Vérifier les protocoles cryptographiques

Véronique Cortier

Loria, INRIA & CNRS, équipe Cassis
615, rue du Jardin Botanique, BP 101,
54602 Villers les Nancy Cedex
cortier@loria.fr

RÉSUMÉ. *Les protocoles cryptographiques sont des règles d'échange entre les points d'un réseau, ils permettent de sécuriser les communications. Leur vérification est devenue cruciale car une petite faille peut avoir des répercussions économiques importantes. Nous proposons ici une synthèse des résultats de décidabilité et d'indécidabilité pour des propriétés de secret et d'authentification en considérant différentes restrictions : borner le nombre de sessions, la taille des messages, le nombre de copies à chaque transition, etc. D'autre part, nous décrivons les outils qui peuvent être utilisés pour vérifier les protocoles.*

ABSTRACT. *A cryptographic protocol is a description of message exchanges on a network. The verification of such programs has become crucial. We propose here a synthesis of decidability and undecidability results for secrecy and authentication properties. We consider several restrictions: bound on the number of sessions, on the size of messages, on the number of copies at each transition, etc. Moreover, we describe which tools may be used to verify the protocols.*

MOTS-CLÉS : *protocoles cryptographiques, sécurité, secret, authentification.*

KEYWORDS: *cryptographic protocols, security, secrecy, authentication.*

1. Introduction

Avec le développement des réseaux de communication comme Internet et les réseaux de téléphonie mobile, le besoin d'assurer la confidentialité et l'authenticité des messages échangés a considérablement augmenté. Les protocoles cryptographiques sont des règles d'échange entre les points d'un réseau, ils permettent de sécuriser les communications. Ils sont utilisés par exemple dans les distributeurs de billets, les abonnements aux chaînes de télévision payantes, la téléphonie mobile, le commerce électronique.

Ces protocoles sont basés sur la cryptographie : étant donnés un message et un nombre secret assez grand (appelé clef), des algorithmes de chiffrement permettent d'obtenir un message chiffré tel qu'il est impossible de retrouver le premier message à moins d'avoir aussi la clef. La description de ces protocoles étant en général très courte, on pourrait penser qu'il est aisé de s'assurer de la fiabilité d'un protocole. Il n'en est rien : de nouvelles attaques sont trouvées régulièrement et parfois sur des protocoles connus depuis plusieurs années.

La difficulté de la conception des protocoles tient au fait que les messages échangés peuvent être écoutés par une tierce personne, interceptés et modifiés. Ainsi les systèmes d'abonnements aux chaînes télévisées payantes ont déjà été beaucoup piratés, en interposant un ordinateur entre les messages reçus et le décodeur ou en modifiant certains composants. Ces attaques ont des répercussions économiques très importantes.

De plus, les protocoles utilisés sont très nombreux et chacun a presque toujours plusieurs variantes. Aussi, il est intéressant de développer des classes décidables de protocoles cryptographiques afin d'obtenir des méthodes automatiques de vérification pour les protocoles. L'objet de cet article est de faire un tour d'horizon des résultats de décidabilité existants.

Pour vérifier les protocoles cryptographiques, une hypothèse classique est *l'hypothèse du chiffrement parfait* que nous introduisons au paragraphe 2. Cette hypothèse suppose qu'on ne peut pas obtenir d'informations sur un message chiffré à moins de connaître la clef inverse. Elle permet d'automatiser la vérification des protocoles. Elle distingue, de plus, le travail de vérification de la robustesse des algorithmes de chiffrements utilisés d'une part, et la détection de failles logiques dans la conception des protocoles d'autre part. Nous présentons la terminologie des protocoles cryptographiques au travers d'exemples au paragraphe 3 et nous décrivons plusieurs propriétés de sécurité étudiées à l'heure actuelle. La modélisation de ces propriétés n'est pas encore terminée pour certaines d'entre elles, aussi nous nous limitons ensuite à l'étude du secret et abordons parfois le problème de l'authentification. Cependant, même pour ces propriétés, différentes sources d'indécidabilité existent, nous les expliquons au paragraphe 4. Puis nous présentons une liste que nous espérons représentative, des résultats de décidabilité (paragraphe 5). Enfin, nous décrivons, de manière non exhaustive, les outils possibles pour vérifier les protocoles cryptographiques (paragraphe 6).

2. Cryptographie et protocoles cryptographiques

Le chiffrement consiste à transformer un message en un autre de manière à ne plus reconnaître le premier. Le déchiffrement est l'opération inverse.

On distingue deux types de chiffrements : les chiffrements symétriques et les chiffrements asymétriques. Le chiffrement symétrique utilise la même clef pour chiffrer comme pour déchiffrer. Le chiffrement asymétrique utilise une clef de chiffrement différente de la clef de déchiffrement. La première est souvent divulguée sur le réseau afin que tout participant puisse chiffrer mais la deuxième reste en général secrète pour qu'une seule personne (ou machine) puisse déchiffrer. L'un des algorithmes de chiffrement les plus connus est l'algorithme RSA, dû à R.L. Rivest, A. Shamir et L.M. Adleman [RIV 78]. Pour le chiffrement RSA, le record actuel a consisté à « casser » (c'est-à-dire calculer la clef correspondant au déchiffrement) une clef de 512 bits en quatre mois. Personne n'a encore réussi à « casser » une clef de 1024 bits par exemple. Le but de cet article n'est pas d'étudier les différents algorithmes de chiffrements, le lecteur pourra se reporter pour cela au livre d'introduction à la cryptographie de B. Schneier [SCH 96b] ou à celui de A. J. Menezes *et al.* [MEN 97].

On pourrait penser que les principales attaques des protocoles cryptographiques reposent sur le déchiffrement des messages chiffrés. En fait, beaucoup d'attaques reposent sur des principes beaucoup plus simples à mettre en œuvre comme l'interception d'un message et le renvoi d'un autre. De plus, la vérification d'un protocole en tenant compte de toutes les propriétés algébriques de l'algorithme de chiffrement utilisé est difficilement automatisable. En effet, même les preuves concernant uniquement la robustesse des algorithmes de chiffrement (et non du protocole tout entier) sont des preuves mathématiques assez longues et complexes qui font intervenir des résultats d'algèbre et la théorie de la complexité. Il s'agit donc de problèmes qui ne sont *a priori* pas décidables.

Aussi, dans le cadre de la vérification des protocoles cryptographiques, une hypothèse très classique est celle *du chiffrement parfait* [DOL 81]. Une telle hypothèse assure en particulier qu'un intrus ne peut déchiffrer un message m chiffré avec une clef k que s'il possède l'inverse de cette clef. Mais les hypothèses induites par le modèle sont souvent encore plus fortes que cela. Par exemple, on suppose que des messages distincts chiffrés avec des clefs distinctes donnent deux chiffrés distincts, alors que, dans la réalité, même si la probabilité est faible, ces deux chiffrés pourraient être identiques. Ce type d'hypothèse conduit à modéliser les messages sous forme de termes. En particulier, supposons que m et m' représentent des messages et que k et k' représentent des clefs. Les deux messages chiffrés, notés $\{m\}_k$ et $\{m'\}_{k'}$, sont égaux si et seulement si $m = m'$ et $k = k'$. Une telle représentation sous forme de termes implique, de plus, que le chiffré d'un chiffré est distinct du message initial, même si la clef est symétrique : $\{\{m\}_k\}_k \neq m$.

Bien sûr, cette hypothèse ne correspond pas à la réalité : certains algorithmes de chiffrement permettent à l'intrus de connaître un message m à partir de son chiffré $\{m\}_k$, dès qu'il a une connaissance partielle de m (le début du message, par exemple)

ou s'il possède d'autres messages chiffrés avec la même clef k. Cependant, comme nous le verrons au paragraphe suivant, de nombreuses attaques de protocoles ne reposent absolument pas sur la façon dont sont chiffrés les messages mais consistent tout simplement à rejouer un ancien message ou à entrelacer plusieurs sessions. Il s'agit donc de vérifier les protocoles cryptographiques sans tenir compte des éventuelles failles dues au chiffrement. Nous verrons cependant que certains résultats récents permettent d'affaiblir cette hypothèse.

Il est à noter cependant que des travaux récents cherchent à rapprocher la cryptanalyse (basée sur la théorie de la complexité) et l'étude des protocoles cryptographiques sous l'hypothèse du chiffrement parfait. Ainsi, des résultats de M. Abadi et P. Rogaway [ABA 00b] font un lien entre ces deux notions dans le cadre d'un intrus passif (qui écoute les messages sans prendre part au protocole). D'autres travaux [CAN 01, PFI 00] ont pour but un résultat de composition de telle sorte que si les algorithmes de chiffrement sont robustes pour une notion basée sur la théorie de la complexité et si les protocoles cryptographiques sont sûrs sous l'hypothèse du chiffrement parfait alors les protocoles cryptographiques sont robustes pour une notion basée sur la théorie de la complexité. Cependant, ces travaux ne sont pas encore satisfaisants car la plupart ne conservent pas la simplicité du modèle abstrait pour les protocoles cryptographiques et perdent ainsi l'automatisation des preuves, ou bien ils ne capturent plus entièrement la notion de sécurité basée sur la théorie de la complexité.

3. 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 à clefs publiques [NEE 78] (*protocole 2*). De très nombreux autres exemples de protocoles sont décrits dans [CLA 97].

3.1. *Protocole*

Ce protocole spécifie des règles d'échange de messages entre deux *participants*, aussi appelés *agents*. Il se décrit de la façon suivante :

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

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 Bob (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 : $\{A, N_a\}_{\mathsf{pub}(B)}$.

À la deuxième étape du protocole, Bob reçoit le message $\{A, N_a\}_{\mathsf{pub}(B)}$ 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 $\{N_a, N_b\}_{\mathsf{pub}(A)}$ et reconnaît son nonce N_a. Comme seuls elle et Bob peuvent connaître ce nonce, elle 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 les nonces N_a et N_b et que ceux-ci permettent de les *authentifier* : lorsqu'Alice reçoit un message contenant N_a ou 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*. De plus, le protocole de Needham-Schroeder décrit ainsi deux programmes : les actions du premier participant (envoi du premier message et recopie du nonce du message reçu) et les actions du deuxième participant (réception d'un message et envoi d'un message avec copie du nonce reçu). Ces deux programmes sont appelés *rôles*. Si un troisième participant se présente, il pourra ainsi jouer le rôle de A, c'est-à-dire suivre le protocole vu par Alice ou jouer le rôle de B, c'est-à-dire suivre le protocole vu par Bob. Notons que Bob peut également jouer le rôle de A et inversement, cela signifie tout simplement qu'un participant peut initier une session d'un côté et répondre en même temps dans une autre session.

3.2. *Attaque*

Comme annoncé en introduction, on suppose que toutes les communications ont lieu en présence d'un intrus. Les premiers intrus considérés étaient *passifs* : ils se contentent d'écouter et d'analyser les messages circulant sur le réseau en déchiffrant les messages lorsqu'ils ont la clef correspondante.

Les intrus généralement considérés à l'heure actuelle sont actifs :

– ils interceptent tous les messages circulant sur le réseau,

– ils analysent les messages et en synthétisent de nouveaux,

– ils envoient des messages.

Ainsi G. Lowe [LOW 96] 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 ». Elle est schématisée à la figure 1. L'agent A commence spontanément une conversation avec un agent C, malhonnête. L'agent C 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 C vient de lui répondre. L'agent A, renvoie donc à C le nonce N_b que

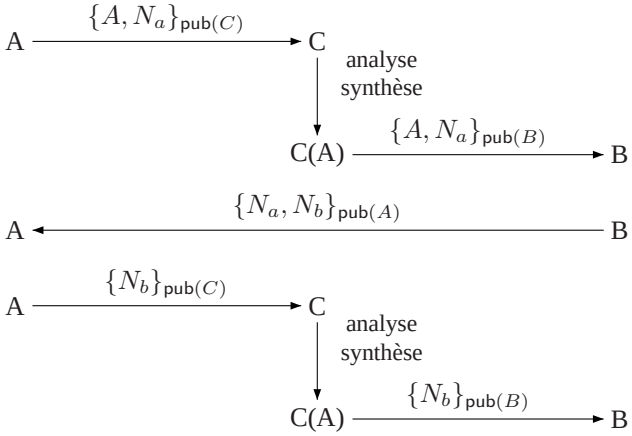


Figure 1. *Attaque du protocole de Needham-Schroeder à clefs publiques, due à G. Lowe.*

Variante du protocole de Needham-Schroeder-Lowe :

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

Attaque de la variante du protocole Needham-Schroeder-Lowe :

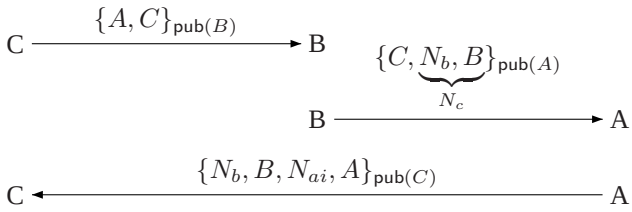


Figure 2. *Attaque de type du protocole de Needham-Schroeder-Lowe, due à J. Millen.*

l'agent C n'aurait pas dû connaître. L'agent C termine alors le protocole avec B qui croit avoir parlé à A.

G. Lowe a alors proposé la correction suivante au protocole de Needham-Schroeder :

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

L'agent B ajoute maintenant son identité dans le message qu'il envoie. Ce protocole modifié est appelé protocole de Needham-Schroeder-Lowe. On peut prouver [MIL 00] que le nonce N_b reste secret même lorsque le protocole est joué un nombre arbitraire de fois en présence d'un intrus. Si on le modifie très légèrement en changeant la place de B dans le deuxième message, on obtient un protocole sujet à une attaque dite de *confusion de type* découverte par J. Millen.

Le protocole et l'attaque sont décrits à la figure 2 : un agent malhonnête C envoie à B le message $\{A, C\}_{\mathsf{pub}(B)}$. L'agent B confond l'identité de C avec un nonce. Il pense donc répondre à A en lui envoyant le message $\{C, N_b, B\}_{\mathsf{pub}(A)}$. L'agent A en recevant ce message pense que C le sollicite en lui envoyant un nonce $N_c = N_b, B$. Il répond donc par le message $\{N_b, B, N_{ai}, A\}_{\mathsf{pub}(C)}$, ce qui permet à C d'apprendre le nonce N_b qui ne lui était pas destiné. Notons que cette attaque n'est pas très réaliste car la taille des identités et celle des nonces étant très différentes, B ne devrait pas pouvoir confondre C avec un nonce.

Un dernier type d'attaque très classique est l'attaque par *rejeu*. Comme son nom l'indique, une telle attaque consiste à renvoyer un message déjà joué à la place d'un autre. Considérons par exemple le protocole de Needham-Schroeder à clefs symétriques [NEE 78] (*protocole 1*) décrit à la figure 3. Ce protocole comporte deux participants A et B et un serveur, noté S. Le serveur peut être considéré comme un participant mais dont l'intrus ne peut pas prendre l'identité. Il partage en général des secrets avec chacun des participants, ici une clef symétrique, notée $\mathsf{shr}(A)$ pour la clef partagée entre le serveur et A. Une attaque sur ce protocole [CLA 97] consiste pour l'intrus à utiliser un ancien message $\{K'_{ab}, A\}_{\mathsf{shr}(B)}$ pour lequel il a réussi à obtenir la clef K'_{ab} et le jouer à la place de A à la troisième étape. Il n'y a alors aucun moyen pour B de savoir si la clef K'_{ab} est récente ou non et C peut se faire passer pour A puiqu'il répond au challenge de B. Cette attaque est présentée à la figure 3.

Nous venons de décrire deux attaques pour des propriétés de *secret* et également d'*authentification*. Mais les protocoles peuvent assurer d'autres propriétés de sécurité que nous décrivons au paragraphe suivant.

3.3. *Propriétés de sécurité*

Nous appellerons ici *propriété de sécurité* toute propriété qu'un protocole cherche à assurer. Il serait très intéressant de définir plus précisément ce qu'est une propriété de sécurité et, mieux, de définir une logique appropriée mais pour le moment, chaque propriété est définie dans un cadre spécifique. Le but de ce paragraphe est de faire une liste (non exhaustive) des propriétés que peuvent tenter d'assurer les protocoles.

3.3.1. *Secret*

On dira en général qu'un protocole assure le secret d'une donnée s si, pour tout déroulement valide du protocole, l'intrus ne peut pas apprendre cette donnée. Cependant, dans le cadre particulier des algèbres de processus [ABA 97], le secret d'une

Protocole de Needham-Schroeder à clefs symétriques

$$
\begin{aligned}
&1.\ A \rightarrow S : A, B, N_a \\
&2.\ S \rightarrow A : \{N_a, B, K_{ab}, \{K_{ab}, A\}_{\mathsf{shr}(B)}\}_{\mathsf{shr}(A)} \\
&3.\ A \rightarrow B : \{K_{ab}, A\}_{\mathsf{shr}(B)} \\
&4.\ B \rightarrow A : \{N_b\}_{K_{ab}} \\
&5.\ A \rightarrow B : \{N_b - 1\}_{K_{ab}}
\end{aligned}
$$

Attaque du protocole :

$$
\begin{aligned}
C(A) &\xrightarrow{\{K'_{ab}, A\}_{\mathsf{shr}(B)}} B \\
C(S) &\xleftarrow{\{N_b\}_{K'_{ab}}} B \\
C(A) &\xrightarrow{\{N_b - 1\}_{K'_{ab}}} B
\end{aligned}
$$

Figure 3. *Protocole de Needham-Schroeder à clefs symétriques et attaque par rejeu.*

donnée peut être défini par une propriété plus forte : on dit qu'une donnée s est secrète si la session qui contient la donnée s est indistinguable de toute session contenant une donnée s' en lieu et place de s. On dira que la première propriété de secret est une propriété d'*accessibilité* tandis que la seconde est une propriété d'*équivalence observationnelle*. D'autre part, parmi les propriétés de secret, on distingue les propriétés de secret globales et locales : on peut demander qu'une donnée soit secrète « tout le temps » ou bien qu'elle soit secrète « tant que la session correspondante n'est pas terminée ». La première notion est plus facile à modéliser puisqu'il suffit d'exprimer que l'intrus ne peut jamais déduire le secret. La seconde propriété demande un modèle plus précis qui permette d'exprimer les débuts et les fins de sessions.

3.3.2. *Authentification*

L'authentification admet de nombreuses définitions [SCH 97, LOW 97b, ABA 00a] et le lien entre celles-ci n'est pas clair. Ainsi, S. Schneider [SCH 97] dresse une liste d'une dizaine de définitions distinctes. Dans les grandes lignes, les propriétés d'authentification sont des propriétés de la forme : si l'agent B reçoit un message de la forme m_1 alors l'agent A a envoyé un message de la forme m_2 (qui est en correspondance avec le premier). Ainsi, dans le cas du protocole de Needham-Schroeder, on peut formuler la propriété suivante : si l'agent A émet le message $\{N_b\}_{\mathsf{pub}(B)}$, alors le nonce N_b a bien été engendré par l'agent B. Mais de nombreuses variantes sont possibles : on peut seulement demander que si B accepte le dernier message, alors A a été actif, ou on peut au contraire demander une propriété plus forte à savoir que si l'agent A émet n fois un message de la forme $\{N_b\}_{\mathsf{pub}(B)}$ alors l'agent B a engendré au moins n nonces N_b, *etc.* Comme les outils de vérification des protocoles crypto-

graphiques se consacrent presque tous à la vérification du secret, il serait intéressant de faire le lien entre le secret et l'authentification. Ainsi, au lieu de prouver que le protocole de Needham-Schroeder-Lowe satisfait la propriété d'authentification pour laquelle il a été conçu, on vérifie souvent une propriété de secret dont on pense qu'elle est équivalente. B. Blanchet [BLA 02] a fait un premier pas vers un lien formel entre les deux propriétés en associant à une propriété d'authentification particulière, une propriété de secret telle que prouver le secret suffit à assurer l'authentification. Certains auteurs [AMA 02b, COM 03a] traitent directement des propriétés d'authentification codées par une forme de négation du secret (*i.e.* à la fin du protocole, on demande qu'un certain message soit connu de l'intrus) ou à l'aide de formules d'implication : si l'intrus connaît un certain message, c'est qu'un message lui correspondant a été émis.

3.3.3. *Anonymat*

Les protocoles utilisés dans la téléphonie mobile visent aussi à assurer l'anonymat [SAM 95, HER 94] : une personne qui utilise son téléphone portable peut souhaiter qu'on ne puisse pas suivre ses déplacements en examinant les bornes relais qu'elle a utilisé pour transmettre ses conversations. En même temps, le service de téléphonie doit connaître l'identité de la personne pour lui facturer son service. Ce type de protocole doit donc assurer qu'une personne extérieure ne peut pas lier les messages émis à l'identité de la personne qui émet le message ou à l'identité du transmetteur. Ainsi, considérons le cas d'école où Alice signifie à Bob son intention de lui parler en lui envoyant son identité chiffrée avec la clef publique de Bob :

$$A \rightarrow B : \{A\}_{\mathsf{pub}(B)}.$$

Un tel protocole n'est pas anonyme. En effet, même si l'intrus ne connaît pas la clef privée de Bob, il peut former les messages $\{C\}_{\mathsf{pub}(B)}$ pour toutes les identités C possibles et les comparer avec le message $\{A\}_{\mathsf{pub}(B)}$ qu'il a vu circuler sur le réseau. Lorsqu'il obtient un message égal à celui envoyé, il déduit qui a envoyé le message. Ce problème d'anonymat est très proche du secret d'un vote : si à la place de l'identité A, on souhaite exprimer le nom d'un candidat, un intrus peut, de la même manière savoir pour qui on a voté. Une correction du protocole décrit ci-dessus est possible en ajoutant un nonce secret dans le message envoyé :

$$A \rightarrow B : \{A, N\}_{\mathsf{pub}(B)}.$$

En effet, l'intrus ne peut pas connaître le nonce N et il ne peut donc pas construire le message $\{A, N\}_{\mathsf{pub}(B)}$.

V. Shmatikov et D.J.D Hughes [SHM 02a] proposent une définition reposant sur l'équivalence observationnelle. En effet, intuitivement, un protocole préserve l'anonymat de l'identité du premier participant (par exemple) si, quel que soit l'agent qui joue le rôle du premier participant, les protocoles obtenus sont indistinguables.

3.3.4. *Équité*

Les signatures de contrat sur Internet amènent de nouveaux problèmes de sécurité. Le premier est la *non répudiation* [KRE 01]. En effet, comme les messages envoyés

sur Internet sont aisément modifiables, un agent peut dénier avoir envoyé ou reçu un message. Aussi, des protocoles spécifiques ont pour but d'établir à la fois un certificat d'émission et de réception des messages. Un deuxième problème des signatures de contrat sur un réseau tient au fait que les communications ne sont pas synchronisées. Aussi, lorsque la première personne a signé le contrat, la deuxième personne peut utiliser cette signature pour obtenir par exemple un meilleur prix auprès d'une tierce personne. Des exemples de protocoles de signature de contrat sont présentés dans [ASO 98, GAR 99]. Nous décrivons rapidement l'un d'entre eux :

$$\begin{aligned} A &\rightarrow B : \quad PCS_A(m, B, T) \\ B &\rightarrow A : \quad PCS_B(m, A, T) \\ A &\rightarrow B : \quad S\text{-}Sig_A(m) \\ B &\rightarrow A : \quad S\text{-}Sig_B(m). \end{aligned}$$

$PCS_A(m, B, T)$: promesse de signature de A pour le contrat m avec B.

$S\text{-}Sig_A(m)$: signature du contrat m par A.

L'idée de ce protocole est que chaque participant signe d'abord une promesse de signature pour le contrat m. Ils peuvent alors encore chacun annuler l'opération en lançant des sous-protocoles. Mais une fois que l'agent A s'est engagé en signant le contrat, il peut forcer la signature de B grâce d'une part à sa signature $S\text{-}Sig_A(m)$ et d'autre part à la promesse $PCS_B(m, A, T)$ de B, à l'aide d'un tiers de confiance.

Les deux participants peuvent éventuellement ne pas suivre complètement le protocole pour tenter d'obtenir un avantage sur l'autre. Ces protocoles de signature de contrat doivent donc assurer qu'aucun des participants n'est avantagé par rapport à l'autre. Plusieurs définitions formelles ont été proposées [SHM 02b, CHA 01, KRE 02]. En particulier, S. Kremer et J.-F. Raskin utilisent une définition basée sur la théorie des jeux : un protocole est *équitable* pour le participant Alice si le participant Bob n'est pas de stratégie gagnante pour obtenir le contrat signé par Alice à moins qu'Alice ait elle aussi une stratégie gagnante pour obtenir le contrat signé par Bob.

4. Difficultés de la vérification des protocoles

4.1. *Modélisation*

Comme on peut s'en rendre compte au travers de la description de différentes propriétés de sécurité, la première difficulté des protocoles cryptographiques est de donner une définition formelle des propriétés à vérifier. Ainsi, pour la plupart des propriétés mentionnées au paragraphe précédent, le travail de modélisation est encore en cours à l'heure actuelle. C'est pourquoi nous ne considérerons ici que des propriétés de secret et parfois d'authentification. Mais même lorsqu'on se restreint à l'étude de

ces deux dernières propriétés, de nombreux modèles existent pour décrire le comportement d'un protocole et les données à protéger.

Pour décrire les protocoles, nous avons jusqu'ici utilisé la représentation intuitive employée dans [CLA 97]. Cette présentation est en fait très ambiguë car seul le déroulement *normal* du protocole est décrit. Pour le protocole de Needham-Schroeder, on ne sait pas si le deuxième agent teste si le nonce reçu est bien un nonce, ce qu'il fait si le message n'est pas de la forme attendue ni surtout ce qu'est exactement la « forme attendue » par l'agent. Nous distinguons deux grandes familles de modèles : les modèles de traces et les algèbres de processus.

Les modèles de traces représentent les protocoles et l'intrus par des règles d'inférence. Ces règles décrivent les transitions possibles entre *traces*. Les traces contiennent en général l'ensemble des messages échangés sur le réseau et éventuellement les états locaux des participants ou des annotations décrivant explicitement les transitions effectuées. Ces modèles de traces utilisent des règles de réécriture [CER 99, RUS 01], des clauses de Horn [BLA 03, COM 03a] ou des constructions dédiées comme les strand spaces [THA 99] ou le modèle de L. Paulson [PAU 97].

Un autre type de modélisation est la représentation des protocoles par des processus. Cela correspond à une modélisation plus proche de l'implémentation des protocoles. En effet, chaque rôle est représenté par un processus indépendant des processus représentant les autres rôles. Ces algèbres de processus (comme CSP [SCH 97] ou le spi-calcul [ABA 97]) permettent en général les opérations suivantes :

$P :=$	$\mathbf{0}$	processus nul
	$\mid (\nu n).P$	création du nonce n
	$\mid \bar{p} < m > .P$	envoi du message m sur le canal p
	$\mid p(x).P$	réception du message x sur le canal p
	$\mid P_1 \mid P_2$	composition parallèle
	$\mid !P$	réplication
	$\ldots$	

La principale différence entre les différents modèles utilisant des algèbres de processus est la modélisation de la propriété de sécurité. La plupart des modèles [AMA 02b, SCH 97] ramènent les propriétés de sécurité à une propriété d'accessibilité de la forme : $P \xrightarrow{*} err$. Quelques autres [ABA 97, BOR 99] représentent le secret par exemple sous forme d'une équivalence observationnelle. Cette propriété est nettement plus difficile à démontrer et il existe très peu de résultats de décidabilité à ce sujet. Aussi, dans la suite de cet article et sauf mention explicite, nous appellerons secret la propriété de secret définie par une notion d'accessibilité.

Chacun des résultats de décidabilité sont décrits dans un de ces modèles en particulier. Cependant, on se convainc généralement que ces résultats sont également valables dans les autres modèles. Dans certains cas, des traductions formelles sont proposées d'un modèle vers l'autre. Ainsi, l'article [BIS 03] montre que les strand spaces peuvent être traduits en algèbre de processus et réciproquement, en préservant les propriétés de secret et certaines propriétés d'authentification.

4.2. *Sources d'indécidabilité*

La vérification des protocoles cryptographiques est un cas particulier de model-checking où les systèmes considérés sont des protocoles cryptographiques dans un réseau hostile et les propriétés à vérifier sont celles énoncées aux paragraphes précédents (secret, authentification, anonymat). Les difficultés de la vérification tiennent dans le caractère non borné des paramètres du système à vérifier :

– le nombre de sessions n'est pas borné ;

– le nombre de participants n'est pas borné ;

– les messages sont de taille arbitraire ;

– à chaque étape, n'importe quel message de la connaissance de l'intrus peut être envoyé : le système est à branchement infini ;

– l'intrus peut générer un nombre arbitraire de nouvelles clefs et de nouveaux nonces.

Il n'est donc pas possible de chercher à vérifier un protocole en énumérant naïvement tous les comportements possibles : un tel algorithme ne terminerait pas.
Cependant, un résultat de réduction [COM 03b] montre qu'il est inutile de considérer un nombre arbitraire d'agents : un nombre fini d'agents, dépendant de la propriété à vérifier, suffit. Comme nous allons le voir, le secret est décidable si le nombre de sessions est fixé à l'avance. Lorsque le nombre de sessions est arbitraire, les deux principales sources d'indécidabilité sont la création de nonces et la possibilité de copier des messages arbitraires.

Ainsi, même en considérant des protocoles où aucune clef ni aucun nonce n'est généré au cours des sessions, on peut encoder le problème de correspondance de Post [DAV 83] comme l'ont fait S. Even et O. Golreich dans [EVE 83]. Nous présentons rapidement un de ces codages, suggéré par M. Rusinowitch.

Soient Σ un alphabet fini et $(u_i, v_i)_{1 \leqslant i \leqslant n}, u_i, v_i \in \Sigma^*$ une entrée du problème de correspondance de Post. On considère le protocole à trois règles défini ci-dessous :

$$\begin{aligned} A &\rightarrow B : \{< 0, 0 >\}_{K_{ab}} \\ B &\rightarrow A : \{< N_1, N_2 >\}_{K_{ab}} \\ A : \{< x, y >\}_{K_{ab}} &\rightarrow B : \{< xu_i, yv_i >\}_{K_{ab}}, \{s\}_{\{<xu_i, xu_i>\}_{K_{ab}}} \qquad 1 \leqslant i \leqslant n \end{aligned}$$

Dans la troisième règle, A envoie à B la concaténation des messages formés pour chaque couple (u_i, v_i). La partie gauche de la troisième règle décrit le message attendu par A. Les symboles N_1, N_2 et s représentent des constantes, K_{ab} une clef long terme, secrète entre A et B. On peut alors montrer que s est secret si et seulement s'il n'y a pas de solution au problème de correspondance de Post.

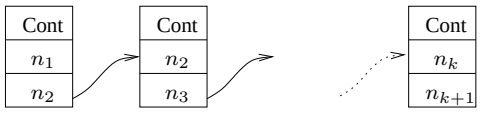
Ce codage présente l'inconvénient d'utiliser des clefs composées, c'est-à-dire des clefs construites à partir d'autres messages. Les clefs composées sont en particulier utilisées dans les algorithmes de distribution de clefs de groupe où la clef est construite

à partir d'informations détenues par chacun des membres du groupe. Dans l'exemple considéré ci-dessus, le secret s est chiffré avec la clef $\{< xu_i, xu_i >\}_{K_{ab}}$. Cependant, il n'est pas très « réaliste » qu'un protocole construise une telle clef composée. On peut pallier cela en modifiant le protocole de manière à ce que l'agent A teste lui-même l'égalité des éléments de la paire reçue :

$$\begin{aligned} A &\rightarrow B : \{< 0, 0 >\}_{K_{ab}} \\ B &\rightarrow A : \{< N_1, N_2 >\}_{K_{ab}} \\ A : \{< x, y >\}_{K_{ab}} &\rightarrow B : \{< xu_i, yv_i >\}_{K_{ab}} \qquad 1 \leqslant i \leqslant n \\ A : \{0x, 0x\} &\rightarrow A : s \end{aligned}$$

L'inconvénient est alors que la dernière règle de ce protocole peut être jouée uniquement si le protocole correspond à une entrée qui est solution du problème de correspondance de Post. Un autre codage sans clefs composées est obtenu en utilisant l'indécidabilité de l'accessibilité dans les réseaux de Petri avec remise à zéro [COM 01]. Il est à noter que même s'il ne semble pas très réaliste de considérer des protocoles qui codent des problèmes tels que le problème de correspondance de Post, il n'est pas évident de déterminer des caractéristiques qui permettraient d'exlure ces protocoles sans exclure les protocoles réels.

Inversement, N. Durgin *et al.* [DUR 99] ont montré que le secret des protocoles avec nonces, mais tels que la profondeur des messages est bornée, est indécidable. R. Amadio et W. Charatonik [AMA 02a] raffinent ce résultat en montrant que le secret des protocoles avec nonces mais n'utilisant que la primitive de chiffrement (et non la paire) et tels que la profondeur des messages est bornée, est aussi indécidable. Pour cela, ils codent une machine à deux compteurs. Pour représenter qu'un compteur contient la valeur n, ils n'utilisent pas un terme de taille n mais n termes de taille bornée. Les nonces servent alors de « pointeurs » pour relier les différents termes entre eux, comme cela est schématiquement représenté ci-après.



En réalité, le codage est un peu plus technique car il faut s'assurer qu'il n'y a pas d'interférences entre les différentes configurations de la machine à deux compteurs.

5. Résultats de décidabilité

La plupart des résultats que nous allons décrire sont résumés dans le tableau de la figure 4.

<table>
<tr><th rowspan="2">Nombre de sessions</th><th rowspan="2">Borné</th><th colspan="3">Non borné</th></tr>
<tr><th colspan="2">Sans nonce</th><th>Avec nonces</th></tr>
<tr><td rowspan="4">Hypothèse du chiffrement parfait</td><td rowspan="4">co-NP-complet
[RUS 01]</td><td colspan="2">Profondeur bornée :
DEXPTIME-complet
[DUR 99, CHE 03c]</td><td>Profondeur bornée :
Indécidable
[DUR 99]</td></tr>
<tr><td colspan="2">Avec des étiquettes :
EXPTIME
[BLA 03]</td><td>Fortement typé :
Décidable
[LOW 98]</td></tr>
<tr><td>Une copie</td><td>Deux copies ou plus</td><td rowspan="2">Indécidable</td></tr>
<tr><td>3-EXPTIME
[COM 03a]
(+authenti-fication)</td><td>Indécidable
[COM 04]</td></tr>
<tr><td>Ajout du « ou » exclusif</td><td>co-NP-complet
[CHE 03a]</td><td>Décidable
[COM 03a]
(+authenti-fication)</td><td>Indécidable</td><td>Indécidable</td></tr>
<tr><td>Ajout de l’exponentiation modulaire ou du chiffrement par bloc</td><td>co-NP-complet
[CHE 03a, COM 03c]
[CHE 03b]</td><td colspan="2">?</td><td>Indécidable</td></tr>
</table>

Figure 4. *Tableau récapitulatif des résultats de décidabilité pour le secret.*

5.1. *Nombre de sessions bornés*

Dans le cadre d’un nombre borné de sessions, les résultats sont maintenant assez nombreux. Citons tout d’abord le résultat de M. Rusinowitch et M. Turuani [RUS 01] qui montrent dans un cadre très général (clefs composées, chiffrement symétrique et asymétrique) que la décision du secret est un problème co-NP-complet. Pour cela, ils montrent que pour construire une attaque, l’intrus n’a besoin que de termes de taille polynomiale pourvu que ceux-ci soient représentés par des graphes acycliques orientés.

Dans le contexte des algèbres de processus, R. Amadio *et al.* [AMA 02b] avaient montré un résultat similaire mais sans modéliser les clefs composées. Des résultats de décidabilité très proches sont établis, toujours pour des processus sans réplication par [BOR 01, FIO 01, MIL 01]. H. Huttel [HUT 02] montre également la décidabilité du secret pour des processus sans réplication mais pour une propriété d'équivalence observationnelle.

Affaiblissement de l'hypothèse du chiffrement parfait

L'un des plus anciens travaux sur le sujet est celui de S. Even *et al* [EVE 86] où les auteurs étudient les protocoles ping-pong utilisant du chiffrement RSA. Ils montrent qu'il est inutile de considérer les propriétés de RSA : si un protocole peut être attaqué en utilisant les faiblesses du chiffrement RSA, alors il existe une attaque qui n'utilise pas ces faiblesses.

D'autres travaux beaucoup plus récents permettent d'affaiblir l'hypothèse du chiffrement parfait en tenant compte des propriétés algébriques du « ou exclusif », aussi dit `xor`, très utilisé dans les algorithmes de chiffrement. Par exemple, si m est un message et k une clef, le `xor` du message et de la clef permet un chiffrement rapide mais qui comporte de nombreuses faiblesses (associativité, commutativité et nilpotence) dont il faut pouvoir tenir compte.
Ainsi, M. Rusinowitch *et al.* [CHE 03a] montrent que la décision du secret reste co-NP-complet en tenant compte des propriétés algébriques du « ou exclusif » : ils prouvent à nouveau que l'intrus n'a besoin que de termes de taille polynomiale pourvu que ceux-ci soient représentés par des dags.
H. Comon-Lundh et V. Shmatikov [COM 03c] montrent que la décision du secret est également décidable dans un cadre plus général. Ils associent aux protocoles un système de contraintes puis ils montrent comment résoudre tout système de contraintes de ce type.

Un autre opérateur algébrique est l'exponentiation modulaire, en particulier utilisée dans le protocole de Diffie-Hellman. M. Rusinowitch *et al.* [CHE 03b] montrent à nouveau que la décision du secret reste co-NP-complet. Cet opérateur a également été étudié dans un cadre plus restreint par J. Millen et V. Shmatikov [MIL 03].

Le chiffrement par bloc a également été étudié [CHE 03a] en ajoutant la règle suivante $\{u, v\}_k \rightarrow \{u\}_k$: un intrus peut obtenir n'importe quel préfixe d'un message chiffré. La décision du secret est à nouveau co-NP-complet.

D'autre part, la faiblesse du chiffrement ne tient pas seulement aux faiblesses de l'algorithme utilisé mais également aux « faiblesses » de l'utilisateur. Ainsi, un mot de passe choisi par un utilisateur est rarement un mot aléatoire mais est en général pris dans un ensemble de mots restreint car l'utilisateur veut s'en souvenir facilement. Il faut donc pouvoir également vérifier des protocoles où certains messages sont chiffrés par une clef obtenue à partir d'un *mot de passe faible*. Le même problème se pose pour le secret d'un vote où l'intrus connaît les valeurs possibles du vote. Plusieurs auteurs [GON 93, LOW 02, DEL 03] ont proposé une modélisation du pouvoir de l'intrus à deviner certaines données. De plus, S. Delaune [DEL 03] a montré qu'étant

donné un ensemble fini de messages S et un message s, savoir si l'intrus pouvait déduire s de T en devinant éventuellement certaines données « faibles » est décidable en temps polynomial.

5.2. *Nombre de sessions non borné*

Dans le cadre d'un nombre de sessions non borné, il faut restreindre assez fortement les classes de protocoles pour obtenir des résultats de décidabilité.

Avec nonces

À notre connaissance, seuls deux résultats de décidabilité [LOW 98, RAM 03] permettent de traiter les nonces dans toute leur généralité. G. Lowe [LOW 98] a montré que pour le secret, on peut se restreindre à l'étude des protocoles tels que les rôles sont joués chacun au plus une fois, par des agents distincts et honnêtes et tels que, de plus, aucun agent honnête ne parle à un agent malhonnête. Pour cela, les protocoles considérés doivent satisfaire un certain nombre d'hypothèses dont nous décrivons les principales.

– Les messages reçus (et envoyés) par les agents sont entièrement spécifiés, à renommage des noms d'agents, de nonces et de clefs près et les agents doivent être capable de distinguer un nonce d'une clef par exemple.

– Chaque message doit contenir (sous chiffrement) l'identité de celui qui est supposé envoyer le message.

– Les clefs sont atomiques ou de la forme $\mathsf{shr}(a)$ ou $\mathsf{pub}(a)$: les clefs composées ne sont pas représentées.

– Les seuls symboles de fonctions applicables à des messages sont la paire et le chiffrement.

– Les secrets « à long terme » (comme les clefs privées) doivent être toujours en position de clefs.

– Il ne doit pas y avoir de confusion possible entre des messages provenant d'étapes différentes du protocole.

– Si une valeur n'est pas déclarée secrète, alors l'intrus la connaît dès qu'elle est créée : il n'y a pas de donnée temporairement secrète.

Il est à noter que la capacité des agents à distinguer chaque type de message (nonce, clef, message chiffré, ...) peut-être assurée par l'ajout d'étiquettes [HEA 00] dans chaque composante d'un message. Ces hypothèses sont donc raisonnables pour certains protocoles mais ne permettent pas de considérer par exemple des protocoles où un agent renvoie une partie inconnue d'un message reçu.

R. Ramanujam et S. P. Suresh [RAM 03] considèrent des protocoles où chaque composante chiffrée d'un message est étiquetée par un numéro d'étape et un nonce. Cet étiquetage alourdit un peu l'implémentation des protocoles puisqu'il nécessite l'ajout de nonces mais il permet d'obtenir la décidabilité du sercet si deux autres

conditions sont respectées :

– les clefs sont atomiques,

– chaque participant doit pouvoir complètement décomposer les messages reçus. Cela exclut en particulier les protocoles avec « tickets ».

Sans nonces

La plupart des auteurs considèrent des protocoles tels qu'aucune donnée fraîche n'est engendrée au cours des sessions. Les nonces sont ainsi remplacés par des constantes ou une fonction des messages reçus. Une telle abstraction est correcte pour le secret : si le protocole abstrait préserve le secret demandé alors le protocole « réel » le préserve également.

Cependant, nous avons vu au paragraphe 4.2 que même avec cette abstraction, le secret est indécidable. C'est pourquoi les auteurs considèrent deux types de restriction : borner la taille des messages ou limiter les copies à chaque transition.

Ainsi, N. Durgin *et al.* [DUR 99] ont montré que le secret est DEXPTIME-complet si la profondeur des messages est bornée. M. Rusinowitch *et al.* [CHE 03c] généralisent ce résultat en l'unifiant avec le cas où le nombre de sessions est borné [RUS 01]. Ils considèrent des protocoles avec un nombre borné de sessions (où les messages utilisés sont de taille arbitraire) et tels que l'intrus possède des « règles d'oracle » qui peuvent à la fois simuler un pouvoir supplémentaire de l'intrus et un nombre arbitraire de sessions où les messages utilisés sont de taille bornée. Ils montrent que dans ce cas, la décision du secret reste DEXPTIME-complet.

R. Amadio *et al.* [AMA 02b] considèrent des protocoles itérés où chaque protocole atomique consiste en une réception de message, une suite de déchiffrements puis de chiffrements avec des clefs constantes (symétriques ou asymétriques) puis l'envoi du message obtenu. Il n'est donc pas possible de tester des messages quelconques ni de former des paires dans les messages. D'autre part, comme les protocoles sont itérés, le nombre de sessions parallèles est borné. Sous ces hypothèses, ils montrent que l'accessibilité de l'état « erreur » est décidable en temps polynomial. Le secret peut en particulier être traduit en un problème d'accessibilité de l'état « erreur » en ajoutant en parallèle un processus qui teste la réception du secret et se met dans l'état « erreur » si c'est le cas.

H. Comon-Lundh et V. Cortier [COM 03a] montrent plus généralement que le secret est décidable en au plus 3-EXPTIME pour des protocoles tels qu'au plus une partie inconnue du message (distincte d'un nom d'agent ou d'un nonce ou d'une clef déjà connus) est testée et/ou copiée à chaque transition. La plupart des protocoles vérifient cette hypothèse, en particulier le protocole de Needham-Schroeder décrit au paragraphe 3. Pour montrer ce résultat, les auteurs développent un fragment décidable de la logique du premier ordre permettant de représenter les protocoles tels qu'au plus une partie inconnue du message est testée et/ou copiée à chaque transition. La décidabilité de ce fragment est prouvée en fournissant un système de résolution ordonnée, correct, complet et terminant en au plus 3-EXPTIME.

Deux extensions de ce résultat existent. Une preuve beaucoup plus complexe permet d'obtenir la décidabilité du secret lorsque plusieurs copies sont autorisées [COM 04], sous une hypothèse de non entrelacement. D'autre part, dans le cas où un seul message est testé et/ou copié à chaque transition, le secret reste décidable en ajoutant les propriétés algébriques du `xor` mais la borne de complexité donnée est alors non élémentaire.

Enfin, B. Blanchet et A. Podelski [BLA 03] considèrent des protocoles étiquetés : chaque message ou sous-message chiffré doit contenir une étiquette (*i.e.* une constante) permettant de l'identifier. Les agents doivent vérifier la cohérence de tous tags auxquels ils ont accès. Ainsi le protocole de Needham-Schroeder étiqueté est le suivant :

$$\begin{aligned} A &\to B : \quad \{c_1, A, N_a\}_{\mathsf{pub}(B)} \\ B &\to A : \quad \{c_2, N_a, N_b\}_{\mathsf{pub}(A)} \\ A &\to B : \quad \{c_3, N_b\}_{\mathsf{pub}(B)} \end{aligned}$$

où c_1, c_2, c_3 sont trois constantes distinctes. Les étiquettes évitent en particulier de confondre un message provenant de la première étape d'une session d'un message provenant de la troisième étape d'une session. Pour cette classe de protocoles, B. Blanchet et A. Podelski montrent que le secret est décidable en temps au plus exponentiel. Pour ce faire, ils traduisent ces protocoles en clauses de Horn et fournissent, pour le fragment de clauses de Horn obtenu, une stratégie de résolution par sélection qui termine. Cette méthode a été implémentée pour vérifier les protocoles. Il est à noter qu'ajouter des étiquettes aux protocoles renforce la capacité des protocoles à préserver le secret : si le protocole original préserve le secret demandé alors le protocole étiqueté le préserve également mais la réciproque est fausse. Aussi, il est intéressant en pratique d'étiqueter les protocoles mais pour ceux qui ont été implémentés sans étiquettes, ce résultat de vérification ne peut pas s'appliquer.

6. Outils pour vérifier les protocoles en pratique

On peut distinguer deux familles d'outils : ceux consacrés à la recherche d'attaque et qui ne considèrent qu'un nombre borné de sessions et ceux consacrés à la preuve des protocoles, pour un nombre non borné de sessions. Nous allons présenter certains de ces outils en quelques mots. Toutefois l'ensemble des outils présentés ici est loin d'être une liste exhaustive des logiciels existants.

6.1. *Outils consacrés à la recherche d'attaques*

Dans le cas d'un nombre borné de sessions, nous avons vu que le secret est décidable. Cependant, il n'est toujours pas possible d'énumérer naïvement tous les comportements possibles car à chaque étape, n'importe quel message constructible par

l'intrus peut-être *a priori* envoyé et l'ensemble des messages constructibles par l'intrus est en général infini. Ces outils visent donc à énumérer de manière « économique » les comportements possibles.

Casper/FDR. Cet outil a été développé par G. Lowe [LOW 97a, RYA 00] pour vérifier les protocoles décrits en CSP [SCH 96a]. Le nombre de sessions et de participants est borné, la taille des messages est également bornée. De plus, les messages sont typés et il n'y a pas de clefs composées. La vérification du protocole correspond alors au model-checking d'un modèle fini. La principale difficulté est le traitement de l'explosion du nombre d'états lorsqu'on considère plusieurs sessions par exemple. C'est en particulier le model-checking avec FDR du protocole de Needham-Schroeder qui a permis à G. Lowe de trouver son attaque sur ce protocole [LOW 96] puis d'en proposer une nouvelle version.

MURϕ. L'outil MURϕ [MIT 97] est assez proche de Casper. Il traite lui aussi des protocoles avec un nombre de sessions et une taille des messages bornée. À nouveau, les messages sont typés. Le modèle utilisé suppose que l'intrus a une mémoire finie. L'outil permet de retrouver quelques unes des attaques classiques mais il ne parvient pas à traiter plus de deux ou trois sessions en parallèle. Cependant, il n'est pas uniquement dédié à la vérification des protocoles cryptographique. Il peut également vérifier la « cohérence de cache » de protocoles (pas nécessairement cryptographiques) [STE 95].

Casrul. L'outil Casrul [JAC 00] permet également 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 [CLA 97].

Aviss. Le logiciel Aviss [ARM 02] est un logiciel assez récent qui traduit les protocoles en un langage commun à trois logiciels de vérification. L'utilisation commune de ces trois logiciels permet de retrouver de très nombreuses attaques déjà connues (la majorité de celles mentionnées dans [CLA 97]) et a également permis de découvrir une attaque nouvelle sur le protocole de Denning-Sacco.

6.2. *Outils consacrés à la preuve*

Pour *prouver* des propriétés sur les protocoles, il est nécessaire de considérer un nombre arbitraire de sessions. Comme nous l'avons vu au paragraphe 5.2, les classes décidables de protocoles permettant de considérer un nombre non borné de sessions sont encore assez restreintes. Pourtant, de nombreux outils permettent de vérifier des protocoles n'appartenant pas à des classes décidables identifiées et les résultats obtenus en pratique sont satisfaisants.

Athena. Les protocoles sont exprimés dans le modèle des *strand spaces* [THA 99] et l'outil Athena [SON 99] suppose que les messages sont typés et qu'il n'y a pas de clefs composées. Lorsque l'outil termine, il permet soit de prouver la propriété demandée, soit de fournir un contre-exemple.

Cryptic Cet outil [GOR 01] permet de prouver des propriétés d'authentification. Chaque protocole est annoté en fonction de la propriété voulue et des types sont associés aux nonces, aux clefs et aux messages. Le but de l'outil est alors de vérifier que le protocoles est bien typé, ce qui est une condition suffisante pour garantir la propriété demandée.

Blanchet. B. Blanchet [BLA 01] exprime les protocoles sous forme de règles Prolog. Il vérifie le secret pour un nombre non borné de sessions. 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.

Hermes. Comme l'outil précédent, Hermes [BOZ 03] permet de prouver le secret de manière complètement automatique, pour un nombre non borné de sessions et de participants. Les clefs doivent également être atomiques et toutes les variables utilisées pour chiffrer doivent être de type clef. Pour vérifier le secret, cet outil commence par abstraire les nonces par un nombre fini de constantes en distinguant une session particulière de toutes les autres, les nonces des autres sessions étant confondus. De plus, il se ramène par abstraction à un nombre fini de participants. Il calcule alors sur le modèle abstrait une approximation des messages protégés et une approximation des messages potentiellement connus de l'intrus. Il reste alors à vérifier qu'il n'y a pas d'incompatibilité entre ces deux ensembles de messages. Cette méthode permet de montrer le secret de nombreux protocoles de [CLA 97].

Securify. L'outil Securify [COR 02, COR 01] cherche à vérifier que toutes les règles du protocole préservent les secrets demandés. Pour ce faire, l'algorithme utilise la notion d'*idéal* développée par L. Paulson [PAU 98] qui consiste en une surapproximation des secrets à protéger, stable par application des règles de déduction de l'intrus.

Timbuk. Cet outil permet de calculer une approximation d'un ensemble de messages reconnaissable par un automate d'arbre, modulo des règles de réécriture. Il a en particulier permis de vérifier un protocole de protection contre la copie [GEN 03]. Il est à noter que dans ce cas, cet outil a permis de vérifier une propriété autre que le secret.

7. Conclusions et perspectives

La vérification des protocoles cryptographiques est un cas particulier de la vérification des systèmes infinis. Nous avons dressé une liste que nous espérons assez

complète des résultats de vérification concernant les protocoles pour des propriétés de secret et d'authentification. Cependant, comme nous l'avons souligné au paragraphe 6, la vérification des protocoles fonctionne assez bien même pour des protocoles n'appartenant pas à des classes décidables identifiées. Il semble donc possible de caractériser des classes décidables encore plus générales, en capturant en particulier la notion de nonces.

D'autre part, deux axes de recherche sont explorés à l'heure actuelle : l'affaiblissement de l'hypothèse du chiffrement parfait et la vérification de propriétés autres que le secret et l'authentification. En effet, quelques résultats existent pour les propriétés du « ou exclusif » mais il serait intéressant d'obtenir des résultats de décidabilité tenant compte de propriétés algébriques plus générales, incluant par exemple les propriétés de l'exponentiation modulaire. De plus, de la même manière qu'il est intéressant de vérifier les protocoles avec mots de passe faibles, il est pertinent de considérer d'autres faiblesses dues, non pas à l'algorithme de chiffrement, mais à l'utilisation qu'on fait du protocole. Ainsi, les nonces sont supposés être des nombres engendrés aléatoirement mais ils sont parfois remplacés par des compteurs, plus faciles à réaliser. C'est en particulier le cas sur les cartes à puces. Il faut donc pouvoir adapter les vérifications à ce type de protocoles.

En outre, nous n'avons décrit ici que des résultats de décidabilité pour le secret et l'authentification mais de nombreuses autres propriétés de sécurité sont pertinentes, nous en avons présenté quelques exemples au paragraphe 3.3. Il faut donc également mettre en place des algorithmes de vérification pour ces propriétés.

8. Bibliographie

[ABA 97] ABADI M., GORDON A. D., « A Calculus for Cryptographic Protocols : The Spi Calculus », *Proc. of the 4th ACM Conference on Computer and Communications Security*, ACM Press, 1997, p. 36-47.

[ABA 00a] ABADI M., FOURNET C., GONTHIER G., « Authentication primitives and their compilation », *Proc. of the 27th ACM Sympoium on Principles of Programming Languages (POPL'00)*, 2000, p. 302-315.

[ABA 00b] ABADI M., ROGAWAY P., « Reconciling two views of cryptography », *Proc. of the International Conference on Theoretical Computer Science (IFIP TCS 2000)*, August 2000, p. 3-22.

[AMA 02a] AMADIO R., CHARATONIK W., « On name generation and set-based analysis in the Dolev-Yao model », *Proc. of the 13th International Conference on Concurrency Theory (CONCUR'02)*, LNCS, Springer Verlag, 2002, p. 499-514.

[AMA 02b] AMADIO R., LUGIEZ D., VANACKÈRE V., « On the symbolic reduction of processes with cryptographic functions », *Theoretical Computer Science*, vol. 290, n° 1, 2002, p. 695–740.

[ARM 02] ARMANDO A., BASIN D., BOUALLAGUI M., CHEVALIER Y., COMPAGNA L., BÖDERSHEIM S., RUSINOWITCH M., TURUANI M., VIGANÒ L., VIGNERON L., « The AVISS Security Protocol Analysis Tool », *Proc. of the 14th International Conference of Computer Aided Verification (CAV 2002)*, vol. 2404 de *LNCS*, Springer, 2002, p. 349-353.

[ASO 98] ASOKAN N., SHOUP V., WAIDNER M., « Optimistic Fair Exchange of Digital Signatures », *Advances in Cryptology - EUROCRYPT '98*, vol. 1403 de *LNCS*, 1998, p. 591-606.

[BIS 03] BISTARELLI S., CERVESATO I., LENZINI G., MARTINELLI F., « Relating Process Algebras and Multiset Rewriting for Security Protocol Analysis », *Proc. of the Third Workshop on Issues in the Theory of Security (WITS'03)*, Warsaw, Poland, 2003, p. 21-31.

[BLA 01] BLANCHET B., « An efficient Cryptographic Protocol Verifier Based on Prolog Rules », *Proc. of the 14th Computer Security Foundations Workshop (CSFW'01)*, IEEE Computer Society Press, June 2001.

[BLA 02] BLANCHET B., « From Secrecy to Authenticity in Security Protocols », *9th International Static Analysis Symposium (SAS'02)*, vol. 2477 de *LNCS*, Springer-Verlag, September 2002, p. 242-259.

[BLA 03] BLANCHET B., PODELSKI A., « Verification of Cryptographic Protocols : Tagging Enforces Termination », GORDON A., Ed., *Foundations of Software Science and Computation Structures (FoSSaCS'03)*, vol. 2620 de *LNCS*, April 2003.

[BOR 99] BOREALE M., DE NICOLA R., PUGLIESE R., « Proof Techniques for Cryptographic Processes », *Logic in Computer Science*, 1999, p. 157-166.

[BOR 01] BOREALE M., « Symbolic trace analysis of cryptographic protocols », *Proc. of the 28th Int. Coll. Automata, Languages, and Programming (ICALP'01)*, Springer Verlag, July 2001.

[BOZ 03] BOZGA L., LAKHNECH Y., PÉRIN M., « Pattern-based abstraction for verifying secrecy in protocols », *Tools and Agorithms for the Construction and Analysis of Sytems (TACAS'03)*, vol. 2619 de *LNCS*, 2003.

[CAN 01] CANETTI R., FISCHLIN M., « Universally Composable Commitments », *CRYPTO 2001*, Santa Barbara, California, 2001, p. 19-40.

[CER 99] CERVESATO I., DURGIN N., LINCOLN P., MITCHELL J., SCEDROV A., « A meta-notation for protocol analysis », *Proc. of the 12th IEEE Computer Security Foundations Workshop (CSFW'99)*, IEEE Computer Society Press, 1999, p. 55-69.

[CHA 01] CHADHA R., KANOVICH M., SCEDROV A., « Inductive methods and contract-signing protocols », SAMARATI P., Ed., *8-th ACM Conference on Computer and Communications Security*, ACM Press, novembre 2001, p. 176-185.

[CHE 03a] CHEVALIER Y., KÜSTERS R., RUSINOWITCH M., TURUANI M., « An NP Decision Procedure for Protocol Insecurity with XOR », *Proc. of 18th Annual IEEE Symposium on Logic in Computer Science (LICS '03)*, 2003.

[CHE 03b] CHEVALIER Y., KÜSTERS R., RUSINOWITCH M., TURUANI M., VIGNERON L., « Deciding the Security of Protocols with Diffie-Hellman Exponentiation and Product in Exponents », *Proc. of the 23rd Conference on Foundations of Software Technology and Theoretical Computer Science*, 2003.

[CHE 03c] CHEVALIER Y., KÜSTERS R., RUSINOWITCH M., TURUANI M., VIGNERON L., « Extending the Dolev-Yao Intruder for Analyzing an Unbounded Number of Sessions », *Proc. of the 17th International Workshop in Computer Science Logic (CSL'03)*, 2003.

[CLA 97] CLARK J., JACOB J., « A Survey of Authentication Protocol Literature », `http ://www.cs.york.ac.uk/~jac/ papers/drareviewps.ps`, 1997.

[COM 01] COMON H., CORTIER V., « Tree automata with one memory, set constraints and cryptographic protocols », *Research Report LSV-01-13*, december 2001.

[COM 03a] COMON-LUNDH H., CORTIER V., « New decidability results for fragments of first-order logic and application to cryptographic protocols », *Proc. of the 14th Int. Conf. on Rewriting Techniques and Applications (RTA'2003)*, vol. 2706 de *LNCS*, Springer-Verlag, June 2003, p. 148-164.

[COM 03b] COMON-LUNDH H., CORTIER V., « Security properties : two agents are sufficient », *Proc. of the 12th European Symposium On Programming (ESOP'03)*, vol. 2618 de *LNCS*, Springer Verlag, April 2003, p. 99-113.

[COM 03c] COMON-LUNDH H., SHMATIKOV V., « Intruder Deductions, Constraint Solving and Insecurity Decision in Presence of Exclusive or », *Proc. of 18th Annual IEEE Symposium on Logic in Computer Science (LICS '03)*, 2003, p. 271-280.

[COM 04] COMON-LUNDH H., CORTIER V., « Tree automata with one memory, set constraints and cryptographic protocols », *Theoretical Computer Science, to appear*, 2004.

[COR 01] CORTIER V., MILLEN J., RUESS H., « Proving Secrecy is easy enough », *Proc. of the 14th Computer Security Foundations Workshop (CSFW'01)*, IEEE Computer Society Press, 2001, p. 97-108.

[COR 02] CORTIER V., « Outil de vérification SECURIFY », Rapport numéro 7 du projet RNTL EVA, mai 2002, 6 pages.

[DAV 83] DAVIS M. D., WEYUKER E. J., « *Computability, complexity and languages* », chapitre 7, p. 128-132, Computer Science and Applied Mathematics, Academic Press, 1983.

[DEL 03] DELAUNE S., « Intruder deduction problem in presence of guessing attacks », *In Proc. Workshop on Security Protocols (SPV 2003)*, Marseille, France, September 2003.

[DOL 81] DOLEV D., YAO A., « On the Security of Public Key Protocols », *Proc. of the 22nd Symp. on Foundations of ComputerScience*, IEEE Computer Society Press, 1981, p. 350-357.

[DUR 99] DURGIN N., LINCOLN P., MITCHELL J., SCEDROV A., « Undecidability of bounded security protocols », *Proc. of the Workshop on Formal Methods and Security Protocols*, 1999.

[EVE 83] EVEN S., GOLDREICH O., « On the security of multi-party ping-pong protocols », *Technical Report*, IEEE Computer Society Press, 1983.

[EVE 86] EVEN S., GOLDREICH O., SHAMIR A., « On the security of ping-pong protocols when implemented using the RSA », *Advances in cryptology—CRYPTO 85*, vol. 218 de *LNCS*, Santa Barbara, California, United States, 1986, p. 58–72.

[FIO 01] FIORE M., ABADI M., « Computing Symbolic Models for Verifying Cryptographic Protocols », *Proc. of the 14th Computer Seurity Foundation Workshop (CSFW'01)*, IEEE Computer Society Press, 2001, p. 160-173.

[GAR 99] GARAY J. A., JAKOBSSON M., MACKENZIE P., « Abuse-Free Optimistic Contract Signing », *Advances in Cryptology : Proc. of Crypto'99*, vol. 1666 de *LNCS*, Springer Verlag, 1999, p. 449-466.

[GEN 03] GENET T., TANG-TALPIN Y.-M., TONG V. V. T., « Verification of copy-protection cryptographic protocol using approximations of term rewriting systems », *Proc. of the Workshop on Issues in the Theory of Security (WITS'03)*, 2003.

[GON 93] GONG L., LOMAS T. M. A., NEEDHAM R. M., SALTER J. H., « Protecting poorly choosen secrets from guessing attacks », *IEEE journal on Selected Areas in Communications*, vol. 11, n° 5, 1993, p. 648-656.

[GOR 01] GORDON A., JEFFREY A., « Authenticity by Typing for Security Protocols », rapport n° MSR-2001-49, 2001, Microsoft Research, Conference version appeared in Proc. IEEE Computer Security Foundations Workshop, 2001.

[HEA 00] HEATHER J., LOWE G., SCHNEIDER S., « How to Prevent Type Flaw Attacks on Security Protocols », *Proc. of the 13th Computer Security Foundations Workshop (CSFW'00)*, IEEE Computer Society Press, 2000.

[HER 94] HERZBERG A., KRAWCZYK H., TSUDIK G., « On Travelling Incognito », *Proc. of the IEEE Workshop on Mobile Computing Systems and Applications*, 1994.

[HUT 02] HUTTEL H., « Deciding Framed Bisimulation », *4th International Workshop on Verification of Infinite State Systems INFINITY'02*, 2002, p. 1-20.

[JAC 00] JACQUEMARD F., RUSINOWITCH M., VIGNERON L., « Compiling and Verifying Security Protocols », *Logic for Programming and Automated Reasoning (LPAR'00)*, vol. 1955 de *LNCS*, November 2000.

[KRE 01] KREMER S., RASKIN J.-F., « A Game-Based Verification of Non-Repudiation and Fair Exchange Protocols », LARSEN K., NIELSEN M., Eds., *Concurrency Theory - Concur 2001*, vol. 2154 de *LNCS*, Springer Verlag, août 2001, p. 551-565.

[KRE 02] KREMER S., RASKIN J.-F., « Game Analysis of Abuse-free Contract Signing », SCHNEIDER S., Ed., *Proc. of the 15th Computer Security Foundations Workshop (CSFW'02)*, IEEE Computer Society Press, juin 2002, p. 206-220.

[LOW 96] LOWE G., « Breaking and fixing the Needham-Schroeder public-key protocol using FDR », MARGARIA T., STEFFEN B., Eds., *Tools and Algorithms for the Construction and Analysis of Systems (TACAS'96)*, vol. 1055, Springer-Verlag, Berlin Germany, march 1996, p. 147–166, Also in Software Concepts and Tools, 17 :93-102, 1996.

[LOW 97a] LOWE G., « Casper : A Compiler for the Analysis of Security Protocols », *Proc. of 10th Computer Security Foundations Workshop (CSFW'97)*, IEEE Computer Society Press, 1997, Also in Journal of Computer Security, Volume 6, pages 53-84, 1998.

[LOW 97b] LOWE G., « A Hierarchy of Authentication Specifications », *Proc. of the 10th Computer Security Foundations Workshop (CSFW'97)*, IEEE Computer Society Press, 1997.

[LOW 98] LOWE G., « Towards a Completeness Result for Model Checking of Security Protocols », *Proc. of the 11th Computer Security Foundations Workshop (CSFW'98)*, IEEE Computer Society Press, 1998.

[LOW 02] LOWE G., « Analysing Protocols Subject to Guessing Attacks », *Proc. of the Workshop on Issues in the Theory of Security (WITS '02)*, 2002.

[MEN 97] MENEZES A. J., VAN OORSCHOT P. C., VANSTONE S. A., *Handbook of applied cryptography*, CRC Press, 1997.

[MIL 00] MILLEN J., RUESS H., « Protocol-Independent Secrecy », *Proc. of the 21st Symposium on Research in Security and Privacy (RSP'00)*, IEEE Computer Society Press, 2000.

[MIL 01] MILLEN J., SHMATIKOV V., « Constraint Solving for Bounded-Process Cryptographic Protocol Analysis », *Proc. of the 8th ACM Conference on Computer and Communications Security (CCS'01)*, 2001.

[MIL 03] MILLEN J., SHMATIKOV V., « Symbolic Protocol Analysis with Products and Diffie-Hellman Exponentiation », *Proc. of the 16th IEE Computer Seurity Foundation Workshop (CSFW'03)*, 2003.

[MIT 97] MITCHELL J., MITCHELL M., STERN U., « Automated analysis of cryptographic protocols using Murphi », *Proc. of the IEEE Symposium on Security and Privacy*, IEEE Computer Society Press, May 1997, p. 141–151.

[NEE 78] NEEDHAM R., SCHROEDER M., « Using Encryption for Authentication in Large Networks of Computers », *Communication of the ACM*, vol. 21, n° 12, 1978, p. 993-999.

[PAU 97] PAULSON L., « Mechanized Proofs for a Recursive Authentication Protocol », *Proc. of the 10th Computer Security Foundations Workshop*, IEEE Computer Society Press, 1997, p. 84-95.

[PAU 98] PAULSON L., « The inductive approach to verifying cryptographic protocols », *Journal of Computer Security*, vol. 6, n° 1, 1998, p. 85-128.

[PFI 00] PFITZMANN B., SCHUNTER M., WAIDNER M., « Cryptographic Security of Reactive Systems », *Electronic Notes in Theoretical Computer Science*, vol. 32, April 2000.

[RAM 03] RAMANUJAM R., S.P.SURESH, « Tagging makes secrecy decidable for unbounded nonces as well », *Proc. of the 23rd Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'03)*, Mumbai, 2003.

[RIV 78] RIVEST R., SHAMIR A., ADLEMAN L., « A Method for Obtaining Digital Signatures and Public Key Cryptosystems », *Communications of the ACM*, vol. 21, ACM, February 1978, p. 120-126.

[RUS 01] RUSINOWITCH M., TURUANI M., « Protocol insecurity with finite number of sessions is NP-complete », *Proc. of the 14th Computer Security Foundations Workshop (CSFW'01)*, IEEE Computer Society Press, 2001, p. 174-190.

[RYA 00] RYAN P., SCHNEIDER S., GOLDSMITH M., LOWE G., ROSCOE A., *The modelling and analysis of security protocols : the CSP approach*, Addison-Wesley, 2000.

[SAM 95] SAMFAT D., MOLVA R., ASOKAN N., « Untraceability in Mobile Networks », *Mobile Computing and Networking*, 1995, p. 26-36.

[SCH 96a] SCHNEIDER S., « Security Properties and CSP », *Proc. of the Symposium on Security and Privacy*, IEEE Computer Society Press, 1996, p. 174–187.

[SCH 96b] SCHNEIER B., *Applied Cryptography Second Edition : protocols, algorithms, and source code in C*, J. Wiley & Sons, Inc., 1996.

[SCH 97] SCHNEIDER S., « Verifying Authentication Protocols with CSP », *Proc. of the 10th Computer Security Foundations Workshop (CSFW'97)*, IEEE Computer Society Press, 1997.

[SHM 02a] SHMATIKOV V., HUGHES D., « Defining Anonymity and Privacy (extended abstract) », *Proc. of the Workshop on Issues in the Theory of Security (WITS '02)*, 2002.

[SHM 02b] SHMATIKOV V., MITCHELL J., « Finite-state Analysis of Two Contract Signing Protocols », *Special issue of Theoretical Computer Science on security*, , 2002, p. 419-450.

[SON 99] SONG D. X., « Athena : A New Efficient Automatic Checker for Security Protocol Analysis », *Proc. of the 12th Computer Security Foundations Workshop (CSFW'99)*, IEEE Computer Society Press, June 1999.

[STE 95] STERN U., DILL D., « Automatic verification of the SCI cache coherence protocol », *Correct Hardware Design and Verification Methods : IFIP WG10.5 Advanced Research Working Conference Proc.*, 1995, p. 21-34.

[THA 99] THAYER J., HERZOG J., GUTTMAN J., « Strand spaces : proving security protocols correct », *IEEE Journal of Computer Security*, vol. 7, 1999, p. 191-230.

Article reçu le 1.
Version révisée le 2.
Rédacteur responsable : 3

Véronique Cortier *est chargée de recherche dans l'équipe Cassis du Loria à Nancy. Ses activités de recherche concernent la déduction automatique et l'étude des protocoles cryptographiques sous plusieurs formes : recherche d'algorithmes de décision, affaiblissement du chiffrement parfait, spécification de propriétés, ...*

Annexe pour le service de fabrication

Article pour les actes :
Technique et science informatiques

Auteurs :
Véronique Cortier

Titre de l'article :
Vérifier les protocoles cryptographiques

Titre abrégé :
Vérifier les protocoles cryptographiques

Traduction du titre :
Verifying cryptographic protocols

Date de cette version :
17 juin 2004

Coordonnées des auteurs :

– téléphone : 03 83 59 30 55
– télécopie : 03 83 59 83 19
– Email : cortier@loria.fr

Logiciel utilisé pour la préparation de cet article :
LaTeX, avec le fichier de style `article-hermes.cls`,
version 1.10 du 17/09/2001.

Formulaire de copyright :
Joindre le formulaire de copyright signé, récupéré sur le web à l'adresse
`http://www.hermes-science.com`