Vérification Automatique Appliquée à un Protocole de Commerce Électronique*

Stéphanie Delaune[1,2], Francis Klay[1]

[1] France Télécom R&D
2, avenue Pierre Marzin – 22307 LANNION Cedex, FRANCE
francis.klay@rd.francetelecom.com
[2] Laboratoire Spécification & Vérification, ENS de Cachan & CNRS UMR 8643
61, avenue du Président Wilson – 94235 CACHAN Cedex, FRANCE
delaune@lsv.ens-cachan.fr

Résumé : Le domaine de la modélisation et de la vérification est une activité délicate et importante qui a connu une véritable explosion dans les années 1990. On dispose à l'entrée des années 2000 de toute une gamme de modèles et de méthodes plus ou moins avancés en ce qui concerne l'expressivité et l'automatisation.

Afin de définir les besoins et les priorités à mettre sur les outils consacrés à la vérification de protocoles cryptographiques qui seront développés au sein du projet RNTL PROUVÉ, nous proposons de travailler en situation réelle, sur des protocoles plutôt « durs », en effectuant le cycle suivant : modélisation, formalisation puis validation dans des outils existants. Ce travail est effectué ici pour un protocole de porte-monnaie électronique, développé récemment par une équipe de France Télécom.

Mots-clés : protocole cryptographique, porte-monnaie électronique, vérification automatique.

1 Introduction

La communication par des canaux publics comme 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...

Parmi les différents aspects de la sécurité informatique, la cryptologie, c'est à dire l'étude des moyens permettant de transmettre des messages secrets, et l'étude des protocoles cryptographiques ont pris une importance considérable avec le développement du commerce électronique. Ce dernier terme recouvre pour commencer les échanges chiffrés d'information qui ont lieu lorsque l'on retire de l'argent dans un distributeur de billets à l'aide d'une carte bancaire, où le distributeur de billets s'engage dans un dialogue avec la carte du client et la banque pour vérifier que l'utilisateur de la carte est honnête, dispose de la somme demandée sur son compte et ne pourra pas récupérer les billets demandés sans que son compte en soit débité. Le commerce électronique recouvre également les paiements par carte à l'aide de terminaux portables où l'utilisateur doit confirmer son identité en entrant son code secret à quatre chiffres, et aussi la monnaie électronique qui a pour but d'émuler électroniquement la monnaie courante. Cette monnaie est, pour des montants peu élevés, plus intéressante qu'une transaction bancaire par carte qui a un coût élevé pour le commerçant.

Toutes ces applications demandent des garanties de sécurité élevées, portant sur des propriétés de secret et d'authenticité, mais aussi de nombreuses autres propriétés, parmi lesquelles, la non-duplication (des factures, dans l'intérêt du client), la non-révocation (des commandes, dans l'intérêt du commerçant)... 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.

*Ce travail est partiellement financé par le projet RNTL PROUVÉ 03V360 et l'ACI-SI Rossignol.

Depuis les années 80, on dispose d'algorithmes cryptographiques suffisamment sûrs, mais 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 », « dictionary attack », ...) même en présence d'un chiffrement parfait. (Voir l'article de synthèse (Clark & Jacob, 1997) : presque tous les protocoles d'authentification comportent des failles). Les failles qui permettent de telles attaques sont qualifiées de failles logiques et on s'accorde pour penser que l'étude des failles logiques des protocoles est orthogonale à l'étude des failles du système de cryptographie sous-jacent. Ces failles sont relativement subtiles, difficiles à déceler à la simple vue du texte du protocole.

Ces protocoles interviennent dans des communications électroniques en assurant des propriétés de sécurité, ils ont 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 graves, en particulier à cause de leur déploiement à grande échelle. De plus, d'autres aspects tels que le respect de la vie privée entre en ligne de compte, il est donc crucial de pouvoir vérifier formellement les propriétés de ces protocoles. Cependant ces vérifications sont dures et laborieuses, il en résulte donc des coûts non négligeables, une approche automatique de ce travail est importante.

Nous présenterons la terminologie et les notations usuelles des protocoles cryptographiques (section 2), ainsi que les principales hypothèses faites, souvent de manière implicite lors de la définition formelle du modèle (section 3). Ensuite, nous nous intéresserons à deux protocoles de porte-monnaie électronique (sections 4 et 6), développés récemment par une équipe de France Télécom. Ces études de cas, réalisées dans le cadre du projet RNTL PROUVÉ, ont pour but de définir les besoins des logiciels de vérification « à venir ». Nous proposons une modélisation de ces protocoles : c'est une tâche délicate car la majeure partie des spécifications sont en langue naturelle, donc incomplète et ambiguë. Après les avoir formalisés, nous procédons à leur vérification (section 5) en utilisant deux outils aux caractéristiques très différentes : CASRUL et PROVERIF.

2 Terminologie

Dans ce paragraphe, nous présentons la terminologie et les notations usuelles des protocoles cryptographiques, à travers un exemple très classique : le protocole de Needham-Schroeder (Needham & Schroeder, 1978). De très nombreux autres exemples de protocoles sont décrits dans (Clark & Jacob, 1997).

2.1 Protocole

Ce protocole décrit des règles d'échange de messages entre deux *participants*, aussi appelés *agents*. Celui-ci a pour but de permettre une authentification mutuelle entre les agents A et B par l'intermédiaire des nonces N_a et N_b qui doivent rester secrets. Il se décrit de la façon suivante :

$$\begin{aligned} A &\rightarrow B : \quad \{\langle A, N_a\rangle\}_{\mathsf{pub}(B)} \\ B &\rightarrow A : \quad \{\langle N_a, N_b\rangle\}_{\mathsf{pub}(A)} \\ A &\rightarrow B : \quad \{N_b\}_{\mathsf{pub}(B)} \end{aligned}$$

À la première étape du protocole, l'agent Alice envoie son nom A et un nombre engendré aléatoirement N_a, aussi appelé *nonce*. Ce message est chiffré par un algorithme de chiffrement asymétrique avec la *clef publique* de B (notée $\mathsf{pub}(B)$), c'est-à-dire que seul l'agent Bob connaît la clef privée correspondant à la clef $\mathsf{pub}(B)$. La notation usuelle pour le chiffrement est $\{\_\}_\_$: le message m chiffré à l'aide d'une clef k est noté $\{m\}_k$. Ainsi, le message envoyé par Alice est noté : $\{\langle A, N_a\rangle\}_{\mathsf{pub}(B)}$.

À la deuxième étape du protocole, Bob reçoit le message $\{\langle A, N_a\rangle\}_{\mathsf{pub}(B)}$ envoyé par Alice. Comme il a la clef privée (souvent notée $\mathsf{prv}(B)$) lui permettant d'ouvrir le message, il comprend qu'Alice veut lui parler et renvoie le nonce d'Alice ainsi qu'un autre nonce N_b qu'il vient d'engendrer, le tout chiffré avec la clef publique $\mathsf{pub}(A)$ d'Alice.

À la troisième étape du protocole, Alice reçoit le message $\{\langle N_a, N_b\rangle\}_{\mathsf{pub}(A)}$ et reconnaît son nonce N_a. Elle en déduit que Bob lui a répondu et elle lui renvoie son nonce N_b chiffré avec sa clef publique pour lui signifier qu'elle connaît maintenant le message N_b. Lorsque Bob reçoit ce message, les deux agents pensent

qu'ils sont seuls à connaître le nonce N_b et que celui-ci permet de les *authentifier* : lorsqu'Alice reçoit un message contenant N_b, elle en déduit qu'il vient de Bob et inversement.

Ce protocole peut être joué plusieurs fois avec différents participants et différentes valeurs pour les nonces. Un déroulement du protocole est aussi appelé *session*. La difficulté de la conception d'un protocole tient au fait que les messages échangés peuvent être écoutés, interceptés ou modifiés par une tierce personne, appelée l'*intrus*.

2.2 Attaque

Les premiers intrus considérés étaient *passifs* : ils se contentaient d'écouter et d'analyser les messages circulant sur le réseau en déchiffrant les messages lorsqu'ils avaient la clef correspondante. Les intrus considérés ici sont *actifs*, ils peuvent intercepter les messages circulant sur le réseau, les analyser, en synthétiser de nouveaux et les envoyer.

Ainsi G. Lowe (Lowe, 1996) a découvert une quinzaine d'années après la publication du protocole de Needham-Schroeder que ce dernier avait une faille en présence d'un intrus actif. Cette faille est souvent appelée « man-in-the-middle attack ».

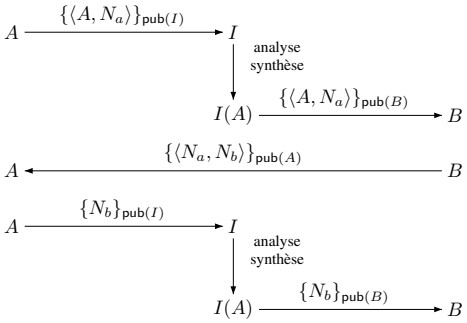


FIG. 1 – Attaque du Protocole de Needham-Schroeder, due à G. Lowe.

Elle est schématisée à la figure 1. L'agent A commence spontanément une conversation avec un agent I, malhonnête. L'agent I se sert de ce premier message pour se faire passer pour A auprès de B. Celui-ci répond donc à A. L'agent A, reconnaissant son nonce N_a pense que I vient de lui répondre. L'agent A, renvoie donc à I le nonce N_b que l'agent I n'aurait pas dû connaître. L'agent I termine alors le protocole avec B qui croit avoir parlé à A.

3 Modélisation et Vérification

Le problème de la vérification des protocoles cryptographiques n'est pas nouveau : il remonte aux années 1980. Cependant, les protocoles utilisés sont de plus en plus compliqués tant du point de vue des propriétés à garantir (propriété d'équité pour les protocoles de signature de contrat) que du point de vue des mécanismes de base tels que le chiffrement et la signature, utilisés pour réaliser ces protocoles (chiffrement homomorphique, à seuil, signature à l'aveugle). L'importance de vérifier ces protocoles difficiles n'a donc fait que croître et ce n'est que depuis 1998 environ que des cadres formels (sémantique) ont été proposés pour effectuer ces vérifications.

3.1 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. Le nombre de sessions d'un protocole (en parallèle ou en séquence) n'est pas borné, les capacités mémoire d'un attaquant ainsi que la taille des messages que peut fabriquer un attaquant ne sont pas a priori bornées. De plus, 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 (Amadio & Charatonik, 2002; Durgin *et al.*, 1999; Comon & Cortier, 2004).

Compte tenu de la difficulté des problèmes de vérification, certaines hypothèses simplificatrices ont été prises. La modélisation des protocoles cryptographiques fait abstraction de certaines particularités des protocoles :

- La modélisation la plus courante du chiffrement repose sur 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 certains cas et ne correspond pas toujours à la réalité, les algorithmes de chiffrement pouvant posséder des propriétés algébriques.
- La concaténation, couramment utilisée dans les protocoles cryptographiques, est en général modélisée par un symbole binaire $\langle \_, \_ \rangle$. Comme les messages sont des termes, le message $\langle m_1, \langle m_2, m_3 \rangle \rangle$ est distinct du message $\langle \langle m_1, m_2 \rangle, m_3 \rangle$. Pour modéliser l'associativité de la concaténation, il faut ajouter la théorie équationelle de l'associativité du symbole $\langle \_, \_ \rangle$. Certains outils de vérification, comme l'outil CASRUL, offre cette possibilité.
- Les nonces, nombres aléatoires engendrés au cours du protocole, sont modélisés par des données toutes distinctes lorsque l'on travaille dans le cadre d'un nombre borné de sessions. Mais dès que l'on considère un nombre non borné de sessions la modélisation des nonces est plus délicate.
- Les actions de l'intrus, dans les modèles, dits *à la Dolev-Yao*, sont décrites par un ensemble de règles où apparaissent la capacité de l'intrus à déchiffrer des messages lorsqu'il connaît la clef de déchiffrement ou sa capacité à former de nouveaux messages. Notons que dans ces modèles, la capacité de calcul de l'intrus n'est pas bornée et que la capacité de l'intrus à comparer des messages n'est pas prise en compte.

La vérification des protocoles cryptographiques s'articule en trois axes. D'une part le test qui consiste à essayer des scénarios dans le but de trouver une attaque. Mais celui-ci est peu intéressant car il n'offre aucune garantie sérieuse. On s'intéresse donc à la recherche d'attaques pour un nombre borné de sessions permettant de garantir la propriété souhaitée pour un nombre de sessions borné (à la fois en parallèle et en séquence). Mais le seul résultat permettant de borner (à 2) le nombre de sessions nécessaires à une attaque (Lowe, 1998) concerne une classe de protocoles extrêmement restreinte. C'est pourquoi une troisième approche, consistant à faire de la preuve, est développée. Les démonstrateurs automatiques procèdent en général par abstraction (Blanchet, 2001; Bozga *et al.*, 2003).

3.2 Recherche d'Attaques

Les premiers résultats en vérification de protocoles cryptographiques ont été des découvertes d'attaques comme l'attaque de G. Lowe (Lowe, 1996) sur le protocole de Needham-Schroeder à clefs publiques. Différents logiciels sont consacrés à la recherche d'attaques sur les protocoles, c'est le cas des outils CASPER/FDR, MURφ, CASRUL,...

Les outils recherchant des attaques travaillent sur un nombre borné de sessions, et l'absence d'attaques sur un protocole cryptographique, ne permet pas de garantir la correction du protocole. C'est pourquoi il faut également développer des méthodes de preuves des protocoles cryptographiques.

3.3 Preuve par Abstraction

Une preuve de sécurité nécessite de développer un cadre convaincant pour modéliser les règles des protocoles et, plus encore, pour exprimer les propriétés de sécurité recherchées. En effet, la propriété sera prouvée dans le modèle considéré donc il faut pouvoir en justifier les limites.

De nombreuses abstractions sont possibles : par exemple, dans le cadre d'un nombre non borné de sessions, une possibilité pour représenter les nonces, consiste à les modéliser par une fonction des messages reçus par l'agent créant le nonce. Dans ce cas, on ne peut éviter l'inconvénient suivant : deux nonces engendrés à deux sessions différentes peuvent être égaux. Cependant, une telle abstraction est correcte : si on trouve que le protocole est correct dans le modèle abstrait alors il est correct dans le modèle concret. Par contre, cette abstraction n'est pas complète et peut conduire à la découverte de «fausses» attaques, basées sur le renvoi d'un nonce déjà engendré, ce qui n'est pas possible dans la réalité. Elle permet cependant de prouver le secret ou l'authentification de nombreux protocoles et a été implémentée dans l'outil PROVERIF.

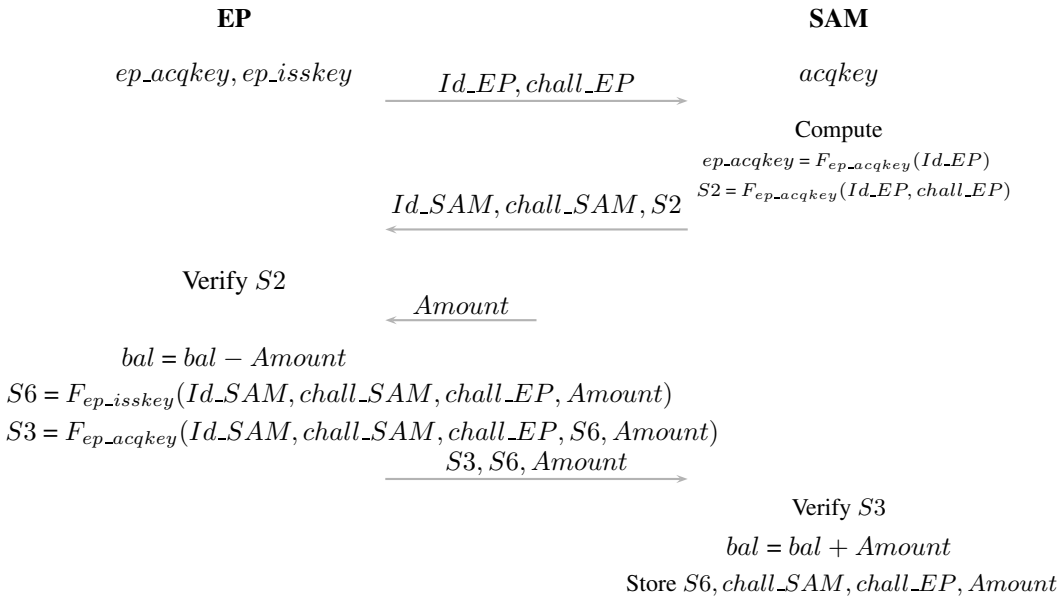


FIG. 2 – Description du Protocole de Porte-Monnaie Électronique – Approche symétrique

4 Protocole de Porte-Monnaie Électronique – Approche Symétrique

Ce protocole permet la réalisation d'une transaction entre un porte-monnaie électronique et un serveur : le but est de garantir un bon niveau de sécurité, et ce avec un bon niveau de performance grâce à l'utilisation du chiffrement symétrique (moins coûteux que les mécanismes de chiffrement asymétrique).

4.1 Description du Protocole

Le protocole donné dans (Girault & Paillès, 2001) est décrit Figure 2. Au cours d'un premier échange, le porte-monnaie électronique EP envoie au serveur SAM (Secure Application Module) son identité Id_EP et un challenge $chall\_EP$ (msg 1). SAM calcule la valeur ep_acqkey, celle-ci sera utilisée pour «chiffrer» les prochains messages. SAM répond au challenge de EP et donne son identité accompagnée d'un nouveau challenge (msg 2). EP vérifie la réponse fournie par SAM et reçoit le montant de la transaction (msg 3). À ce moment, EP débite sa balance et répond à SAM (msg 4) en construisant un message contenant l'identité de SAM, les valeurs des deux challenges, le montant de la transaction ainsi qu'un message noté $S6$. SAM vérifie $S3$ en reconstruisant le message et crédite sa balance si la vérification se passe bien. De plus, il stocke le message $S6$ pour résoudre d'éventuels litiges ultérieurs (F_{ep_isskey} est une donnée secrète entre le porte-monnaie et un tiers de confiance).

Une des propriétés que l'on souhaite vérifier sur ce protocole est la non création de fausse monnaie.

4.2 Modélisation du Protocole

En général, la majeure partie des spécifications sont en langue naturelle, et la première partie du travail consiste à lever les ambiguïtés. La modélisation apparaît alors comme un passage obligé extrêmement délicat, puisque c'est sur elle que repose la confiance que l'on peut avoir dans le résultat de la vérification formelle. La modélisation de ce texte est une tâche délicate et laborieuse qui ne peut se faire qu'en partenariat avec les concepteurs du protocole pour ne pas biaiser leur vision initiale du protocole. En effet, une vérification sans adhésion à ce niveau perd tout son intérêt. Un certain nombre de points, parmi lesquels ceux détaillés ci-dessous, ont ainsi pu être éclaircis :

– Les fonctions F_{ep_acqkey} et F_{ep_isskey} ne doivent pas être vues comme des fonctions de chiffrement symétrique : elles ne sont pas inversibles. Autrement dit, il est impossible de retrouver les composantes du message $F_{ep\_isskey}(Id\_SAM, chall\_SAM, chall\_EP, Amount)$, même si on a connaissance de la fonction F_{ep_isskey}. Ces deux fonctions, F_{ep_acqkey} et F_{ep_isskey}, doivent être vues comme des fonctions de hachage, et elles seront modélisées en tant que telles par la suite. Pour vérifier les messages tels que $S3$ et $S6$, la seule façon est de les reconstruire et d'effectuer une comparaison.

```
EP       →  SAM     :  EP, Nep

SAM      →  I(EP)   :  SAM, Nsam, {EP, Nep}_fepacqkey(EP)
I(SAM)   →  EP      :  SAM, <Nsam, Nep, S6'>, {EP, Nep}_fepacqkey(EP)

bal(EP) = bal(EP) - M

EP       →  I(SAM)  :  {SAM, <Nsam,Nep,S6'> ,Nep, S6, M}_fepacqkey(EP), S6, M
I(EP)    →  SAM     :  {SAM, Nsam, Nep, S6', <Nep, S6, M>}_fepacqkey(EP),
                       S6',
                       <Nep, S6, M>

bal(SAM) = bal(SAM) + <Nep, S6, M>
```

avec `S6 = {SAM, <Nsam,Nep,S6'>, Nep, M}_fepisskey(EP)` et `S6'` un message arbitraire construit par l'intrus.

FIG. 3 – Attaque sur le protocole du Porte-Monnaie Électronique

- Le canal sur lequel circule le montant de la transaction est un canal sécurisé, l'intrus peut écouter sur ce canal, mais en aucun cas intercepter et/ou modifier les messages qui y circulent. Les outils existant à l'heure actuelle ne permettant pas de faire cette distinction entre les canaux, on choisit donc de modéliser le montant par un nonce généré par le porte-monnaie électronique.
- Ce protocole a été conçu pour réaliser du paiement local. L'objectif n'était donc pas d'obtenir un protocole Internet. Les sessions ont donc lieu les unes après les autres sans s'entrelacer (ni côté EP, ni côté SAM). Cependant les outils existants ne permettent pas de prendre en compte cette information.

Tous les points cités plus haut sont implicites dans la description en langue naturelle et sont pourtant indispensables lorsque l'on souhaite formaliser le protocole et vérifier un certain nombre de propriétés.

La propriété de non création de fausse monnaie peut se «traduire» en une propriété d'authentification forte, encore appelée « injective agreement » (Lowe, 1997). Nous allons vérifier qu'à chaque fois que le serveur SAM termine une session avec EP en créditant sa balance d'un certain montant, le porte-monnaie EP a joué une session avec SAM et a débité sa balance du même montant.

5 Résultats Obtenus

Nous avons choisi deux outils différents pour vérifier ce protocole : CASRUL, un outil de recherche d'attaque et PROVERIF un outil de recherche de preuves. Ces deux outils ont été retenus car contrairement à de nombreux outils qui se limitent à vérifier des propriétés de secret, ils permettent d'exprimer des propriétés d'authentification et en particulier des propriétés d'authentification forte (correspondance un pour un) avec accord sur certaines valeurs (ici le montant).

5.1 Outil CASRUL

L'outil CASRUL (Jacquemard *et al.*, 2000) permet de chercher des attaques pour un nombre de sessions et de participants borné. C'est l'un des seuls outils qui permet de traiter les messages dans toute leur généralité : les clefs composées sont supportées et les messages n'ont pas besoin d'être typés. Les protocoles sont décrits sous forme de règles de réécriture. CASRUL retrouve ainsi de nombreuses attaques mentionnées dans le (Clark & Jacob, 1997).

La spécification du protocole de porte-monnaie électronique, présentée à la section précédente, telle qu'elle peut être donnée en entrée à l'outil CASRUL est présentée à l'annexe A. Nous avons obtenu la compilation de ce protocole dans le modèle par rôle ainsi qu'une attaque par confusion de type utilisant l'associativité de la paire, entièrement décrite Figure 3. Côté porte-monnaie, la transaction a lieu avec un montant `M`, montant qui est débité sur la carte, alors que côté serveur, le montant `<Nep, S6, M>` est crédité.

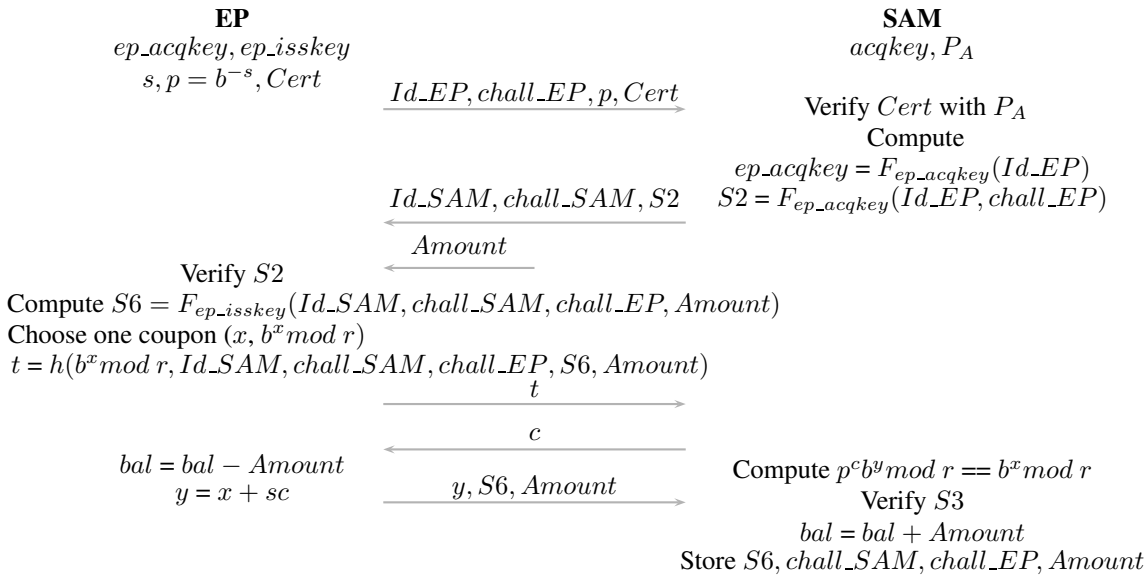


FIG. 4 – Description du Protocole de Porte-Monnaie Électronique – Approche Asymétrique

On peut bien sûr se poser la question de la pertinence d'une telle attaque. Est-il possible pour un serveur de confondre un montant avec un tel triplet ? Si aucune vérification n'est faite, ces deux messages sont des suites de bits et peuvent être confondus. Un simple test pourrait permettre d'éviter cette attaque, mais peut-être en en créant d'autres... Ce qui montre une fois de plus l'importance d'une vérification formelle et complète.

L'outil CASRUL permettant de choisir si l'on souhaite tenir compte de l'associativité de la paire dans la recherche d'attaques, nous avons effectué la vérification du protocole en supposant que le symbole de formation des paires n'était pas associatif et l'outil n'a pas trouvé d'attaques, même en augmentant le nombre de sessions en séquences. Mais l'absence d'attaques pour un nombre de sessions borné, aussi grand soit-il, ne permet pas de garantir la sécurité du protocole testé.

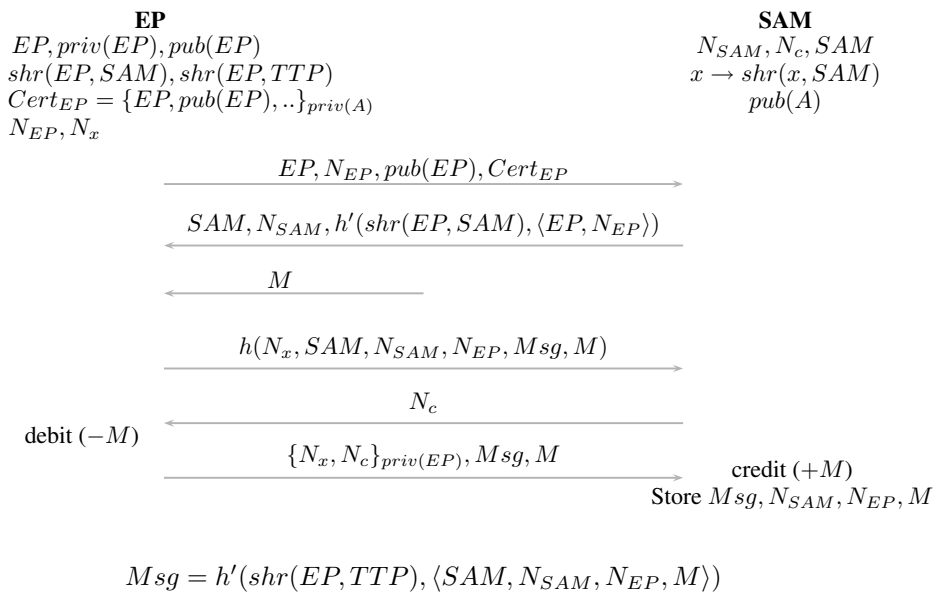
5.2 Outil PROVERIF

PROVERIF est un outil consacré à la preuve (Blanchet, 2001), et les protocoles y sont exprimés sous forme de règles Prolog. Il vérifie le secret et aussi des propriétés d'authentification pour un nombre non borné de sessions. La taille des messages n'est pas bornée non plus. Les nonces sont abstraits par une fonction des paramètres reçus dans les messages précédents ce qui peut conduire à de «fausses» attaques. Il se peut, en effet, qu'un protocole ne soit pas prouvé correct car sujet à des attaques qui utilisent l'abstraction faite sur les nonces alors qu'une telle attaque n'est pas reproductible dans la réalité. Cet outil type, au moins en partie les messages et ne permet pas d'exprimer les clefs composées.

La spécification de ce protocole telle qu'elle peut être donnée en entrée à l'outil PROVERIF est présentée à l'annexe B. PROVERIF conclut en prouvant la propriété souhaitée sur le protocole étudié, ce qui est consistant avec le verdict de CASRUL.

6 Protocole de Porte-Monnaie Électronique – Approche Asymétrique

Ce protocole permet la réalisation d'une transaction entre un porte-monnaie électronique et un serveur : le but est de garantir un bon niveau de sécurité et de gagner en ouverture par l'utilisation de méthodes de chiffrement asymétrique, et ce à faible coût. Ces besoins spécifiques ont conduit à développer une solution originale décrite dans (Girault & Paillès, 2001).



$$Msg = h'(shr(EP, TTP), \langle SAM, N_{SAM}, N_{EP}, M\rangle)$$

FIG. 5 – Formalisation du Protocole de Porte-Monnaie Électronique – Approche Asymétrique

6.1 Description du Protocole

Le protocole est décrit Figure 4. Au cours d'un premier échange, le porte-monnaie électronique envoie au serveur SAM son identité EP, un challenge $chall\_EP$, sa clef publique et $Cert$ pour certifier sa clef publique (msg 1). SAM vérifie le certificat et calcule la « clef symétrique » ep_acqkey, celle-ci sera utilisée pour chiffrer les prochains messages. SAM répond au challenge de EP et donne son identité accompagnée d'un nouveau challenge (msg 2). EP vérifie la réponse fournie par SAM et reçoit le montant de la transaction (msg 3). Ensuite, EP construit un message contenant $b^x \ mod \ r$, l'identité de SAM, son identité, les valeurs des deux challenges ainsi que le montant de la transaction, il le hache et l'envoie à SAM (msg 4). SAM stocke le message et génère un nombre aléatoire c qu'il envoie à EP. À ce moment, EP débite sa balance et répond à SAM (msg 6) pour lui permettre de vérifier (en le reconstruisant) le hash qu'il a reçu à l'étape 4, ceci est possible car $b^x mod \ r$ n'est rien d'autre que $p^c b^y mod \ r$ compte tenu des propriétés algébriques de l'exponentielle. Si la vérification se passe bien, il crédite sa balance. De plus, il stocke le message $S6$ pour résoudre d'éventuels litiges ultérieurs (F_{ep_isskey} est une donnée secrète entre le porte-monnaie et un tiers de confiance).

6.2 Modélisation du Protocole

La modélisation du protocole proposée Figure 5 a pour but de formaliser au mieux le protocole en contournant les difficultés dues à l'utilisation de l'exponentielle modulaire et de ses propriétés puisqu'aucun outil ne permet à l'heure actuelle de prendre en compte cet opérateur.

Bien que la modélisation faite des 3 derniers messages soit assez différente de la spécification initiale du protocole, elle reflète au mieux l'esprit du protocole dans le sens où le dernier message $\{N_x, N_c\}_{priv(EP)}$ dépend bien du nonce N_c engendré à l'étape précédente, comme y dépend de c dans le protocole. De même, le secret $priv(EP)$ permet de vérifier le hash reçu à l'étape 4, tout comme le secret s le permettait.

Remarquons tout de même que dans la modélisation proposée les rôles de x et $b^x \ mod \ r$ sont fusionnés dans le rôle de N_x. À la fin de la session, le nonce N_x est connu de l'intrus contrairement à x, et les propriétés algébriques de l'exponentielle utilisées lors de la vérification sont remplacées par un chiffrement asymétrique dans la modélisation. Cette solution n'a pas été retenue par les concepteurs du protocole pour des raisons évidentes de coûts.

6.3 Résultats Obtenus

Avec CASRUL, on obtient une attaque due à l'associativité de la paire, similaire à celle trouvée sur la première version du protocole. L'outil PROVERIF n'arrive pas à prouver la propriété souhaitée et fournit

en sortie une trace. Mais cette dernière correspond en fait à la description d'une « fausse attaque » due a l'abstraction faite sur les nonces. C'est une des limitations des outils travaillant par abstraction.

7 Conclusion

Sur la première version du protocole de porte-monnaie électronique, nous avons proposé une modélisation convaincante puisque que relativement proche de la description fournie par les concepteurs du protocole. Ceci a été possible car les primitives utilisées sont des primitives du modèle classique : chiffrement, fonctions de hachage,.... Cette étude de cas, relativement simple, a ensuite pu être traitée par des outils dédiés à la vérification de protocoles cryptographiques.

Du côté des propriétés à vérifier, nous nous sommes intéressés à une propriété relativement simple, mais d'autres propriétés moins étudiées telles que la non-duplication, la non-répudiation sont toutes aussi importantes pour ce genre de protocole. Mais aucun outil à l'heure actuelle (à notre connaissance) ne permet de traiter ce type de propriétés.

La deuxième version du protocole de porte-monnaie électronique fait intervenir des propriétés algébriques de l'exponentiation modulaire. Cet opérateur n'étant accepté par aucun outil, seul un effort au niveau de la modélisation du protocole peut nous permettre d'obtenir des résultats sur ce protocole. Mais l'étape de modélisation est une tâche délicate qui ne peut se justifier que de manière informelle. Il est donc nécessaire de disposer de modèles suffisamment expressifs pour être convaincant et pouvoir par la suite avoir confiance dans le résultat fourni par la vérification formelle.

Les résultats obtenus sur ces études de cas constituent pour le projet RNTL PROUVÉ un point de départ pour définir les besoins des logiciels « à venir ». Les problèmes spécifiques rencontrés lors de ce travail ont permis d'alimenter les autres tâches du projet comme la mise au point du langage : ajout de variables mutables, équations permettant d'exprimer les propriétés algébriques des opérateurs, gestion des différents modes multi-sessions...

Remerciements

Les auteurs tiennent à remercier Marc Girault de France Télécom T&I/FTRD&/DTL/SSR/SSR-CAE pour l'aide et les précieux conseils qu'il a apportés dans le cadre de ce travail.

Références

AMADIO R. & CHARATONIK W. (2002). On Name Generation and Set-Based Analysis in the Dolev-Yao Model. In *Proceedings of the 13th International Conference on Concurrency Theory (CONCUR)*.

BLANCHET B. (2001). An Efficient Cryptographic Protocol Verifier Based on Prolog Rules. In *Proceedings of the 14th Computer Security Foundations Workshop (CSFW)*.

BOZGA L., LAKHNECH Y. & PÉRIN M. (2003). HERMÈS : An Automatic Tool for Verification of Secrecy in Security Protocols. In *Proceedings of the 15th International Conference of Computer Aided Verification (CAV)*.

CLARK J. & JACOB J. (1997). A Survey of Authentication Protocol Literature.

COMON H. & CORTIER V. (2004). Tree Automata with One Memory, Set Constraints and Cryptographic Protocols. *Theoretical Computer Science*. To appear.

DURGIN N., LINCOLN P., MITCHELL J. & SCEDROV A. (1999). Undecidability of Bounded Security Protocols. In *Proceedings of the Workshop on Formal Methods and Security Protocols (FMSP)*.

GIRAULT M. & PAILLÈS J.C. (2001). Contactless EP : A Public-Key Solution with Good Performances.

JACQUEMARD F., RUSINOWITCH M. & VIGNERON L. (2000). Compiling and Verifying Security Protocols. In *Proceedings of the 7th International Conference of Logic for Programming and Automated Reasoning (LPAR)*.

LOWE G. (1996). Breaking and Fixing the Needham-Schroeder Public-key Protocol Using FDR. In *Proceedings of the 2nd International Conference of Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*.

LOWE G. (1997). A Hierarchy of Authentication Specifications. In *Proceedings of the 10th Computer Security Foundations Workshop (CSFW)*.

LOWE G. (1998). Towards a Completeness Result for Model Checking of Security Protocols. In *Proceedings of the 11th Computer Security Foundations Workshop (CSFW)*.

NEEDHAM R. & SCHROEDER M. (1978). Using Encryption for Authentication in Large Networks of Computers. *Communication of the ACM*, **21**(12), 993–999.

A Codage en CASRUL

```
Protocol EP;
Identifiers
  EP,SAM : user;
  fepacqkey, fissqkey, hash: function;
  Nep, Nsam, M: number;
Knowledge
  EP : EP, SAM, hash, fepacqkey(EP), fissqkey(EP);
  SAM : SAM, hash, fepacqkey;
Messages
 1. EP -> SAM : EP, Nep
 2. SAM-> EP  : SAM, Nsam, hash(fepacqkey(EP), (EP, Nep))
 3. EP -> SAM : hash(fepacqkey(EP), (SAM,
                                     Nsam,
                                     Nep,
                                     hash(fissqkey(EP), (SAM, Nsam, Nep, M)),
                                     M)),
                  hash(fissqkey(EP), (SAM, Nsam, Nep, M)),
                  M

Session_instances
[EP:ep; SAM:sam; hash:h; fepacqkey: fepacq; fissqkey: fissq];
Intruder divert, impersonate, Eaves_dropping;
Intruder_knowledge ep, sam;
Goal SAM authenticate EP on M;
```

B Codage en PROVERIF

```
free c.
(**************** Configuration Parameters *****************)
param attacker = active.
param keyCompromise = none.
param injectiveAg = true.
param predicatesImplementable = check.
(******************** Functions ****************************)
private fun facqkey/1.
private fun fisskey/1.
fun hash/2.
authquery authentification/3.
(********************* Process EP **************************)
let processEP =
        in(c, IdX);
        new Nep; out(c, (IdX, Nep));
        in(c, m);
        let (IdY, NY, S2)= m in
        if S2 = hash(facqkey(IdX),(IdX, Nep)) then
        new M;
        begin authentification(IdX,IdY,M);
        let S6 = hash(fisskey(IdX), (IdY, NY, Nep, M)) in
        let S3 = hash(facqkey(IdX), (IdY, NY, Nep, S6, M)) in
        out(c, (S3,S6,M)).
(********************** Process SAM ************************)
let processSAM =
        in(c, IdY); in(c, m1);
        let (IdX, Nep) = m1 in
        new Nsam;
        let fepacqkey = facqkey(IdX) in
        let s2 =  hash(fepacqkey, (IdX, Nep)) in
        out(c, (IdY, Nsam, s2));
        in(c, m2);
        let (S3, S6, M) = m2 in
        if S3 = hash(fepacqkey, (IdY, Nsam, Nep, S6, M)) then
        end authentification(IdX, IdY, M).
(********************** Protocol *****************************)
process new Idep; out(c, Idep);
        new Idsam; out(c, Idsam);
        (!processSAM | !processEP)
```