble des termes tel que :
- *si $t \in T$ alors $t \in St(T)$*
- *si $t^{-1} \in St(T)$ alors $t \in St(T)$*
- *si $\{u\}_v \in St(T)$ alors $u, v \in St(T)$*
- *si $\langle u, v\rangle \in St(T)$ alors $u, v \in St(T)$*

Le nombre d'éléments dans $St(T)$ est linéaire en la taille de T. La taille d'un ensemble de termes étant définie comme la somme des tailles de chacun des termes de l'ensemble.

3.2.2 Présentation du modèle de Dolev-Yao

Le *problème de déduction de l'intrus* dépend des capacités de l'intrus à analyser et synthétiser des messages. Le modèle le plus répandu est le modèle de *Dolev-Yao* [16] défini à la figure 3.1.

L'intrus peut construire des paires et des messages chiffrés à partir de messages connus (règles P, E), décomposer des paires (règles UL, UR), déchiffrer des messages seulement lorsqu'il connaît la clef de déchiffrement (règle D). Sans avoir la clef de déchiffrement, on ne peut obtenir aucune information sur un texte chiffré : ce modèle repose donc sur l'*hypothèse dite du chiffrement parfait.*

3.2.3 Présentation du problème de déduction de l'intrus

Le problème de déduction de l'intrus est le suivant :

Etant donné un ensemble fini de messages clos T et un secret s, l'intrus peut-il déduire le terme s à partir des messages contenus dans T ?

Axiome (A)	$\dfrac{u \in T}{T \vdash u}$	Paire (P)	$\dfrac{T \vdash u \quad T \vdash v}{T \vdash \langle u, v \rangle}$
Projection (UL)	$\dfrac{T \vdash \langle u, v \rangle}{T \vdash u}$	Projection (UR)	$\dfrac{T \vdash \langle u, v \rangle}{T \vdash v}$
Chiffrement (E)	$\dfrac{T \vdash u \quad T \vdash v}{T \vdash \{u\}_v}$	Déchiffrement (D)	$\dfrac{T \vdash \{u\}_v \quad T \vdash v^{-1}}{T \vdash u}$

FIGURE 3.1 - *Le modèle de Dolev-Yao.*

Par la suite, nous noterons $T \overset{?}{\vdash} s$ le *problème de déduction de l'intrus.*

Définition 3.3 *(preuve de $T \vdash u$)*
Une preuve de $T \vdash u$ est un arbre étiqueté avec des séquents $T \vdash v$ et tels que :
- *chaque feuille de l'arbre est étiquetée avec un terme $v \in T$.*
- *chaque nœud étiqueté par $T \vdash v$ a n fils $T \vdash s_1$, ..., $T \vdash s_n$ tels que $\dfrac{T \vdash s_1 \ \dots \ T \vdash s_n}{T \vdash v}$ soit une instance d'une des règles d'inférence de la figure 3.1*
- *la racine de l'arbre est étiquetée par $T \vdash u$*

La taille d'une preuve est le nombre de ses noeuds.

Autrement dit, le *problème de déduction de l'intrus* consiste à se demander s'il existe une preuve de $T \vdash s$.

3.2.4 Rappel des résultats

Théorème 3.1
Le problème de déduction de l'intrus, $T \overset{?}{\vdash} s$ est décidable en temps polynomial dans le modèle de Dolev-Yao.

Ce résultat peut être obtenu en appliquant le théorème de D. McAllester [1] puisque la théorie donnée par les règles de la figure 3.1 définit une théorie locale.

Définition 3.4 *(théorie locale)*
Une théorie donnée par un système de règles d'inférence définit une théorie locale si lorsque $T \vdash s$ est dérivable dans le système, alors il existe une preuve qui ne fait intervenir que les termes et sous-termes de T et de s.

Par la suite, nous allons considérer la même question mais pour un nouveau modèle d'intrus.

3.3 Le problème de déduction de l'intrus dans le modèle de Dolev-Yao étendu

Nous allons introduire un nouveau modèle pour représenter le pouvoir de déduction de l'intrus et nous allons prouver que le *problème de déduction de l'intrus* reste décidable en temps polynomial dans ce modèle étendu.

3.3.1 Présentation du «nouveau» problème de déduction de l'intrus

La présence de données faibles peut permettre à l'intrus de réaliser des « attaques par prédictions ». Nous utilisons la définition donnée par Lowe dans [23] qui généralise celle de Gong [18].

Une « attaque par prédictions » consiste à deviner la valeur d'une donnée faible g, et à vérifier si ce choix s'avère correct. La vérification faite par l'intrus consiste à produire un vérifieur v à partir de g et peut se faire de différentes façons :

1. l'intrus connaît v initialement. (cf. exemple 3.1)
2. l'intrus produit v de deux façons différentes à partir de g. (cf. exemple 3.2)
3. v est une clef asymétrique et l'intrus connaît son inverse. (cf. exemple 3.3)

Pour formaliser le *problème de déduction de l'intrus* en présence de données faibles, nous devons distinguer dans la connaissance de l'intrus, les messages « fortement connus » et les messages « faiblement connus ».

Intuitivement, les messages « fortement connus » sont ceux que l'intrus connaît exactement. C'est le cas par exemple des messages interceptés, des clefs publiques des agents ...

Les messages « faiblement connus » sont ceux qui prennent leur valeur dans un domaine fini connu de l'intrus. Si l'intrus dispose de suffisament d'informations, il peut alors deviner la valeur d'une donnée faible et vérifier sa prédiction afin de savoir si celle-ci est correcte. Si la vérification s'avère correcte, on peut supposer que le message est désormais « fortement connu » par l'intrus.

Définition 3.5 *(terme atomique)*
Un terme est dit atomique si c'est une constante , ou l'inverse k^{-1} d'une clef k.

Nous pouvons reformuler le *problème de déduction de l'intrus* de la façon suivante :

Etant donné un ensemble fini de messages fortement connus T, un ensemble fini de données atomiques faiblement connues T' et un secret s, l'intrus peut-il déduire s à partir des messages de T et de T' ?

Nous noterons $T/T' \overset{?}{\vdash} s$ le *problème de déduction de l'intrus* en présence de données faibles.

3.3.2 Présentation du nouveau modèle

Dans cette section, nous décrivons comment modéliser les « attaques par prédictions » en adaptant le modèle standard de *Dolev-Yao*.

Ce nouveau modèle, appelé modèle de *Dolev-Yao* étendu, est présenté à la figure 3.2. Il fait intervenir deux formes deux séquents :

Axiome (A) $\dfrac{u \in T}{T/\emptyset \vdash u}$ Paire (P) $\dfrac{T/T'_1 \vdash u \quad T/T'_2 \vdash v}{T/T'_1 \cup T'_2 \vdash \langle u, v\rangle}$

Projection (UL) $\dfrac{T/T' \vdash \langle u, v\rangle}{T/T' \vdash u}$ Projection (UR) $\dfrac{T/T' \vdash \langle u, v\rangle}{T/T' \vdash v}$

Chiffrement (E) $\dfrac{T/T'_1 \vdash u \quad T/T'_2 \vdash v}{T/T'_1 \cup T'_2 \vdash \{u\}_v}$ Déchiffrement (D) $\dfrac{T/T'_1 \vdash \{u\}_v \quad T/T'_2 \vdash v^{-1}}{T/T'_1 \cup T'_2 \vdash u}$

Axiome (A′) $\dfrac{u \text{ terme atomique}}{T/u \vdash' u}$ Paire (P′) $\dfrac{T/T'_1 \vdash' u \quad T/T'_2 \vdash' v}{T/T'_1 \cup T'_2 \vdash' \langle u, v\rangle}$

Projection (UL′) $\dfrac{T/T' \vdash' \langle u, v\rangle}{T/T' \vdash' u}$ Projection (UR′) $\dfrac{T/T' \vdash' \langle u, v\rangle}{T/T' \vdash' v}$

Chiffrement (E′) $\dfrac{T/T'_1 \vdash' u \quad T/T'_2 \vdash' v}{T/T'_1 \cup T'_2 \vdash' \{u\}_v}$ Déchiffrement (D′) $\dfrac{T/T'_1 \vdash' \{u\}_v \quad T/T'_2 \vdash' v^{-1}}{T/T'_1 \cup T'_2 \vdash' u}$

Affaiblissement (W) $\dfrac{u \in T}{T/\emptyset \vdash' u}$

Comparaison (C)

$$\frac{P_1 \left\{ \dfrac{\dfrac{\ldots}{\vdash' x_1} \cdots \dfrac{\ldots}{\vdash' x_n}}{T/T'_1 \vdash' u} \right. \text{(R1)} \qquad P_2 \left\{ \dfrac{\dfrac{\ldots}{\vdash' y_1} \cdots \dfrac{\ldots}{\vdash' y_n}}{T/T'_2 \vdash' v} \right. \text{(R2)}}{T/T'_1 \cup T'_2 \vdash w}$$

où :

(i). $w \in T'_1 \cup T'_2$

(ii). P_1 et P_2 sont des preuves en forme normale

(iii). R1 $\neq$ R2 or $\{u, x_1, .., x_n\} \neq \{v, y_1, ..., y_n\}$

(iv). $R(u, v)$ avec $R = Id \cup \{(k, k^{-1}) |$ k est une clef $\}$

FIGURE 3.2 - *Le modèle de Dolev-Yao étendu.*

- $T/T' \vdash u$ signifie que si l'intrus connaît fortement les messages de T et faiblement les messages atomiques de T', alors il peut déduire fortement le message u.
- $T/T' \vdash' u$ signifie que si l'intrus connaît fortement les messages de T et faiblement les messages atomiques de T', alors il peut déduire faiblement le message u. Autrement dit, il peut déduire que u appartient à un ensemble fini qu'il sait calculer.

Les règles (A, P, UL, UR, E, D) représentent la capacité de l'intrus à faire de la déduction forte à partir d'hypothèses fortes, alors que les règles (A', P', UL', UR', E', D') représentent la capacité de l'intrus à faire de la déduction faible à partir d'hypothèses faibles. La règle d'affaiblissement (W) exprime le fait que les messages « fortement connus » sont des cas particuliers de messages « faiblement connus ».

Enfin, la règle (C) qui mélange les deux formes de séquents est utilisée pour formaliser le fait qu'un intrus peut deviner la valeur d'une donnée faible quand :

1. l'intrus peut produire le vérifieur v de deux façons différentes, dans ce cas on a $u = v$ et la condition (iii) assure le fait que les deux preuves sont bien différentes.
2. l'intrus peut produire le vérifieur v qui est une clef asymétrique dont il connaît l'inverse v^{-1}.

Définition 3.6 *(preuve de $T/T' \vdash u$)*
Une preuve de $T/T' \vdash u$ est un arbre étiquetée avec des séquents de la forme $T/T' \diamond v$ et tels que :

- $\diamond \in \{\vdash, \vdash'\}$.
- *chaque feuille de l'arbre est étiquetée avec une instance des règles* A, A' *ou* W.*(figure 3.2)*
- *chaque nœud étiqueté par $T/T' \diamond v$ a n fils $T/T'_1 \diamond_1 v_1, ..., T/T'_n \diamond_n v_n$ tels que*
  - $\diamond, \diamond_1, ...\diamond_n \in \{\vdash, \vdash'\}$
  - $\dfrac{T/T'_1 \diamond_1 u_1 \;\; ... \;\; T/T'_n \diamond_n u_n}{T/T' \diamond v}$ *soit une instance d'une des règles d'inférence de la figure 3.2*
- *la racine de l'arbre est étiquetée par $T/T' \vdash u$*

Remarque : Dans l'arbre de preuve, les membres gauches des séquents qui étiquettent les différents noeuds de l'arbre ne sont pas tous identiques, contrairement aux arbres de preuve dérivés dans le modèle de *Dolev-Yao* standard.

Définition 3.7 *(preuve en forme normale)*
Une preuve est en forme normale lorsque les transformations suivantes ne peuvent plus s'appliquer :

$$\cfrac{\cfrac{\cfrac{P_1}{T/T'_1 \vdash' u_1} \qquad \cfrac{P_2}{T/T'_2 \vdash' u_2}}{T/T'_1 \cup T'_2 \vdash' \langle u_1, u_2 \rangle}\ \mathsf{P}'}{T/T'_1 \cup T'_2 \vdash' u_1}\ \mathsf{UL}' \quad \rightsquigarrow \quad \cfrac{P_1}{T/T'_1 \vdash' u_1}$$

$$\cfrac{\cfrac{\cfrac{P_1}{T/T_1' \vdash' \langle u_1, u_2\rangle}}{T/T_1' \vdash' u_1}\ \mathsf{UL}' \qquad \cfrac{P_2}{T/T_2' \vdash' u_2}}{T/T_1' \cup T_2' \vdash' \langle u_1, u_2\rangle}\ \mathsf{P}' \qquad \rightsquigarrow \qquad \cfrac{P_1}{T/T_1' \vdash' u_1}$$

Des règles similaires existent pour (P'/UR'),(D'/E') et pour (P/UL,UR),(D/E).

Nous pouvons reformuler le *problème de déduction de l'intrus* de la façon suivante :

Etant donné deux ensembles finis de messages T et T', et un secret s, est-ce qu'il existe une preuve de $T/T_1' \vdash s$ telle que $T_1' \subseteq T'$?

Remarque : Dans un arbre de preuve il y a au plus une instance de la règle Comparaison par branche.

3.3.3 Exemples

Dans cette section, nous allons donner des exemples d'utilisation de la règle (C). Nous allons formaliser les raisonnements intuitifs que nous avons présentés précedemment (cf. section 3.1).

Exemple 3.4
Nous continuons l'exemple 3.1.
Supposons que $T = \{\{m\}_{pub(S)}, pub(S)\}$ et que m soit une donnée faible, l'attaque décrite à la section 3.1 correspond à la dérivation suivante :

$$\cfrac{\cfrac{\{m\}_{pub(S)} \in T}{T/\emptyset \vdash' \{m\}_{pub(S)}}\ \mathsf{W} \qquad \cfrac{\cfrac{}{T/\{m\} \vdash' m}\ \mathsf{A}' \qquad \cfrac{pub(S) \in T}{T/\emptyset \vdash' pub(S)}\ \mathsf{W}}{T/\{m\} \vdash' \{m\}_{pub(S)}}\ \mathsf{E}'}{T/\{m\} \vdash m}\ \mathsf{C}$$

L'intrus devine la valeur de m, produit le vérifieur $\{m\}_{pub(S)}$ (*sous-arbre gauche*), et il vérifie ensuite sa prédiction puisque le vérifieur fait partie de ses connaissances initiales.

Exemple 3.5
Nous continuons l'exemple 3.2.
Supposons que $T = \{\{n\}_k, \{n+1\}_k\}$ et que k (clef symétrique) est une donnée faible, alors on peut dériver $n+1$ de deux façons différentes :

La preuve P_1 :

$$\dfrac{\dfrac{}{T/\emptyset \vdash' \{n\}_k}\,\mathsf{W} \qquad \dfrac{}{T/\{k\} \vdash' k}\,\mathsf{A'}}{\dfrac{T/\{k\} \vdash' n}{T/\{k\} \vdash' n+1}}\,\mathsf{D'}$$

La preuve P_2 :

$$\dfrac{\dfrac{}{T/\emptyset \vdash' \{n+1\}_k}\,\mathsf{W} \qquad \dfrac{}{T/\{k\} \vdash' k}\,\mathsf{A'}}{T/\{k\} \vdash' n+1}\,\mathsf{D'}$$

P_1 et P_2 sont deux preuves différents du vérifieur $n+1$. On en déduit que l'intrus connaît k et il peut déchiffrer $\{n\}_k$ pour en déduire n :

$$\dfrac{\dfrac{\{n\}_k \in T}{T/\emptyset \vdash \{n\}_k}\,\mathsf{A} \qquad \dfrac{P_1 \qquad P_2}{T/\{k\} \vdash k}\,\mathsf{C}}{T/\{k\} \vdash n}\,\mathsf{D}$$

Exemple 3.6
Nous continuons l'exemple 3.3.
Supposons que $T = \{pub(A)\}$ et que $priv(A)$ soit une donnée faible. L'intrus peut deviner une valeur pour $priv(A)$ et vérifier sa prédiction en testant si la valeur choisie est bien l'inverse de la clef $pub(A)$.
Nous avons la dérivation suivante :

$$\dfrac{\dfrac{}{T/\{priv(A)\} \vdash' priv(A)}\,\mathsf{A'} \qquad \dfrac{pub(A) \in T}{T/\emptyset \vdash' pub(A)}\,\mathsf{W}}{T/\{priv(A)\} \vdash priv(A)}\,\mathsf{C}$$

L'intrus a donc déduit la clef privée de l'agent A.

3.4 Résultats de décidabilité et de complexité

Notre but est de montrer que le *problème de déduction de l'intrus* est décidable en temps polynomial dans ce nouveau modèle d'intrus. Pour obtenir ce résultat, nous allons commencer par prouver (comme dans le cas du modèle de *Dolev-Yao*) un résultat de localité : si $T/T' \vdash s$, alors il existe une preuve en forme normale qui utilise uniquement des termes de $St(T \cup T' \cup \{s\})$.

Théorème 3.2
Si P est une preuve en forme normale de $T/T' \diamond u$, alors elle n'utilise que des termes de $St(T \cup T' \cup \{u\})$

Autrement dit, P est un arbre de preuve étiqueté par des séquents $T/T'_1 \diamond v$ tels que :
- $\diamond \in \{\vdash, \vdash'\}$
- $T'_1 \subseteq T'$
- $v \in St(T \cup T' \cup \{u\})$

Preuve :
Nous prouvons ce résultat par récurrence sur la taille de la preuve de $T/T' \diamond u$:

1. Une preuve en forme normale de $T/T' \diamond u$ contient uniquement des termes de $St(T \cup T' \cup \{u\})$.
2. Si la dernière règle d'inférence de la preuve est une règle de décomposition, (A, UL, UR, D, A′, UL′,UR′, D′, W, C), alors la preuve contient seulement des termes de $St(T \cup T')$.

Considérons les différentes possibilités pour la dernière régle d'inférence :

- Si la dernière règle est (A), alors le résultat est immédiat.

- Si la dernière règle est (P), alors $u = \langle u_1, u_2 \rangle$ et la preuve est de la forme :

$$\dfrac{\dfrac{P_1}{T/T'_1 \vdash u_1} \quad \dfrac{P_2}{T/T'_2 \vdash u_2}}{T/T'_1 \cup T'_2 \vdash \langle u_1, u_2 \rangle} \ (\mathsf{P})$$

  Par l'hypothèse d'induction (1), P_1 est une preuve en forme normale de $T/T'_1 \vdash u_1$, contenant uniquement des termes de $St(T \cup T'_1 \cup \{u_1\})$ et P_2 est une preuve en forme normale de $T/T'_2 \vdash u_2$ qui ne contient que des termes de $St(T \cup T'_2 \cup \{u_2\})$. On en déduit donc que P n'utilise que des termes de $St(T \cup T' \cup \{\langle u_1, u_2 \rangle\})$.

- Le cas où la dernière règle est (E) est similaire.

- Si la dernière règle est (UL), alors la preuve est de la forme :

$$\dfrac{\dfrac{P_1}{T/T'_1 \vdash \langle u, v \rangle}}{T/T'_1 \vdash u} \ (\mathsf{UL})$$

  Regardons la dernière règle de la preuve P_1. Cette dernière règle est nécessairement (A), (UL), (UR) ou (D). En effet, (P) est impossible par minimalité de la preuve et (C) est impossible car $\langle u, v \rangle$ n'est pas un terme atomique. On peut donc appliquer l'hypothèse d'induction (2) et on obtient que P_1 ne contient que des termes de $St(T \cup T'_1)$, et donc P ne contient que des termes de $St(T \cup T'_1)$.

- Le cas où la dernière règle est (UR) est similaire.

- Si la dernière règle est (D), alors

$$\dfrac{\dfrac{P_1}{T/T'_1 \vdash \{u\}_v} \quad \dfrac{P_2}{T/T'_2 \vdash v^{-1}}}{T/T'_1 \cup T'_2 \vdash u}\ (\mathsf{D})$$

Regardons la dernière règle de la preuve P_1. Cette dernière règle est nécessairement (A), (UL), (UR) ou (D). En effet, (E) est impossible par minimalité de la preuve et (C) est impossible car $\{u\}_v$ n'est pas un terme atomique. On peut donc appliquer l'hypothèse d'induction (2) et on obtient que P_1 ne contient que des termes de $St(T \cup T'_1)$.

On distingue alors deux cas :
– Si v est un inverse v'^{-1}, alors on a $v^{-1} = v'^{-1^{-1}} = v'$. Par hypothèse d'induction (1) P_2 ne contient que des termes de $St(T \cup T'_2 \cup \{v'\})$; v' est un sous-terme de v, donc $v' \in St(T \cup T'_1)$. On en déduit donc que P ne contient que des termes de $St(T \cup T'_1 \cup T'_2)$.

– Sinon v n'est pas un inverse, la dernière règle de P_2 est alors (A), (UL), (UR) ou (D). Par hypothèse d'induction (2), on en déduit que P_2 ne contient que des termes de $St(T \cup T'_2)$, et donc on en conclut que P ne contient que des termes de $St(T \cup T'_1 \cup T'_2)$.

- Les cas où la dernière règle est (A'), (P'), (UL'), (UR'), (D'), (E') ou (W) sont similaires au cas précédents.

- Si la dernière règle est (C), alors

$$\dfrac{P_1 \left\{ \dfrac{\dfrac{..}{\vdash' x_1} \cdots \dfrac{..}{\vdash' x_n}}{T/T'_1 \vdash' u}\ (\mathsf{R1}) \right. \qquad P_2 \left\{ \dfrac{\dfrac{..}{\vdash' y_1} \cdots \dfrac{..}{\vdash' y_n}}{T/T'_2 \vdash' v}\ (\mathsf{R2}) \right.}{T/T'_1 \cup T'_2 \vdash w}$$

où :

(i). $w \in T'_1 \cup T'_2$

(ii). P_1 et P_2 sont des preuves en forme normale

(iii). $\mathsf{R1} \neq \mathsf{R2}$ ou $\{u, x_1, .., x_n\} \neq \{v, y_1, ..., y_n\}$

(iv). $R(u, v)$ avec $R = Id \cup \{(k, k^{-1}) |$ k est une clef $\}$

Nous distinguons deux cas :
– $u = v$
Si la dernière règle d'inférence de P_1 est (A'), (UL'), (UR'), (D') ou (W) ; alors P_1 ne contient que des termes de $St(T \cup T'_1)$, et on a $u \in St(T \cup T'_1)$. Par hypothèse d'induction, P_2 ne contient que des termes de $St(T \cup T'_2 \cup \{u\})$. De plus, par (i) nous avons que $w \in T'_1 \cup T'_2$, on en déduit donc P ne contient que des termes de $St(T \cup T'_1 \cup T'_2)$.

Si la dernière règle d'inférence de P_2 est (A'), (UL'), (UR'), (D') ou (W) ; alors par un

raisonnement similaire, on en déduit que P ne contient que des termes de $St(T \cup T'_1 \cup T'_2)$.
Sinon, les preuves P_1 et P_2 se terminent toutes les deux par la même instance de (E'), (ou par la même instance de (P')). Mais ceci contredit la condition (iii).

– u est une clef asymétrique et v son inverse
Supposons sans perte de généralité que $v = u^{-1}$ et u, v sont en forme normale. La dernière règle d'inférence de P_2 est (A'), (UL'), (UR'), (D') ou (W) ; et on obtient le résultat par un raisonnement similaire au premier cas.

$\square$

Théorème 3.3
Le problème de déduction de l'intrus $T/T' \overset{?}{\vdash} s$, *est décidable en temps polynomial dans le modèle de* Dolev-Yao *étendu.*

Preuve :
Soit n la taille de $T \cup T' \cup \{s\}$. La taille d'un ensemble de termes est définie comme la somme des tailles de chacun des termes de l'ensemble. Le nombre d'éléments de $St(T \cup T' \cup \{s\})$ est donc linéaire en la taille de $T \cup T' \cup \{s\}$.

Pour décider le *problème de déduction de l'intrus* en présence de données de type faibles, il est suffisant de :
- trouver parmi les données faibles, celles que l'intrus peut déduire de façon forte.
- ajouter ces données à la connaissance forte de l'intrus, puis résoudre le *problème de déduction de l'intrus* dans le modèle de *Dolev-Yao* standard.

$IK(x)$	$\Rightarrow$	$I(x)$	Connaissance intiale de l'intrus.
$I(x), I(y)$	$\Rightarrow$	$I(\langle x, y \rangle)$	L'intrus peut construire une paire.
$I(x), I(y)$	$\Rightarrow$	$I(\{x\}_y)$	L'intrus peut chiffrer un message avec une clef.
$I(\langle x, y \rangle)$ $I(\langle x, y \rangle)$	$\Rightarrow$ $\Rightarrow$	$I(x)$ $I(y)$	L'intrus peut décomposer une paire.
$I(\{x\}_y), I(y^{-1})$	$\Rightarrow$	$I(x)$	L'intrus peut déchiffrer un texte seulement s'il connaît la clef de déchiffrement.

FIGURE 3.3 - *Les clauses de Horn de Dolev-Yao.*

Ce deuxième point est décidable en $O(n)$: il suffit de coder le système d'inférence de *Dolev-Yao* en un ensemble de clauses de Horn (cf. figure 3.3). Puisque la théorie de *Dolev-Yao* est locale [1], (*i.e.* s'il existe une preuve de $T \vdash t$, alors il existe une preuve de $T \vdash t$ qui n'utilise que des sous-termes de $T \cup \{t\}$), il nous suffit donc de considérer les instances de ces clauses qui ne font intervenir que des termes de $St(T \cup \{t\})$. Cet ensemble de clauses closes $\mathcal{S}$ est linéaire en la taille de $T \cup \{t\}$.
Déterminer si un terme t est déductible par l'intrus est donc équivalent à déterminer si $I(t)$ est dérivable à partir de l'ensemble $\mathcal{S}$. L'existence d'une telle dérivation est décidable en temps linéaire en le nombre de clauses de l'ensemble $\mathcal{S}$, c'est à dire en temps linéaire en la taille de $T \cup \{t\}$.

Pour le premier point, nous allons suivre le même schéma et coder une partie du système d'inférence de la figure 3.2 (règles $\mathsf{A}', \mathsf{W}, \mathsf{UL}', \mathsf{UR}', \mathsf{P}', \mathsf{D}', \mathsf{E}'$) par un ensemble de clauses de Horn, et nous allons considérer toutes les instances possibles de ces clauses qui ne font intervenir que des termes de $St(T \cup T' \cup \{t\})$. Ainsi, savoir si une donnée faible est fortement connu de l'intrus sera équivalent à déterminer si un symbole propositionnel peut être dérivé à partir d'un ensemble de clauses de Horn closes. Les conditions qui sont demandées pour appliquer la règle (C) seront pris en compte dans le codage en clauses de Horn. Mais, nous allons montrer que le nombre total de clauses de Horn closes est polynomial en la taille du problème et donc que l'existence d'une dérivation est décidable en temps polynomial (HORNSAT).

Par la suite, nous cherchons donc à résoudre le problème suivant :

Etant donné une donnée atomique faible $w \in T'$, est-ce qu'il existe une preuve dont la racine est étiquetée par :

$$\dfrac{P_1 \left\{ \dfrac{\dfrac{..}{\vdash' x_1} \quad \cdots \quad \dfrac{..}{\vdash' x_n}}{T/T'_1 \vdash' u}\ (\mathsf{R1}) \right. \qquad P_2 \left\{ \dfrac{\dfrac{..}{\vdash' y_1} \quad \cdots \quad \dfrac{..}{\vdash' y_n}}{T/T'_2 \vdash' v}\ (\mathsf{R2}) \right.}{T/T'_1 \cup T'_2 \vdash w}\ (\mathsf{Comparaison})$$

où :

(i). $w \in T'_1 \cup T'_2 \subseteq T'$

(ii). P_1 et P_2 sont des preuves en forme normale

(iii). $\mathsf{R1} \neq \mathsf{R2}$ ou $\{u, x_1, .., x_n\} \neq \{v, y_1, ..., y_n\}$

(iv). $R(u, v)$ avec $R = Id \cup \{(k, k^{-1})|$ k est une clef $\}$

Remarque : Les preuves P_1 et P_2 n'utilisent que les règles $\mathsf{A}', \mathsf{W}, \mathsf{UL}', \mathsf{UR}', \mathsf{P}', \mathsf{E}', \mathsf{D}'$.

Si l'on suppose qu'initialement l'intrus connaît un ensemble de messages représenté par le prédicat IK (messages fortement et faiblement connus), le prédicat I défini à la figure 3.3 est tel que $I(t)$ est vrai si et seulement si t est déductible en utilisant les régles ($\mathsf{UL}', \mathsf{UR}', \mathsf{P}', \mathsf{E}', \mathsf{D}'$).

Pour prendre en compte les conditions d'application de la règle (C), on remplace le prédicat unaire I par deux prédicats binaires :

- le **prédicat** $I(t, lr)$, le deuxième argument est utilisé pour mémoriser l'instance de la dernière règle d'inférence utilisée dans la preuve de t. Par exemple, $I(t_1, UL(t_2))$ signifie que la preuve dont la racine est étiquetée par le terme t_1 a la forme suivante :

$$\dfrac{\dfrac{\cdots}{T/T' \vdash' \langle t_1, t_2 \rangle}}{T/T' \vdash' t_1}\ (\mathsf{UL}')$$

  Ce nouvel argument va permettre d'assurer les conditions (ii) et (iii) lorsqu'il s'agira d'appliquer la règle (C).

- le **prédicat** $J(t, lr)$ est une copie de I, il est utilisé pour marquer un terme qui a été produit à partir d'une donnée faible w donnée. Il va permettre d'assurer la condition (i) au moment de l'application de la règle (C).

	$\Rightarrow$	$J(w, A)$	w la donnée faible donnée dans le problème.	(1)
$IK(x)$	$\Rightarrow$	$I(x, A)$	Connaissance initiale de l'intrus (faible et forte).	(2)

$$\left.\begin{array}{rcl} I(x, lr_1), I(y, lr_2) & \Rightarrow & I(\langle x, y\rangle, P) \\ I(x, lr_1), J(y, lr_2) & \Rightarrow & J(\langle x, y\rangle, P) \\ J(x, lr_1), I(y, lr_2) & \Rightarrow & J(\langle x, y\rangle, P) \\ J(x, lr_1), J(y, lr_2) & \Rightarrow & J(\langle x, y\rangle, P) \end{array}\right\} \text{ avec } lr_1 \neq UL(y) \text{ et } lr_2 \neq UR(x) \qquad (3)$$

$$\left.\begin{array}{rcl} I(x, lr_1), I(y, lr_2) & \Rightarrow & I(\{x\}_y, E) \\ I(x, lr_1), J(y, lr_2) & \Rightarrow & J(\{x\}_y, E) \\ J(x, lr_1), I(y, lr_2) & \Rightarrow & J(\{x\}_y, E) \\ J(x, lr_1), J(y, lr_2) & \Rightarrow & J(\{x\}_y, E) \end{array}\right\} \text{ avec } lr_1 \neq D(y^{-1}) \qquad (4)$$

$$\left.\begin{array}{rcl} I(\langle x,y\rangle,lr) & \Rightarrow & I(x,UL(y)) \\ J(\langle x,y\rangle,lr) & \Rightarrow & J(x,UL(y)) \end{array}\right\} \quad \text{avec } lr \neq P \tag{5}$$

$$\left.\begin{array}{rcl} I(\langle x,y\rangle,lr) & \Rightarrow & I(y,UR(x)) \\ J(\langle x,y\rangle,lr) & \Rightarrow & J(y,UR(x)) \end{array}\right\} \quad \text{avec } lr \neq P \tag{6}$$

$$\left.\begin{array}{rcl} I(\{x\}_y,lr_1),I(y^{-1},lr_2) & \Rightarrow & I(x,D(y^{-1})) \\ I(\{x\}_y,lr_1),J(y^{-1},lr_2) & \Rightarrow & J(x,D(y^{-1})) \\ J(\{x\}_y,lr_1),I(y^{-1},lr_2) & \Rightarrow & J(x,D(y^{-1})) \\ J(\{x\}_y,lr_1),J(y^{-1},lr_2) & \Rightarrow & J(x,D(y^{-1})) \end{array}\right\} \quad \text{avec } lr_1 \neq E \tag{7}$$

$$\left.\begin{array}{rcl} I(x,lr_1),J(x,lr_2) & \Rightarrow & Goal \\ J(x,lr_1),J(x,lr_2) & \Rightarrow & Goal \end{array}\right\} \quad \text{avec } lr_1 \neq lr_2 \tag{8}$$

$$\left.\begin{array}{rcl} I(x,lr_1),J(x^{-1},lr_2) & \Rightarrow & Goal \\ J(x,lr_1),I(x^{-1},lr_2) & \Rightarrow & Goal \\ J(x,lr_1),J(x^{-1},lr_2) & \Rightarrow & Goal \end{array}\right\} \quad \text{avec } x \neq x^{-1} \tag{9}$$

Ajoutons les clauses :

$$\Rightarrow \quad IK(t) \quad \text{pour } t \in T \cup T'.$$

Le prédicat I est tel que $I(lr,u)$ si et seulement s'il existe une preuve en forme normale (se terminant par la règle lr) et dont la racine est étiquetée par $T/T'_1 \vdash' u$ avec $T'_1 \subseteq T'$.

Le prédcat J est tel que $J(lr,u)$ si et seulement s'il existe une preuve en forme normale (se terminant par la règle lr) et dont la racine est étiquetée par $T/T'_1 \vdash' u$ avec $w \in T'_1$ et $T'_1 \subseteq T'$.

Pour appliquer la règle Comparaison (pour déduire w), nous devons avoir deux preuves d'un même terme (le vérifieur) et une (au moins) de ces deux preuves doit utiliser la donnée faible w. Cette condition est assurée par l'ensemble de clauses de Horn (8) et (9). Le prédicat $Goal$ est vrai si et seulement si la donnée faible w peut être déduite par l'intrus.

Soit S l'ensemble de toutes ces clauses *((1)-(9))*.

Maintenant, nous allons déduire du théorème de localité (cf. théorème 3.2), un résultat de décidabilité et de complexité. S'il existe une preuve de $T/T' \vdash t$, alors il existe une preuve qui n'utilise que les sous-termes de $T \cup T' \cup \{t\}$, il nous suffit donc de considérer les instances des clauses de S qui ne font intervenir que les termes de $St(T \cup T' \cup \{t\})$.

Ainsi, savoir si une donnée faible w est fortement connu de l'intrus est équivalent à déterminer si le prédicat $Goal$ peut être dérivé à partir d'un ensemble de clauses de Horn closes.

L'existence d'une telle dérivation est décidable en temps linéaire par rapport au nombre de clauses de l'ensemble.

Nous allons donner une borne supérieure sur le nombre de clauses à considérer, et nous avons besoin pour cela de quelques définitions supplémentaires.

Posons
$\mathcal{ST} = St(T \cup T')$,
$\mathcal{LR} = \{A, P, E, D(x^{-1}), UL(x), UR(x) | x \in ST\}$.

La taille de ces deux ensembles est $O(n)$ où n est la taille de $T \cup T' \cup \{s\}$.

Définition 3.8 *(ordre d'une clause)*
Une clause c est dite d'ordre k s'il existe des termes $e_1, ..., e_k$, tels que chaque variable appa-raissant dans c apparaisse aussi dans un des e_i.

Exemple 3.7
La clause $I(x, lr_1), I(y, lr_2) \Rightarrow I(\langle x, y\rangle, P)$ avec $lr_1 \neq UL(y)$ et $lr_2 \neq UR(x)$ est d'ordre 3 car les 3 expressions lr_1, lr_2 et $\langle x, y\rangle$ satisfont la condition demandée dans la définition.
La clause $I(\langle x, y\rangle, lr) \Rightarrow I(x, UL(y))$ avec $lr \neq P$ est d'ordre 2.
Les clauses de l'ensemble S sont d'ordre 3 (ou moins).

Soit c une clause dans S, et soient e_1, e_2 ... e_j les termes satisfaisant les conditions de la définition disant que c est une clause d'ordre j.

Nous avons simplement besoin de considérer les instances $\rho(c)$ où ρ est une substitution telle que $\rho(e_1)$, $\rho(e_2)$,.., $\rho(e_j)$ soient dans $\mathcal{ST}$ ou dans $\mathcal{LR}$. Puisque chaque variable dans c apparaît dans un des e_i, l'ensemble de toutes les instances $\rho(c)$ peut être calculé en matchant les expressions e_1, e_2,..., e_j avec les éléments de $\mathcal{ST}$ et de $\mathcal{LR}$. Pour une clause fixée c, l'ensemble de toutes les combinaisons possibles pour choisir les valeurs des expressions e_1, e_2, ..., e_j parmi les éléments de $\mathcal{ST} \cup \mathcal{LR}$ est calculable en $O(n^j)$.

La restriction qui impose que chaque $\rho(e_i)$ soit un élément de $\mathcal{ST}$ ou de $\mathcal{LR}$ ne garantie pas le fait que la conclusion et les prémisses de $\rho(c)$ soient composées uniquement par des termes de $\mathcal{ST} \cup \mathcal{LR}$. Soit $I(c)$ l'ensemble formé des instances des clauses dont la conclusion et les prémisses sont formées des termes dans $\mathcal{ST} \cup \mathcal{LR}$. L'ensemble $I(c)$ est de taille n^j. Soit $I(S)$ l'union de tous les ensembles $I(c)$ pour les clauses c dans S. L'ensemble $I(S)$ est de taille $O(n^3)$ (puisque les clauses de l'ensemble S sont d'ordre 3 ou moins).

Nous pouvons maintenant prouvé le lemme suivant :

Lemme 3.1
Il existe une preuve dont la racine est étiquetée par $T/T'_1 \vdash w$ si et seulement si le symbole Goal est dérivable à partir de l'ensemble de clauses closes $I(S)$. Ce problème est donc équivalent à HORNSAT pour un ensemble de clauses de taille $O(n^3)$.

Pour résoudre le problème de déduction de l'intrus en présence de données faibles, nous devons trouver parmi les données faibles, celles que l'intrus peut déduire fortement. Ce problème est décidable en $n^3 \times |St(T')|$, i.e. $O(n^4)$.

Le problème de déduction de l'intrus en présence de données faibles, $T/T' \overset{?}{\vdash} s$, est donc décidable en temps polynomial en la taille de $T \cup T' \cup \{s\}$.

$\square$

3.5 Conclusion et Perspectives

Nous avons étendu le modèle de *Dolev-Yao* pour prendre en compte les « attaques par prédictions », et nous avons montré que le *problème de déduction de l'intrus* dans ce modèle étendu est PTIME.

Ce travail peut être étendu pour vérifier la sécurité d'un protocole pour un nombre borné de sessions. En effet, vérifier si un protocole est sûr est équivalent à décider si la séquence de messages représentant l'attaque est accessible, c'est à dire si l'intrus peut, en utilisant le protocole construire cette séquence. Ce problème peut donc être vu comme un ensemble ordonné de contraintes $T_1/T_1' \vdash u_1$, ..., $T_n/T_n' \vdash u_n$ où chacune des contraintes $T_i/T_i' \vdash u_i$ est un relèvement du *problème de déduction de l'intrus* aux termes avec variables.

Si ce problème de résolution de contraintes est NP, comme dans le cas du modèle standard de *Dolev-Yao*, le problème de la sécurité d'un protocole en présence de données faibles pour un nombre borné de sessions sera co-NP [27].

Chapitre 4

Vérification pratique des protocoles cryptographiques

La vérification des protocoles cryptographiques est difficile car de nombreux paramètres ne sont pas bornés : nombre arbitraire de sessions et de participants, taille arbitraire des messages...

Securify est un des outils permettant de prouver la correction d'un protocole dans le cadre d'un nombre non borné de sessions. Il a été développé au LSV par Véronique Cortier, puis intégré au projet RNTL EVA. Dans ce chapitre, nous commençons par décrire le fonctionnement de cet outil, et nous proposons quelques améliorations.

Securify permet de vérifier automatiquement les protocoles cryptographiques, mais il comporte (comme les démonstrateurs existants actuellement) de nombreuses faiblesses. En particulier, il se concentre sur un seul modèle d'attaquant, celui de *Dolev-Yao*. Securify repose donc sur l'*hypothèse du chiffrement parfait*, et son étude a révélé qu'il serait difficile de séparer l'algorithme de recherche de preuve de la théorie de l'intrus (section 4.1).

Afin d'expérimenter les résultats trouvés au chapitre 3, nous avons donc choisi de développer une procédure de recherche d'attaque, simple mais modulaire ainsi que les modules d'intrus correspondant à la théorie de *Dolev-Yao* standard et à la théorie de *Dolev-Yao* étendue aux données faibles (section 4.2).

Enfin, nous exposons les résultats obtenus après implémentation de la procédure.

4.1 Securify : un outil de vérification du secret

Cet outil peut être expérimenté en ligne à l'adresse suivante :
`http://www-eva.imag.fr/outils1.html`

Dans cette section, nous présentons Securify ainsi que le modèle formel sur lequel il repose. Le but est de comprendre les bases du fonctionnement de cet outil afin d'être en mesure de :

- comprendre la généralisation des tests élémentaires que nous proposons,
- apprécier les limites de cet outil.

Pour plus de détails dans les définitions, les propositions et les théorèmes énoncés, se reporter au chapitre 4 et 7 de la thèse de Véronique Cortier [14].

Securify est un outil dédié à la preuve du secret des protocoles pour un nombre non borné de sessions, cette procédure demande que le protocole soit écrit dans le modèle de Millen-Rueß. Ce modèle présente la particularité de définir la propriété à vérifier en même temps que le protocole lui-même et ne permet d'exprimer qu'une seule propriété : le secret (long terme) de nonces ou de clefs engendrés au cours du protocole. De plus, les protocoles ne doivent pas utiliser de clefs composées.

Les réponses de l'outil sont de trois types :
- réponse « oui » : le secret des données à protéger est garanti dans le modèle Millen-Rueß,
- réponse « échec » : la méthode de preuve échoue, on ne peut pas conclure si le protocole est sûr ou non. Cependant, l'arbre d'échec de la preuve peut assez souvent permettre de construire une attaque et montrer ainsi que le protocole n'est pas correct.
- pas de réponse : l'outil ne termine pas.

Exemple 4.1
Le protocole (informel) de Needham-Schroeder-Lowe :

$$\begin{array}{lll} A & \rightarrow B : & \{A, N_a\}_{\mathsf{pub}(B)} \\ B & \rightarrow A : & \{N_a, N_b, B\}_{\mathsf{pub}(A)} \\ A & \rightarrow B : & \{N_b\}_{\mathsf{pub}(B)} \end{array}$$

4.1.1 Présentation du modèle Millen-Rueß

Il s'agit d'un modèle de trace : on considére l'historique de tous les évènements qui ont eu lieu lors du déroulement des différentes instances du protocole. Le protocole spécifie quels sont les évènements qui peuvent être ajoutés à partir d'un historique donné. Ces évènements sont de trois sortes :
- l'émission d'un message
- la mise à jour de l'état local d'un agent (mais cette possibilité du modèle Millen-Rueß n'est pas exploitée par Securify)
- la déclaration qu'une donnée doit rester secrète, que l'on note $\{N\} \ddagger \{A, B\}$ et qui spécifie que le nonce N ne doit être connu qu'au plus par les agents A et B.

Notation :
Soit $C = S \ddagger L$ une spécification, on note :
- $SpecSec(C) = S$
- $SpecAgent(C) = L$

– $Sec(C) = SpecSec(C) \cup \{priv(a), shr(a) | a \in L\}$
où $priv(a)$ et $shr(a)$ représentent les clefs privées long terme de l'agent a.

Un **protocole** (définition 4.12 [14]) est simplement un ensemble de spécifications de transitions et s'exécute à partir d'un historique vide. Les **transitions** (cf. définition 4.11 [14]) doivent respecter un certain nombre de conditions, parmi lesquelles :

- *condition de fraîcheur*, seules les données nouvellement créées (nonces) peuvent être déclarées secrètes
- *spécifications disjointes*, si $\{S_1\} \ddagger \{L_1\}$ et $\{S_2\} \ddagger \{L_2\}$ sont dans $Post(t)$ alors S_1 et S_2 sont disjoints
- ...

Exemple 4.2
Le protocole de Needham-Schroeder-Lowe dans le modèle de Millen-Rueß est présenté à la figure 4.1.

$$\emptyset \xrightarrow{\{N_a\}} \left\{\mathsf{A}_1(A, B, N_a), \{N_a\} \ddagger \{A, B\}\right\} \tag{4.1}$$

$$\emptyset \xrightarrow{\{N_b\}} \left\{\mathsf{B}_1(B, N_b), \{N_b\} \ddagger \{A, B\}\right\} \tag{4.2}$$

$$\{\{\mathsf{A}_1(A, B, N_a)\}, \{N_a\} \ddagger \{A, B\}\} \xrightarrow{\emptyset} \left\{\mathsf{A}_2(A, B, N_a), \{A, N_a\}_{\mathsf{pub}(B)}\right\} \tag{4.3}$$

$$\left\{\begin{matrix}\mathsf{B}_1(B, N_b), \{N_b\} \ddagger \{A, B\}, \\ \{A, N_a\}_{\mathsf{pub}(B)}\end{matrix}\right\} \xrightarrow{\emptyset} \left\{\mathsf{B}_2(B, N_b, A, N_a), \{N_a, N_b, B\}_{\mathsf{pub}(A)}\right\} \tag{4.4}$$

$$\left\{\begin{matrix}\mathsf{A}_2(A, B, N_a), \{N_a\} \ddagger \{A, B\}, \\ \{N_a, N_b, B\}_{\mathsf{pub}(A)}\end{matrix}\right\} \xrightarrow{\emptyset} \left\{\mathsf{A}_3(A, B, N_a, N_b), \{N_b\}_{\mathsf{pub}(B)}\right\} \tag{4.5}$$

FIGURE 4.1 - *Protocole de Needham-Schroeder-Lowe.*

Les règles (4.1) et (4.2) spécifient que les participants A, B sont prêts à commencer une nouvelle session du protocole. Les nonces N_a et N_b qui vont être utilisés dans la session et qui sont déclarés secrets sont générés.

La règle (4.3) spécifie que si l'agent A est dans son état initial $\mathsf{A}_1(A, B, N_a)$, il envoie le message $\{A, N_a\}_{pub(B)}$ et passe dans l'état $\mathsf{A}_2(A, B, N_a)$.

La règle (4.4) dit que si l'agent B est dans son état initial $\mathsf{B}_1(B, N_b)$ et s'il reçoit un message de la forme $\{A, N_a\}_{pub(B)}$ (la variable A (resp. N_a) peut être instanciée par n'importe quel message de type agent (resp. nonce)), alors B utilise son nonce N_b engendré en début de session et supposé secret pour envoyer le message $\{N_a, N_b, B\}_{pub(A)}$ et passe dans l'état $\mathsf{B}_2(B, N_b, A, N_a)$.

La règle (4.5) est construite de manière similaire.

L'étude d'un protocole revient à l'étude de l'ensemble des historiques accessibles en *suivant le protocole.*

Idéaux et Coidéaux

Maintenant, il nous faut définir l'ensemble des messages que l'on doit protéger et qui ne devront pas circuler sur le réseau, c'est la notion d'idéal. L'**idéal** (cf. définition 4.7 [14]) est le plus petit sur-ensemble de l'ensemble des secrets à protéger, clos par concaténation avec des messages arbitraires et clos par chiffrement par des clefs dont l'inverse n'est pas dans l'ensemble considéré.

Exemple 4.3
Soit $S = \{N_a\} \ddagger \{A, B\}$ *la spécification dans le membre droit de la transition (4.1) de l'exemple 4.2, on cherche à protéger le secret* N_a *ainsi que les clefs privées long terme des agents* A *et* B*, c'est à dire* $priv(A)$ *et* $priv(B)$*. L'ensemble des messages que l'on doit protéger est donc* $Ideal(\{N_a, priv(A), priv(B)\})$*.*

Le complémentaire d'un Idéal $I(S)$, appelé **coidéal** de S, est donc l'ensemble des messages qui peuvent être révélés sans compromettre, en aucun cas, les termes de S. L'ensemble $Coideal(S)$ est clos par analyse et synthèse (c'est à dire par application des règles UL, UR, D et E, P de la figure 3.1) (cf. proposition 4.2 et 4.3 [14]). Par conséquent, l'ensemble des messages calculables par l'intrus à partir de $Coideal(S)$ reste dans $Coideal(S)$.

Remarque : La fermeture de $Coideal(S)$ par synthèse fait intervenir le fait que S est un ensemble de messages atomiques.

Notion de secret

Il nous reste à définir la notion de secret.

Définition 4.1 *(protocole impénétrable) (cf. définition 4.15 [14])*
Un protocole est impénétrable *si :*
- *pour toute connaissance initiale* I *de l'intrus*
- *pour tout historique* H *accessible (par des étapes honnêtes, c'est à dire en jouant une règle du protocole, ou par des étapes malhonnêtes, c'est à dire en ajoutant dans l'historique un message construit par l'intrus)*
- *pour toute spécification* $C \in H$

alors les messages de H *sont dans le coidéal de* C*,*

$$Msg(H) \subseteq Coideal(Sec(C))$$

où $Msg(H)$ *est l'ensemble des évènements de* H *qui sont des messages.*

Autrement dit, les messages de l'historique ne peuvent pas compromettre l'ensemble des secrets $Sec(C)$.

Si un protocole P est impénétrable, alors pour toute connaissance initiale I de l'intrus, P préserve le secret pour la définition de secret suivante :

Définition 4.2 *(protocole préserve le secret) (cf. définition 4.16 [14])*
Un protocole préserve le secret pour la connaissance initiale I de l'intrus si pour tout historique H, ensemble d'évènements, accessible et pour toute spécification C de H, alors les secrets de C ne sont pas connus de l'intrus :

$$fake(Msg(H) \cup I)) \cap Sec(C) = \emptyset$$

où fake(Msg(H)∪ I) représente l'ensemble des messages déductibles par les règles de la figure 3.1 à partir des messages contenus dans I et dans l'historique H.

Proposition 4.1
Si P est un protocole impénétrable alors P préserve le secret.

Preuve : *(impénétrable ⇒ secret)*
La connaissance intiale I de l'intrus ne contient pas de données secrètes, puisque par hypothèse les données dont on peut demander le secret sont uniquement des nonces engendrés au cours du protocole. Les messages contenus dans l'historique ne compromettent pas les secrets, puisqu'ils sont dans le coidéal. Ce dernier étant fermé par analyse et synthèse, les messages calculés par l'intrus resteront dans le coidéal.

En fait, montrer qu'un protocole est impénétrable consiste à vérifier que pour tout historique H accessible ne compromettant aucun secret, alors les évènements ajoutés par les successeurs directs de H ne compromettent pas non plus de secret.

Autrement dit, il s'agit de montrer que le protocole préserve la confidentialité :

Définition 4.3 *(protocole préserve la confidentialité) (cf. définition 4.17 [14])*
Un protocole P préserve la confidentialité si :
- *pour tout ensemble de messages I représentant la connaissance initiale de l'intrus,*
- *pour tout historique H accessible,*
- *pour toute spécification C de H,*
- *pour toute transition t applicable à H,*

on a

$$Msg(Post(t)) \subseteq Coideal(Sec(C))$$

Théorème 4.1 *[25]*
Un protocole P est impénetrable si et seulement si P préserve la confidentialité.

Preuve : *(confidentiel ⇒ impénetrable)*
C'est l'implication dont on a besoin pour prouver la correction de l'algorithme et la preuve se fait par induction sur la taille de l'historique :
- si on ajoute un message dans l'historique par une étape malhonnête, il s'agit donc d'un message construit par l'intrus à partir de messages qui se trouvaient dans le coidéal, le message ajouté se trouve donc également dans le coidéal et ne compromet aucun secret.

- si on ajoute un message qui provient d'une transition honnête, autrement dit, on a joué une règle du protocole, alors puisque le protocole préserve la confidentialité, le message ajouté ne compromet aucun secret.
- si on ajoute une spécification dans l'historique (par une étape honnête), celle-ci demande le secret d'un nonce fraîchement engendré, et donc aucun message de l'historique ne peut compromettre ce « nouveau » secret.

Remarque : Par définition d'une transition, on ne peut pas avoir à la fois des messages et des spécifications dans le *Post* d'une règle.

4.1.2 L'algorithme

Décomposition en branches

On a donc besoin de tester si un message $m \in Coideal(S)$ où S est un ensemble de messages atomiques.

A chaque nonce d'un message m, on associe l'ensemble des clefs nécessaires au déchiffrement du nonce, par une fonction notée $branch(m)$.

Exemple 4.4
Si $m = \{A, N_a\}_{\mathsf{pub}(B)}$, *on a* $branch(m) = \{(N_a, \{prv(B)\})\}$
Si $m = \{N_a, N_b, B\}_{\mathsf{pub}(A)}$, *on a* $branch(m) = \{(N_a, \{prv(A)\}), (N_b, \{prv(A)\})\}$

La proposition suivante permet de ramener la question de savoir si un message m est dans le $Coideal(S)$ a un test sur chacune des branches de m.

Proposition 4.1 *(cf. proposition 7.1 [14])*
$m \in Coideal(S)$ *si et seulement si pour toute branche* $(z, K_s) \in branch(m)$ *:*

$$z \notin S \text{ ou } K_s \cap S \neq \emptyset$$

Tests élementaires

Le but est de trouver des conditions suffisantes sur les branches $b = (z, K_s)$ de m (où m est un message envoyé) pour montrer que $m \in Coideal(Sec(C))$ pour toute spécification C pouvant se trouver dans un historique accessible. Ces tests sont au nombre de trois.

Soit $b = (z, K_s)$ une branche de m, E_s un ensemble d'évènements (de l'historique H), N_s un ensemble de messages atomiques « frais », (c'est à dire qui n'apparaissent pas dans H), alors on dit que l'on peut conclure à vrai par test élémentaire pour la branche b si l'on est dans un des 3 cas suivants :

New : si $z \in N_s$. En effet, puisque le nonce z est frais, aucun message de l'historique ne peut le compromettre. Actuellement, ce test ne s'applique pas après avoir fait une recherche en arrière.

Almost in : s'il existe un ensemble de clefs K'_s tel que $K'_s \subseteq K_s$ et $(z, K'_s) \in branch(Msg(E_s))$. Ce test permet de conclure lorsque le nonce z, dont on veut montrer qu'il n'a pas été compromis à cette étape du protocole a déjà circulé dans un message en étant moins protégé qu'actuellement.

Secrets disjoints : s'il existe une spécification $C \in E_s$ telle que $z \in SpecSec(C)$ et $K_s \cap Sec(C) \neq \emptyset$. Ce test permet de conclure, lorsqu'une spécification de l'historique H parle du nonce z. En effet, compte tenu de la forme des transitions, il est impossible d'avoir un historique contenant deux spécifications sur un même nonce. La propriété que l'on doit vérifier pour toute spécification et donc trivialement vraie lorsque la spécification est différente de C. Maintenant, pour la spécification C, la proposition est vérifiée, puisque z est protégé par (au moins) une clef dont l'inverse est secrète.

Procédure de recherche en arrière

La procédure de recherche en arrière repose sur le fait que tout message présent dans un historique a été produit par le jeu d'une étape honnête (en jouant une règle du protocole) ou malhonnête (en ajoutant à l'historique un message construit par l'intrus).

Dans les deux cas, on obtient des informations supplémentaires sur le contenu de l'historique considéré et on peut appliquer à nouveau les tests élèmentaires aux évènements obtenus.

Un arbre de preuve est étiqueté par la valeur courante de l'ensemble E_s, ensemble des évènements dont on sait qu'il font partie de l'historique, et chaque arête est étiquetée par la fonction intermédiaire appelée :
- *honest* (étape honnête) ou $fake$ (étape malhonnête) entre deux procédures de recherche,
- *secrets disjoints* (*ds*), *almost in* ou *new* lorsqu'un test élémentaire permet de conclure.

Exemple 4.5
L'ensemble des arbres de preuve pour chacune des branches de chaque règle du protocole de Needham-Schroeder-Lowe est représenté à la figure 4.2. On en déduit que le protocole de Needham-Scroeder-Lowe préserve la confidentialité et donc a fortiori le secret.

4.1.3 Amélioration de Securify

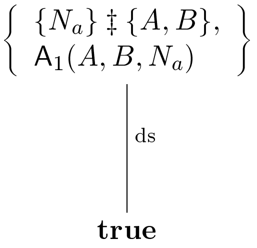
Il est à noter que le secret est indécidable pour la classe des protocoles considérée par cet outil, la méthode de preuve ainsi obtenue ne peut donc pas être complète. Lorsque la procédure termine avec la valeur $false$, cela ne signifie pas que le protocole ne préserve pas secret mais seulement que la procédure de preuve du secret a échoué.

Dans certains cas, l'arbre d'échec de la preuve fournit des informations qui peuvent permettre de trouver une véritable attaque sur le protocole, mais la procédure peut également échouer sans qu'on puisse reconstruire une attaque.

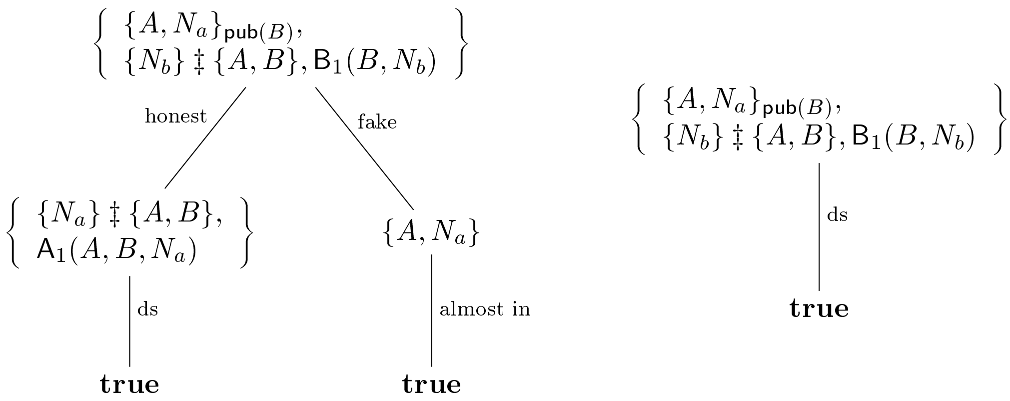
Une première raison est qu'un protocole peut préserver le secret sans être impénétrable (la réciproque de la proposition 4.1 n'est pas vraie). Ainsi, la procédure échoue à montrer le secret du protocole de Yahalom (figure 4.3) car celui-ci n'est pas impénétrable : le dernier message envoyé $\{N_b\}_{K_{ab}}$ n'est pas dans le coidéal de $\{N_b\} \ddagger \{A, B\}$.

↪ Première et deuxième règles (4.1) et (4.2) : pas de branche, il n'y a rien à vérifier.

↪ Troisième règle (4.3) : $\mathsf{branch}(\{A, N_a\}_{\mathsf{pub}(B)}) = \{(N_a, \{\mathsf{prv}(B)\})\}$.



↪ Quatrième règle (4.4) : $\mathsf{branch}(\{N_a, N_b, B\}_{\mathsf{pub}(A)}) = \{(N_a, \{\mathsf{prv}(A)\}), (N_b, \{\mathsf{prv}(A)\})\}$.



↪ Cinquième règle (4.5) : $\mathsf{branch}(\{N_b\}_{\mathsf{pub}(B)}) = \{(N_b, \{\mathsf{prv}(B)\})\}$.

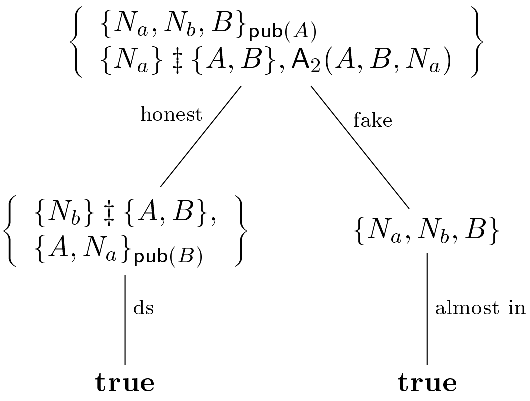


FIGURE 4.2 - *Preuve du secret du protocole de Needham-Schroeder-Lowe.*

$$\begin{aligned}\textbf{Yahalom :}\quad & 1.\ A \to B : A, N_a \\ & 2.\ B \to S : B, \{A, N_a, N_b\}_{\mathsf{shr}(B)} \\ & 3.\ S \to A : \{B, K_{ab}, N_a, N_b\}_{\mathsf{shr}(A)}, \{A, K_{ab}\}_{\mathsf{shr}(B)} \\ & 4.\ A \to B : \{A, K_{ab}\}_{\mathsf{shr}(B)}, \{N_b\}_{K_{ab}}\end{aligned}$$

FIGURE 4.3 - *le protocole de Yahalom*

Une deuxième raison est le manque de tests pertinents pour les données dont on ne demande pas le secret dans la spécification. Ainsi considérons le protocole suivant où l'on demande le secret du nonce N_2 :

$$\begin{array}{lll} A & \to B : & A, \{N_1, N_2\}_{\mathsf{pub}(B)} \\ B & \to A : & \{N_1\}_{N_2} \end{array}$$

Le secret du nonce est clairement préservé et pourtant la procédure échoue (voir figure 4.4). Une perspective pour l'amélioration de Securify serait de permettre de conclure à vrai lorsque le nonce de la branche a été engendré sans spécification de secret.

L'amélioration de Securify que je propose passe par la généralisation des tests élémentaires présentés à la section 4.1.2. Les nouveaux tests permettront de conclure à vrai sur plus de branches.

De plus, on se donne la possibilité de regrouper les spécifications de l'historique qui concerne les mêmes agents. La notion de confidentialité prouvée sera alors plus faible, mais elle implique également le secret du protocole.

Notation : Soit H un historique, on note $\widetilde{H}$ l'historique H dans lequel on autorise le regroupemenent des spécifications. Ainsi $S \ddagger L \in \widetilde{H}$ s'il existe $S_1 \ddagger L$, ... , $S_n \ddagger L$ dans H tels que $S_1 \cup S_2 ... \cup S_n = S$.

Généralisation du test New

Soit t une transition, soit $b = (z, K_s)$ une branche d'un message de $Post(t)$, E_s un ensemble d'évènements de l'historique H, N_s un ensemble de messages atomiques « frais », (c'est à dire qui n'apparaissent pas dans H), alors on peut conclure à vrai sur la branche b par le test $\widetilde{New}$ si $z \in N_s$, et si pour toute spécification $C \in Pos(t)$, $z \notin SpecSec(C)$. Cela ne signifie pas que le nonce z est secret, mais simplement que l'on n'a pas demandé le secret de ce nonce.

Exemple 4.6
Ce nouveau test va permettre de conclure au secret du protocole présenté à la figure 4.4.

$$\begin{array}{lll} A & \to B : & A, \{N_1, N_2\}_{\mathsf{pub}(B)} \\ B & \to A : & \{N_1\}_{N_2}. \end{array}$$

Protocole dans le modèle Millen-Rueß (nous ne précisons pas les changements d'états pour alléger les notations).

$$\begin{aligned}
\emptyset &\xrightarrow{\{N_2\}} \{\{N_2\} \ddagger \{A,B\}\} \\
\{\{N_2\} \ddagger \{A,B\}\} &\xrightarrow{\{N_1\}} \{A, \{N_1, N_2\}_{\mathsf{pub}(B)}\} \\
\{A, \{N_1, N_2\}_{\mathsf{pub}(B)}\} &\xrightarrow{\emptyset} \{\{N_1\}_{N_2}\}
\end{aligned}$$

On considère l'unique branche $(N_1, \{N_2\})$ du message $\{N_1\}_{N_2}$ émis à la dernière règle.

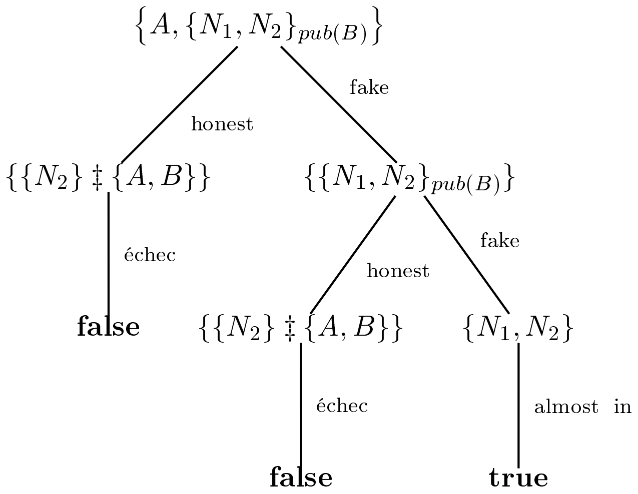


Figure 4.4 - *Exemple de fausse attaque.*

L'ensemble des arbres de preuve pour chacune des branches de chaque règle du protocole est représenté à la figure 4.5. On en déduit que ce protocole préserve le secret de N_2.

$\hookrightarrow$ Première règle : pas de branche, il n'y a rien à vérifier.

$\hookrightarrow$ Deuxième règle : $\mathsf{branch}(A, \{N_1, N_2\}_{\mathsf{pub}(B)}) = \{(N_1, \{\mathsf{prv}(B)\}), (N_2, \{\mathsf{prv}(B)\})\}$.

$\hookrightarrow$ Troisième règle : $\mathsf{branch}(\{N_1\}_{N_2}) = \{(N_1, \{N_2\})\}$.

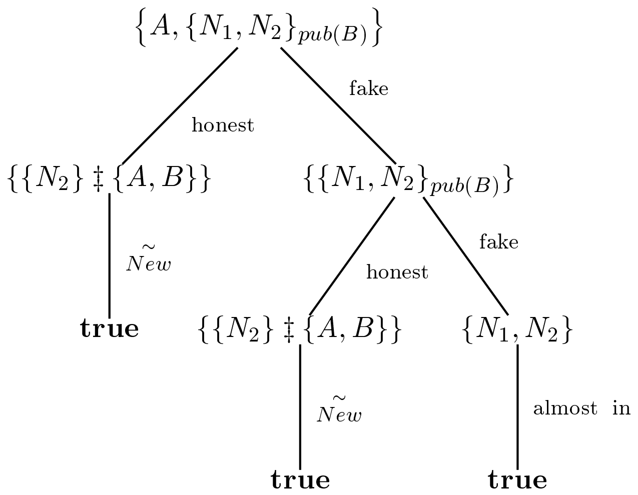


FIGURE 4.5 - *Preuve du secret du protocole de l'exemple 4.6.*

Généralisation du test Secrets disjoints

Soit $b = (z, K_s)$ une branche, E_s un ensemble d'évènements de l'historique H, N_s un ensemble de messages atomiques « frais », (c'est à dire qui n'apparaissent pas dans H), alors on peut conclure à vrai sur la branche b par le test $\widetilde{ds}$, s'il existe une spécification $C \in \widetilde{E_s}$ telle que $z \in SpecSec(C)$ et $K_s \cap Sec(C) \neq \emptyset$.

Exemple 4.7

Ce nouveau test va permettre de conclure sur la dernière branche de la dernière du protocole de Yahalom 4.3 dont la traduction dans le modéle Millen-Rueß est présentée à la figure 4.6.

L'arbre de preuve pour la branche $(N_b, \{K_{ab}\})$ est représenté sur les figures 4.7, 4.8, 4.9, 4.10.

Cette preuve permet de conclure au secret du protocole de Yahalom.

Protocole dans le modèle Millen-Rueß (nous ne précisons pas les changements d'états pour alléger les notations).

$$\emptyset \xrightarrow{N_a} \{< A, N_a >\} \tag{4.6}$$

$$\left\{ \begin{array}{l} \{N_b\} \ddagger \{A, B\}, \\ < A, N_a > \end{array} \right\} \xrightarrow{\emptyset} \left\{ < B, \{A, N_a, N_b\}_{K_{bs}} > \right\} \tag{4.7}$$

$$\left\{ \begin{array}{l} \{K_{ab}\} \ddagger \{A, B\}, \\ < B, \{A, N_a, N_b\}_{K_{bs}} > \end{array} \right\} \xrightarrow{\emptyset} \left\{ \begin{array}{l} < \{B, K_{ab}, N_a, N_b\}_{K_{as}} >, \\ \{A, K_{ab}\}_{Kbs} \end{array} \right\} \tag{4.8}$$

$$\left\{ < \{B, K_{ab}, N_a, N_b\}_{K_{as}}, X > \right\} \xrightarrow{\emptyset} \left\{ < X, \{N_b\}_{K_{ab}} > \right\} \tag{4.9}$$

FIGURE 4.6 - *Traduction du protocole de Yahalom dans le modèle de Millen-Rueß.*

On considère la deuxième branche $(N_b, \{K_{ab}\})$ du message $\langle X, \{N_b\}_{K_{ab}} \rangle$ émis à la dernière règle.

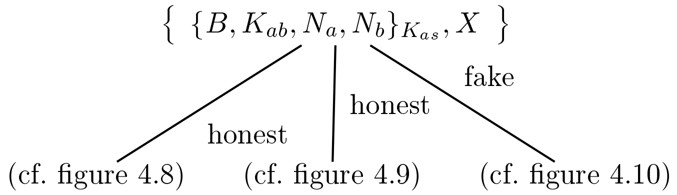


FIGURE 4.7 - *Preuve du secret pour la branche $(N_b, \{K_{ab}\})$ du protocole de Yahalom.*

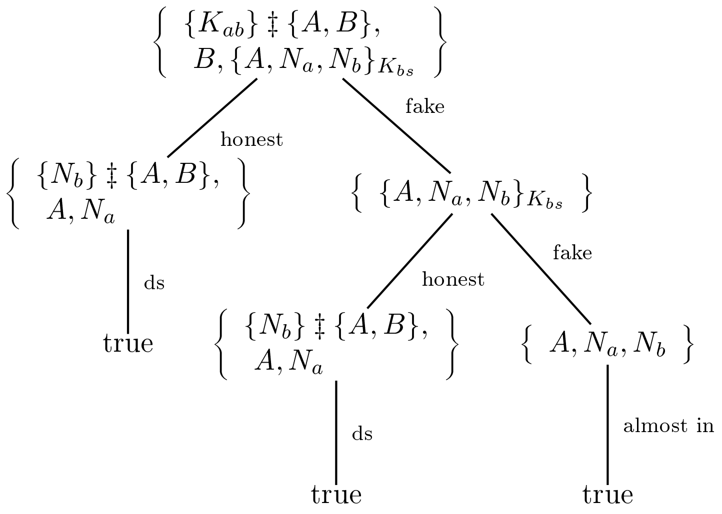


FIGURE 4.8 - *Sous-arbre de preuve pour la branche $(N_b, \{K_{ab}\})$ du protocole de Yahalom.*

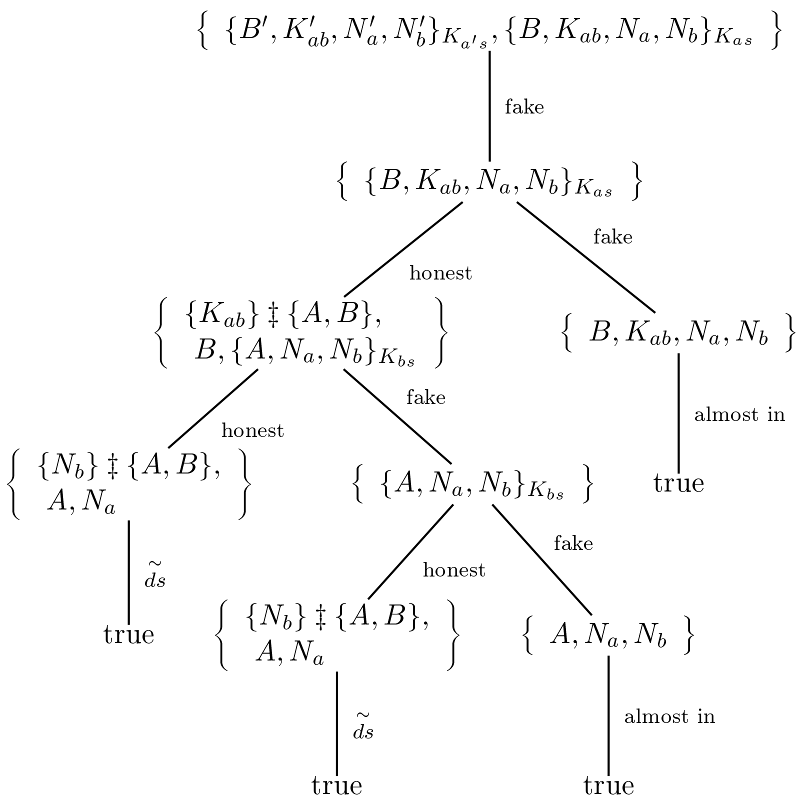


FIGURE 4.9 - *Sous-arbre de preuve pour la branche $(N_b, \{K_{ab}\})$ du protocole de Yahalom.*

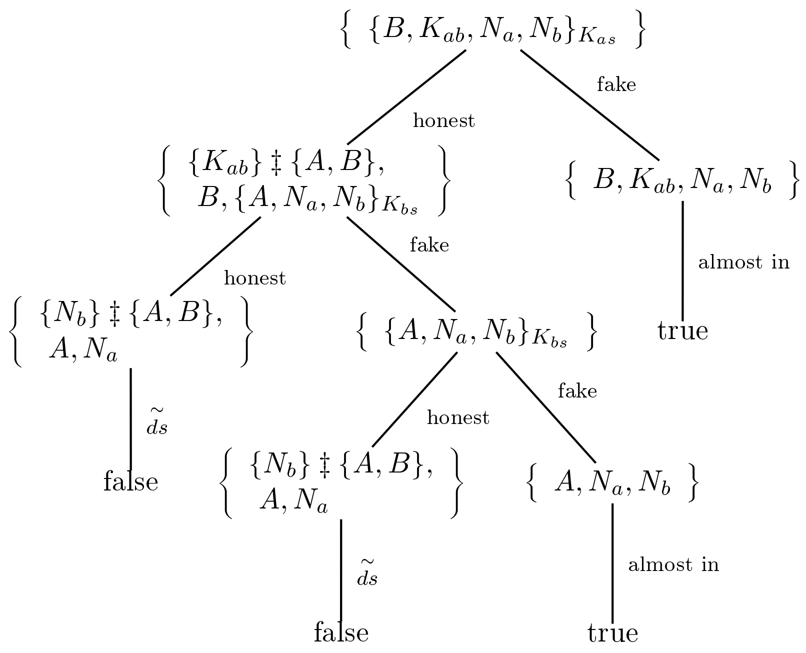


FIGURE 4.10 - *Sous-arbre de preuve pour la branche $(N_b, \{K_{ab}\})$ du protocole de Yahalom.*

Généralisation du test Almost in

Soit $b = (z, K_s)$ une branche, E_s un ensemble d'évènements de l'historique H, N_s un ensemble de messages atomiques « frais », (c'est à dire qui n'apparaissent pas dans H), alors on peut conclure à vrai sur la branche b par le test $\widetilde{al\ in}$ s'il existe une spécification $C \in \widetilde{E_s}$ telle que $z, N \in SpecSec(C)$, et un ensemble K'_s tel que $K'_s \subseteq K_s$ et $(N, K'_s) \in branch(M)$ pour un message M dans E_s.

Autrement dit, z et N sont dans une même spécification, N a dejà circulé sur le réseau en étant protégé par moins de clés que z. Donc on en conclut que z est suffisament protégé. (sinon N n´était pas suffisament protégé non plus)

Exemple 4.8
Le protocole (informel) de Needham-Schroeder-Lowe suivi de l'échange d'un secret s :

$$\begin{array}{lll} A & \to B: & \{A, N_a\}_{\mathsf{pub}(B)} \\ B & \to A: & \{N_a, N_b, B\}_{\mathsf{pub}(A)} \\ A & \to B: & \{N_b\}_{\mathsf{pub}(B)}, \{s\}_{N_b} \end{array}$$

L'arbre de preuve pour la dernière branche de la dernière règle du protocole est représenté à la figure 4.11.

$\hookrightarrow$ Troisième règle : $\mathsf{branch}(\{s\}_{N_b}) = \{(s, \{N_b\})\}$.

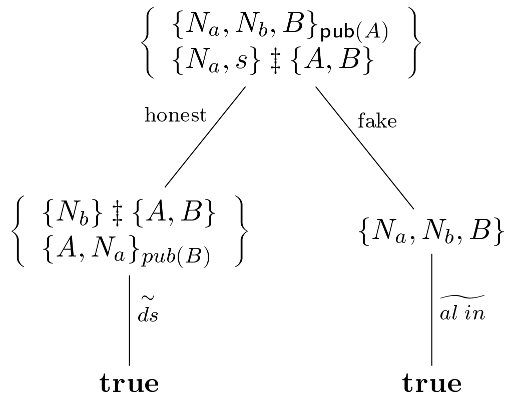


FIGURE 4.11 - *Preuve du secret pour la branche* $\{s\}_{N_b}$ *du protocole de l'exemple 4.8.*

4.1.4 Limites de Securify

L'outil Securify fut l'un des premiers à *prouver* des propriétés de secret dans le cadre d'un nombre de sessions non borné et il a permis de vérifier de nombreux protocoles. L'amélioration de cet outil par l'ajout de nouveaux « tests de base » permet de garantir une meilleure terminaison et d'éviter de trop nombreuses « fausses attaques ».

Cependant pour son bon fonctionnement, il faut que les protocoles soient suffisament typés. De plus, il serait intéresssant de pouvoir modifier l'algorithme afin de prendre en compte une classe plus large de protocoles, (comme par exemple, la suppression de la restriction sur les clefs atomiques), et de considérer d'autres modèles d'intrus que celui dit de *Dolev-Yao*.

Mais l'étude de cet outil a révélé que le modèle de déduction de l'intrus et l'algorithme de recherche étaient intiment liés et ne pouvaient pas être dissociés facilement.

4.2 Prototype : une procédure modulaire de recherche d'attaques

Afin d'expérimenter les résultats trouvés au chapitre 3, nous avons donc choisi de développer une procédure de recherche d'attaque, simple mais modulaire ainsi que les modules d'intrus correspondant à la théorie de *Dolev-Yao* standard et à la théorie de *Dolev-Yao* étendue aux données faibles.

Concrètement, nous partons d'un protocole exprimé dans une syntaxe abstraite où les transitions sont explicitées pour chaque participant du protocole, et nous considérons un nombre borné de sessions. Nous cherchons, en considérant tous les entrelacements possibles des messages des différentes sessions du protocole à trouver une attaque.

Cette procédure a été implémentée en *Ocaml* (environ 1800 lignes). Les comportements de l'outil sont de deux types :

- réponse « oui » : une attaque a été trouvée pour le module d'intrus passé en paramètre et une description de l'attaque est alors fournie.
- réponse « non » : aucune attaque n'a été trouvée.

4.2.1 L'algorithme

Comme énoncé en introduction, la recherche d'attaque est effectuée pour un modèle d'intrus fourni en paramètre. Nous allons décrire les deux modules d'intrus qui ont été implémentés ainsi que l'algorithme de recherche.

Les modules d'intrus

Les deux modules d'intrus correspondent aux systèmes d'inférence des figures 3.1 et 3.2 et ont une interface commune composée d'une unique fonction implémentant le *problème de déduction de l'intrus* dans le cas non clos (les messages considérés peuvent contenir des variables). Seule cette fonction peut donc être appelée par l'algorithme de recherche.

Etant donné un ensemble de termes T correspondant à la connaissance de l'intrus et un terme s, cette fonction retourne une liste de substitutions closes l telle que si $\sigma \in l$ alors $T\sigma \vdash s\sigma$. Cette fonction simule la réception d'un message par un agent A : si s est un pattern de message attendu par A, alors $\{s\sigma \mid \sigma \in l\}$ correspond à l'ensemble des messages que l'intrus pourra faire accepter à A. Pour ce faire, on regarde toutes les substitutions possibles et l'on résoud le *problème de déduction de l'intrus* dans le cas clos. Afin d'éviter une explotion du nombre de substitutions à considérer, on se restreint à la substitution des variables par des constantes.

Ensuite, pour chacun des modèles d'intrus a été implémenté le *problème de déduction de l'intrus* dans le cas clos (T et s clos).

Le modèle standard de Dolev-Yao

Pour ce faire, on essaie par recherche arrière de reconstituer une dérivation pouvant conduire au terme s. Autrement dit, on considère tous les instances de règles qui ont pour conclusion s, puis on essaie de voir comment dériver les prémisses de l'instance de la règle considérée, et ainsi de suite. Grâce au résultat de localité (section 3.2), on sait que s'il existe une dérivation de s, alors il y a une dérivation qui n'utilise que des sous-termes des termes de $T \cup \{s\}$. Donc dès qu'un terme n'appartient pas à cet ensemble, il est inutile de considérer plus longtemps cette dérivation. Après avoir exploré toutes les dérivations ne faisant intervenir que des sous-termes des termes de $T \cup \{s\}$, on peut alors répondre si oui ou non s est déductible par l'intrus à partir des messages contenus dans T.

Le modèle de Dolev-Yao étendu

Dans ce module, on trouve tout d'abord une fonction permettant de savoir si une donnée faible w peut être déduite par l'intrus. Pour ce faire, on regarde tous les vérifieurs possibles (celui-ci est forcément un sous-terme de $T \cup T' \cup \{s\}$ grâce au résultat de localité) et l'on recherche toutes les preuves en forme normale permettant de dériver ce terme d'une part en sachant que w est faible (soit n_1 le nombre de preuves ainsi obtenu) et d'autre part sans cette hypothèse (soit n_2 le nombre de preuves obtenu). S'il est possible de faire deux preuves différentes d'un vérifieur v dont l'une au moins dépend de w (autrement dit, si $n_1 \geq 2$ et $n_2 - n_1 \geq 1$), alors on déduit que l'application de la règle Comparaison est possible et donc que w peut être ajouté à la connaissance de l'intrus. Après avoir statué pour les différentes données faibles, on utilise la fonction du premier module permettant de résoudre le *problème de déduction de l'intrus* dans le cas du modèle de Dolev-Yao standard.

La procédure de recherche en avant

La procédure de recherche consiste en une recherche en profondeur d'abord où l'on considère tous les entrelacements possibles des différentes sessions données dans la spécification. Cette procédure s'arrête soit lorsque l'intrus a obtenu suffisament d'informations pour dériver le secret, soit lorsque tous les entrelacements possibles ont été explorés.

4.2.2 Implémentation

L'algorithme ainsi obtenu a été implémenté en *Ocaml*.

La structure du prototype est résumée à la figure 4.12. Le prototype admet en entrée un fichier .eva contenant la spécification d'un protocole dans le langage *eva*. Le langage *eva* est l'une des contributions du projet RNTL EVA. Ce dernier a consisté, dans un premier temps, à élaborer un langage commun de spécification des protocoles cryptographiques avec deux versions du langage [19] :

- le langage concret destiné aux utilisateurs, où la description des protocoles est proche de celle utilisée dans [9] et des exemples donnés dans ce rapport,
- le langage abstrait, destiné aux outils, où les transitions sont explicitées pour chaque participant du protocole.

De plus, il existe un traducteur automatique du langage concret (fichier .eva) vers le langage abstrait. Pour traiter un protocole décrit en *eva*, notre prototype commence par utiliser le traducteur pour obtenir la spécification du protocole en syntaxe abstraite.

Les messages sont échangés sur des canaux de communication non fiables, on peut donc supposer que :

- tout message envoyé est envoyé à l'intrus,
- tout message reçu a été émis par l'intrus.

Ainsi, lorsqu'un message m est envoyé sur le réseau ($!m$), on peut considérer que celui-ci est intercepté par l'intrus qui l'ajoute à sa connaissance. Lorsqu'un agent est en attente d'un message $?m$ (ce message peut contenir des variables), on peut considérer que ce message a été émis par l'intrus, la procédure interroge donc le module d'intrus pour savoir quelles sont les substitutions σ telles que le message $m\sigma$ soit déductible de l'intrus.

Exemple 4.9
Nous reprenons l'exemple 3.1 du protocole de vote présenté en introduction. La spécification de ce protocole en eva telle qu'elle peut être donnée en entrée au prototype, est présentée à la figure 4.13.

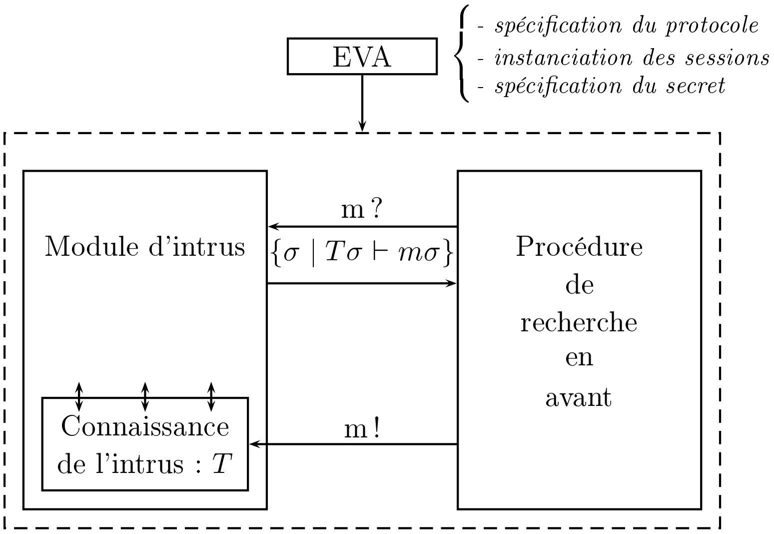


FIGURE 4.12 - *Structure du prototype.*

```
Vote

constant keypair PK, SK (principal):number

parameter S, A : principal
parameter fresh N: weak



S knows S, PK, SK(S)
A knows A, S, N, PK

{
  1. A -> S : {N}_PK(S)
}

value a, s: principal
value na:number


s1. session (A=a; S=s; N=na)

claim secret(na)
```

FIGURE 4.13 - **Fichier** *Vote.eva*

4.2.3 Description des protocoles testés

Dans cette section, nous décrivons les différents protocoles sur lesquels le prototype a été testé. Les exemples choisis permettent de mettre en évidence des solutions pour éviter les « attaques par prédictions » : utiliser un nonce pour ne pas permettre à l'intrus de construire un message (échange de Gong), utiliser un chiffrement asymétrique pour ne pas permettre à l'intrus de déchiffrer un message même s'il a obtenu des informations sur la clef de chiffrement (protocole de Bellovin et Merritt).

Vote

$$A \rightarrow S: \quad \{m\}_{pub(S)}$$

avec m donnée de type faible.

Alice chiffre son vote m avec la clef publique du serveur S. Lorsque le serveur reçoit le message chiffré, il le déchiffre avec sa clef privée $priv(S)$. On demande que m soit un secret partagé uniquement entre A et S.

Challenge

$$\begin{array}{ll} A \rightarrow B: & \{N\}_{K_{ab}} \\ B \rightarrow A: & \{N+1\}_{K_{ab}} \end{array}$$

avec K_{ab} clef symétrique de type faible.

Alice engendre un nonce N, le chiffre avec la clef K_{ab} qu'elle partage avec Bob. Lorsque Bob reçoit ce message, il le déchiffre, incrémente la valeur obtenue et retourne le résultat chiffré avec la clef K_{ab}. On demande le secret du nonce N.

Protocole de Andrew Secure RPC

$$\begin{array}{ll} A \rightarrow B: & A, \{N_a\}_{K_{ab}} \\ B \rightarrow A: & \{N_a+1, N_b\}_{K_{ab}} \\ A \rightarrow B: & \{N_b+1\}_{K_{ab}} \\ B \rightarrow A: & \{K'_{ab}, N'_b\}_{K_{ab}} \end{array}$$

avec K_{ab} clef symétrique de type faible.

Ce protocole a pour but de générer une nouvelle clef de session K'_{ab} entre les agents A et B. Si la clef K_{ab} est de type faible alors ce protocole est sujet a une «attaque par prédictions». L'intrus peut faire deux preuves différentes du vérifieur N_a et en déduire la valeur de K_{ab}. Et, par simple déchiffrement du dernier message, il récupère alors la clef K'_{ab}.

Clefs asymétriques

$$A \quad \to B: \quad pub(B), \{N\}_{pub(B)}$$

avec $priv(B)$ clef de type faible.

Ce protocole permet d'illustrer le cas d'application de la règle Comparaison, lorsque le vérifieur est une paire de clefs asymétriques.

Echange de Gong

$$\begin{aligned} A \quad &\to B: \quad \{C, N\}_{pub(B)} \\ B \quad &\to A: \quad \{N+1\}_{K_{ab}} \end{aligned}$$

avec K_{ab} clef de type faible.

Alice génére deux nonces C et N. N est un nonce dont on demande le secret alors que C a pour unique but d'éviter une « attaque par prédictions ». L'intrus peut (à partir du deuxième message) calculer un ensemble fini de valeurs possibles pour le nonce N mais il ne peut pas utiliser le premier message pour déterminer la valeur exacte de N. En effet, il ne peut pas déchiffrer le premier message puisqu'il ne connaît pas la clef privée de B et il ne peut pas esprérer construire ce premier message puisque C est un nombre aléatoire.

Echange de Gong modifié

$$\begin{aligned} A \quad &\to B: \quad \{N\}_{pub(B)} \\ B \quad &\to A: \quad \{N+1\}_{K_{ab}} \end{aligned}$$

avec K_{ab} clef de type faible.

Ce protocole illustre l'intérêt du nonce C. En effet, après une session de ce protocole, l'intrus est en mesure de récupérer le nonce N. Il peut en effet faire deux preuves du vérifieur N et en déduire la valeur de la clef K_{ab}. En déchiffrant le deuxième message, il récupère alors la valeur du nonce N.

Protocole de Bellovin et Meritt

$$\begin{aligned} A \quad &\to B: \quad A, \{pk\}_{pw} \\ B \quad &\to A: \quad \{\{K_{ab}\}_{pk}\}_{pw} \\ A \quad &\to B: \quad \{N_a\}_{K_{ab}} \\ B \quad &\to A: \quad \{N_a, N_b\}_{K_{ab}} \\ A \quad &\to B: \quad \{N_b\}_{K_{ab}} \end{aligned}$$

avec pw mot de passe faible partagé entre A et B.

Ce protocole utilise un mot de passe faible pw et permet aux agents A et B de se mettre d'accord sur une clef K_{ab}. Alice génére une paire de clef asymétrique ($priv$, pk) et envoie pk à Bob chiffré avec le mot de passe faible pw. Bob génére la clef de session K_{ab} et l'envoie à Alice dans le message $\{\{K_{ab}\}_{pk}\}_{pw}$. Ensuite, trois messages permettent à Alice et Bob de s'authentifier. Il est important que pk soit une clef asymétrique, sinon ce protocole serait sujet à une « attaque par prédictions ».

Protocole de Bellovin et Meritt modifié

$$\begin{array}{lll} A & \rightarrow B: & A, \{symkey\}_{pw} \\ B & \rightarrow A: & \{\{K_{ab}\}_{symkey}\}_{pw} \\ A & \rightarrow B: & \{N_a\}_{K_{ab}} \\ B & \rightarrow A: & \{N_a, N_b\}_{K_{ab}} \\ A & \rightarrow B: & \{N_b\}_{K_{ab}} \end{array}$$

avec pw mot de passe faible partagé entre A et B. On suppose qu'Alice génére une clef symétrique $symkey$. Ce protocole est alors vulnérable, une « attaque par prédictions » est possible. L'intrus peut en effet déduire faiblement $symkey$ (grâce au message 1), puis K_{ab} (grâce au message 2). C'est cette dernière étape qui n'était pas possible dans le protocole précédent. Et enfin, il peut de deux façons différentes obtenir le nonce N_a (messages 3 et 4) ou N_b (messages 4 et 5), ce qui lui permet de vérifier si son choix était correct. Il en déduit donc la valeur de pw et par déchiffrement du message 1 puis 2, la valeur de la clef K_{ab}.

4.2.4 Résultats

Les résultats sont récapitulés dans le tableau ci-dessous. Ils ont été obtenus avec un Pentium III, 933 Mhz, 256 Mo de RAM.

Protocole	**Modèle d'intrus**			
	Dolev-Yao standard		Dolev-Yao étendu	
	secret	temps (ms)	secret	temps (ms)
Vote	oui	20	non	20
Challenge	oui	20	non	20
Andrew Secure RPC	oui	110	non	60
Clefs asymétriques	oui	20	non	30
Echange Gong	oui	60	oui	310
Echange Gong modifié	oui	30	non	70
Bellovin et Merritt	oui	50	oui	220
Bellovin et Merritt modifié	oui	50	non	110

Les spécifications en *eva* des protocoles cités dans ce tableau sont données dans l'annexe A. La sortie de ce prototype permet dans le cas où une attaque a été trouvée de visualiser l'entrelacement permettant d'obtenir le secret.

Chapitre 5

Conclusion et perspectives

La vérification des protocoles cryptographiques se heurte à plusieurs difficultés : la modélisation des protocoles et des propriétés, la modélisation des capacités de l'intrus, la mise en oeuvre pratique des méthodes de vérification, ...

Nous avons, au cours de ce stage, apporté une contribution sur le plan de la modélisation des capacités de l'intrus en étendant le modèle de *Dolev-Yao*, mais aussi sur le plan de la vérification pratique des protocoles en améliorant l'outil Securify et en développant un prototype de recherche d'attaque modulaire.

Modélisation

Une faiblesse des outils existants actuellement est de reposer systématiquement sur l'*hypothèse du chiffrement parfait* et de ne considérer qu'un seul modèle d'intrus, celui de *Dolev-Yao*.

En tenant compte du fait que certaines données prennent leurs valeurs dans un domaine fini connu de l'intrus, on a pu prendre en compte un nouveau genre d'attaques, dites « par prédiction », qui n'était pas étudié de manière formelle jusqu'à présent. La formalisation du pouvoir de déduction de l'intrus par un système de règles d'inférence et le résultat de décidabilité obtenu pour le *problème de déduction de l'intrus* dans ce modèle étendu, permet de détecter ce genre d'attaque de façon plus systématique.

Mais l'affaiblissement de l'*hypothèse du chiffrement parfait* ne se borne pas à la prise en compte des « attaques par prédictions ».

Les protocoles de distribution de clefs s'appuient sur des propriétés d'associativité et de commutativité de certains opérateurs de chiffrement, et de certaines primitives cryptographiques, telles que le ou exclusif. Un autre exemple est donné par les protocoles de vote, qui satisfont cette fois une propriété d'homomorphisme, le chiffré d'une somme étant égal à la somme des chiffrés. L'étude de ce genre de protocoles nécessite la prise en compte des propriétés algébriques des primitives cryptographiques utilisées.

Définir et étudier d'autres modèles d'intrus apparaît donc comme un élèment déterminant pour l'étude des protocoles.

Vérification pratique

L'outil Securify que nous avons décrit à la section 4.1 fut l'un des premiers à *prouver* des propriétés de secret et il a permis de vérifier un certain nombre de protocoles. Les améliorations apportées en ajoutant de nouveaux « tests de base » permet d'éviter de trop nombreuses « fausses attaques ». Cependant, les restrictions prises sur les protocoles traités (pas de clefs composées par exemple) ne sont peut-être pas nécessaires. Cet outil pourrait peut-être être étendu pour traiter une classe plus large de protocoles. En revanche, étendre cet outil pour prendre en compte d'autres modèles d'intrus que celui de *Dolev-Yao* semble impossible sans faire une modification profonde de l'algorithme de recherche de preuve.

Le prototype de recherche d'attaques a été implémenté dans le but d'expérimenter le modèle de *Dolev-Yao* étendu aux données de type faible. Cependant cet outil est modulaire et pourra donc être complété en implémentant dans un nouveau module le *problème de déduction de l'intrus* pour d'autres modèles d'attaquants qui nous seront apparus pertinents.

Annexe A

Description des protocoles testés par le prototype

L'objet de cette annexe est de présenter les spécifications *eva* manquantes des protocoles mentionnés dans la section 4.2.

```
Challenge

 constant succ (number) : number

 parameter A, B: principal
 parameter fresh Na : number
 parameter Kab: weak


 A knows A, B, Na, Kab, succ
 B knows A, B, Kab, succ

 {

   1. A -> B : {Na}_Kab
   2. B -> A : {succ(Na)}_Kab
 }


value a, b: principal
value na, kab:number

s1. session (A=a; B=b; Na=na; Kab= kab)

claim secret(na)
```

FIGURE A.1 - **Fichier** *Challenge.eva*

```
Clefs_asymetriques

parameter keypair K1, K2: weak
parameter A, B : principal
parameter fresh Na : number

A knows A, B, K1, Na
B knows A, B, K1, K2

{
  1. A -> B :  K1(),{Na}_K1()
}

value a, b : principal
value na, k1, k2: number


s1. session (A=a; B=b; Na=na; K1=k1; K2=k2)

claim secret (na)
```

```
 Andrew_Secure_RPC

 constant succ (number) : number

 parameter A, B : principal
 parameter fresh Na, fresh Nb, fresh Nb1 : number
 parameter fresh Kab1 : number
 parameter Kab: weak

 A knows A, B, Kab, Na, succ
 B knows B, Kab, Nb, Kab1, Kab, Nb1, succ


 {
   1. A -> B :  A, {Na}_Kab
   2. B -> A : {succ(Na), Nb}_Kab
   3. A -> B : {succ(Nb)}_Kab
   4. B -> A : {Kab1, Nb1}_Kab
 }


value a, b : principal
value na, nb, nb1, kab, kab1: number


s1. session (A=a; B=b; Na=na; Nb=nb; Nb1=nb1;
         Kab= kab; Kab1= kab1)

claim secret (kab1)
```

Figure A.2 - *Spécification des protocoles en eva.*

```
Echange_Gong

constant keypair PK,SK (principal): number
constant succ (number): number
parameter A, B : principal
parameter fresh N, fresh C : number
parameter fresh PW : weak


A knows A, B, PW, N, C, PK
B knows A, B, PW, PK, SK(B), succ

{
  1. A -> B : {C,N}_PK(B)
  2. B -> A : {succ(N)}_PW
}

value a, b : principal
value n, c, p: number



s1. session (A=a; B=b; N=n; N=n; PW = p; C= c)


claim secret (n)
```

```
Echange_Gong_modifie

constant keypair PK,SK (principal): number
constant succ (number): number
parameter A, B : principal
parameter fresh N, fresh C : number
parameter fresh PW : weak


A knows A, B, PW, N, C, PK
B knows A, B, PW, PK, SK(B), succ

{
  1. A -> B : {N}_PK(B)
  2. B -> A : {succ(N)}_PW
}

value a, b : principal
value n, c, p: number



s1. session (A=a; B=b; N=n; N=n; PW = p; C= c)


claim secret (n)
```

FIGURE A.3 - *Spécification en eva de l'échange de Gong avec ou sans le nonce C.*

```
Bellovin_Merritt

parameter keypair K1,K2: number
parameter A, B : principal
parameter fresh Na, fresh Nb, fresh K : number
parameter fresh PW : weak


A knows A, B, PW, Na, K1, K2
B knows A, B, PW, Nb, K

{
  1. A -> B : A, {K1()}_PW
  2. B -> A : {{K}_K1()}_PW
  3. A -> B : {Na}_K
  4. B -> A : {Na, Nb}_K
  5. A -> B : {Nb}_K
}

value a, b : principal
value na, nb, k, p, k1,k2: number



s1. session (A=a; B=b; Na=na; Nb=nb; K=k;
          PW = p; K1= k1; K2=k2)


claim secret (k)
```

```
Bellovin_Merritt_modifie


parameter A, B : principal
parameter fresh Na, fresh Nb, fresh K, fresh Kab : number
parameter fresh PW : weak


A knows A, B, PW, Na, Kab
B knows A, B, PW, Nb, K, Kab

{
  1. A -> B : A, {Kab}_PW
  2. B -> A : {{K}_Kab}_PW
  3. A -> B : {Na}_K
  4. B -> A : {Na, Nb}_K
  5. A -> B : {Nb}_K
}

value a, b : principal
value na, nb, k, p, kab: number


s1. session (A=a; B=b; Na=na; Nb=nb; K=k; PW = p; Kab= kab)

claim secret (k)
```

FIGURE A.4 - *Spécification en eva du protocole de Bellovin et Merritt (clef symétrique et asymétrique).*

Bibliographie

[1] D. M. Allester. Automatic recognition of tractability in inference relations. *Journal of the Association for Computing Machinery (ACM)*, 40(2) :284–303, April 1993.

[2] R. Amadio and W. Charatonik. On name generation and set-based analysis in the dolev-yao model. In *Proceedings of the 13th International Conference on Concurrency Theory (CONCUR'02)*, Lecture Notes in Computer Science, pages 499–514, Brno, Czech Republic, 2002. Springer Verlag.

[3] M. Bouallagui, Y. Chevalier, M. Rusinowitch, M. Turuani, and L. Vigneron. Analyse automatique de protocoles de sécurité avec CASRUL. In *Actes de SAR'2002, Sécurité et Architecture Réseaux*, 2002.

[4] L. Bozga, Y. Lakhnech, and M. Périn. Abstract interpretation for secrecy using patterns. Technical report, EVA : `http://www-eva.imag.fr/`, 2002.

[5] L. Bozga, Y. Lakhnech, and M. Périn. L'outil de vérification hermès. Technical report, EVA : `http://www-eva.imag.fr/`, 2002.

[6] L. Bozga, Y. Lakhnech, and M. Périn. Abstract interpretation for secrecy using patterns. LNCS, 2003. To appear.

[7] Y. Chevalier, R. Kuester, M. Rusinowitch, and M. Turuani. An NP decision procedure for protocol insecurity with xor. 2003. To appear.

[8] Y. Chevalier and L. Vigneron. Automated Unbounded Verification of Security Protocols. In E. Brinksma and K. Guldstrand Larsen, editors, *14th International Conference on Computer Aided Verification, CAV'2002*, volume 2404 of *Lecture Notes in Computer Science*, pages 324–337, Copenhagen (Denmark), July 2002. Springer.

[9] J. Clark and J. Jacob. A survey of authentication protocol literature. `http ://www.cs.york.ac.uk/~jac/ papers/drareviewps.ps`, 1997.

[10] H. Comon-Lundh and V. Cortier. Tree automata with one memory, set constraints and cryptographic protocols. In *Theoretical Computer Science, to appear*, 2003.

[11] H. Comon-Lundh and V. Shmatikov. Constraint solving, exclusive or and the decision of confidentiality for security protocols assuming a bounded number of sessions. June 2003. To appear.

[12] H. Comon-Lundh and R. Treinen. Easy intruder deductions. 2003. To appear.

[13] V. Cortier. Outil de vérification SECURIFY. Rapport numéro 7 du projet RNTL EVA, May 2002. 6 pages.

[14] V. Cortier. *Verification automatique des protocoles cryptographiques*. PhD thesis, Ecole Normale Superieure de Cachan, 2003.

[15] V. Cortier, J. Millen, and H. Rueß. Proving secrecy is easy enough. In *Proceedings of the 14th Computer Security Foundations Workshop (CSFW'01)*, pages 97–108, Cape Breton, Nova Scotia, Canada, 2001. IEEE Computer Society Press.

[16] D. Dolev and A. Yao. On the security of public key protocols. In *Proceedings of the 22nd Symp. on Foundations of ComputerScience*, pages 350–357, Nashville, Tennessee, USA, 1981. IEEE Computer Society Press.

[17] N. Durgin, P. Lincoln, J. Mitchell, and A. Scedrov. Undecidability of bounded security protocols. In *Proceedings of the Workshop on Formal Methods and Security Protocols*, Trento, Italia, 1999.

[18] L. Gong, T. M. A. Lomas, R. M. Needham, and J. H. Saltzer. Protecting poorly chosen secrets from guessing attacks. In *IEEE Journal on Selected Areas in Communications, Vol.11, No.5, June, 1993, pp.648-656*, 1993.

[19] J. Goubault-Larrecq. Les syntaxes et la sémantique du langage de spécification EVA. In *Rapport numéro 3 du projet RNTL EVA*, November 2001.

[20] G. Lowe. Casper : A compiler for the analysis of security protocols. In *Proceedings of 10th Computer Security Foundations Workshop (CSFW'97)*, Rockport, Massachusetts, USA, 1997. IEEE Computer Society Press. Also in Journal of Computer Security, Volume 6, pages 53-84, 1998.

[21] G. Lowe. A family of attacks upon authentication protocols. In *Proceedings of the 10th Computer Seurity Foundation Workshop (CSFW'97)*, Rockport, Massachusetts, USA, 1997. IEEE Computer Society Press.

[22] G. Lowe. Towards a completeness result for model checking of security protocols. In *Proceedings of the 11th Computer Security Foundations Workshop (CSFW'98)*, Rockport, Massachusetts, USA, 1998. IEEE Computer Society Press.

[23] G. Lowe. Analysing Protocols Subject to Guessing Attacks. In *Proceedings of the Workshop on Issues in the Theory of Security (WITS '02)*, 2002.

[24] J. Millen. Common authentication protocol specification language. http ://www.csl.sri.com/ millen/caspl.

[25] J. Millen and H. Rueß. Protocol-independent secrecy. In *Proceedings of the 21st Symposium on Research in Security and Privacy (RSP'00)*. IEEE Computer Society Press, 2000.

[26] R. Needham and M. Schroeder. Using encryption for authentication in large networks of computers. *Communication of the ACM*, 21(12) :993–999, 1978.

[27] M. Rusinowitch and M. Turuani. Protocol insecurity with finite number of sessions is NP-complete. In *Proceedings of the 14th Computer Security Foundations Workshop (CSFW'01)*, pages 174–190, Cape Breton, Nova Scotia, Canada, 2001. IEEE Computer Society Press.