METHODES FORMELLES ET PROTOCOLES CRYPTOGRAPHIQUES

Stage de deuxième année de magistère d'informatique, réalisé au laboratoire d'informatique de SRI International, Menlo Park, Californie, Etats-Unis par David Monniaux, été 1997

Plan

Résumé

Les réseaux informatiques utilisent un grand nombre de protocoles, dont certains (par exemple Kerberos) à sécurité garantie par de la cryptographie. Afin de s'assurer de leur fiabilité, on aimerait prouver celle-ci formellement dans un certain modèle simplifié. En particulier, on ne s'intéressera pas aux problèmes de fiabilité des primitives cryptographiques impliquées.

Une approche peu coûteuse est celle des logiques modales de croyances adaptées à ces analyses ; elles permettent d'éliminer certains mauvaises conceptions flagrantes. Je propose une méthode systématique pour l'automatisation de ce genre de logique, que j'applique à deux logiques particulièrement connues (BAN, GNY).

La démonstration de propriétés de sécurité dans un modèle plus ambitieux est largement plus pénible. Afin d'éviter de demande un effort humain trop important, on a essayé d'utiliser le plus possible les ressources du calcul par machine ; cependant, les connaissances sur la méthodologie à appliquer sont encore faibles. Je donne une abstraction mécanisable du problème.

Partie 1

Déroulement du stage

1.1 Présentation du laboratoire

1.1.1 Environnement

La compagnie SRI International [1], connue auparavant sous le nom de *Stanford Research Institute*, est une entreprise a but non lucratif, dont les activités sont le developpement d'applications scientifiques et le conseil à des organismes privés ou gouvernementaux.

Le laboratoire d'informatique [2] (*computer science laboratory*) fait de la recherche théorique et appliquée dans les domaines suivants :

- méthodes formelles et preuve automatisée de théorèmes,
- logique linéaire et théorie de la preuve,
- sécurité des systèmes informatiques,
- communications a haut débit,
- émulation de circuits intégrés.

1.1.2 Méthodes formelles

Dès les années 70, SRI a développé une série de prouveurs de théorèmes : Boyer–Moore, EHDM, et maintenant PVS. PVS [3] est un système comprenant

- un langage de définition, permettant d'exprimer, dans un style purement fonctionnel typé les fonctions étudiées, ainsi que les axiomes supposés et les lemmes et théorèmes que l'on veut établir ;
- un prouveur interactif, basé sur la logique classique (séquents), automatisant partiellement les tâches par des tactiques, et connaissant les propriétés de base des types usuels, comme les entiers, afin de les appliquer dans des procédures de décision.

Ce système sert à l'analyse de propriétés de circuits ou de programmes. Il peut être couplé à des *model-checkers* pour certaines applications.

Les activités de l'équipe se dirigent donc dans les directions de :

- développements en théorie de la preuve,
- développement du logiciel PVS et d'accessoires,
- application de méthodes formelles a différents problèmes, en particulier à l'aide du logiciel PVS.

1.2 Remerciements

Merci à John Rushby pour l'organisation du stage et l'aide sur PVS, Natarajan Shankar pour ses précieux conseils toujours pertinents, Sam Owre pour sa patience dans l'aide aux pauvres débutants, Pat Lincoln pour ses conseils, Li Gong pour ses explications complémentaires, Judith Burgess pour la solution des problèmes administratifs et matériels et le logement, Sergey Berezin et Drew Dean pour le soutien moral, et tous ceux qui ont contribué à

1. Voir `http://www.sri.com`.
2. Voir `http://www.csl.sri.com`.
3. Divers articles sur PVS, ainsi que les manuels du système, sont disponibles à `http://www.csl.sri.com/pvs.html`.

GNU Emacs et ses modules additionnels, TEX et les programmes qui gravitent autour, LATEX [4] et ses *packages*, et Objective CAML.

4. Pour la petite histoire, j'ai rencontré Leslie Lamport à l'université de Stanford !

Partie 2

Problèmes scientifiques

2.1 Cryptographie et communications

Nous nous intéressons a des systèmes de communication ou d'authentification réputés sûrs ; par exemple, des réseaux de telephonie mobile, de paiement par carte bancaire, de micro-ordinateurs dans un réseau local. Les critères de construction de ces systèmes sont de deux catégories :

1° le système doit fournir des performances (vitesse de transmission, temps de réponse) acceptables, tout en restant d'un coût raisonnable;

2° le système doit être sûr.

On étudie le plus souvent le second point en séparant la sécurité des primitives cryptographiques (fonctions de chiffrement, de hachage...) et la sécurité des protocoles qui les emploient. Tandis qu'on a porté une attention considérable sur la sécurité des primitives cryptographiques [1] des protocoles publiés et utilisés présentent des fautes graves de sécurité.

Les méthodes formelles d'analyse permettent d'attaquer le problème de la preuve de sécurité suivant différents degrés de coût et de performance.

2.1.1 Logiques modales

Au bas de l'échelle des coûts, nous trouvons certaines logiques modales permettant de débusquer quelques erreurs assez facilement ; ces logiques sont essentiellement utiles lorsqu'elles donnent des résultats négatifs, car cela montre que l'on a négligé d'introduire certaines présuppositions sur le système étudié. Par exemple, la littérature ([BAN89], [GNY90]) montre comment une analyse grâce à ces logiques peut montrer que certains protocoles ne fonctionnent que sous des hypothèses fortes (par exemple, qu'une vieille communication sur le même réseau n'a pas été percée).

Les faiblesses de ces logiques sont :

- elles s'appuient sur une idéalisation parfois contestable du protocole ;
- elles ne s'intéressent qu'à des problèmes « sémantiques » du protocole.

D'un autre côté, ces logiques sont facilement implémentables. *L'annexe A contient mes résultats sur des procédures de décision effectives pour certaines de ces logiques, dérivées d'une même méthode de transformation en un système de chaînage avant à théorie finie ; en particulier, je propose pour la première fois une procédure de décision de la logique GNY.*

2.1.2 Preuve de théorèmes

Ici, nous nous intéressons à un modèle formel (*i.e.* mathématique) de ce qu'est un protocole et un intrus. Différents modèles peuvent être considérés. [Pau97] considère un modèle où l'intrus contrôle le réseau et un nombre quelconque de principaux communiquent en utilisant le protocole (exécutions entrelacées ou *interleaved runs*). Ce modèle est très fort ; en effet, il suppose que l'intrus peut

- écouter tous les messages communiqués,
- intercepter certains messages et empêcher leur retransmission, ou les retransmettre plus tard,

1. [Sch96], contient une description détaillée des primitives les plus courantes et des références sur leur sécurité.

– émettre tout message qu'il peut constituer au vu des données qu'il a en sa possession.

Je propose dans B.1.1 un modèle général sous forme d'un système de transitions.

Les conditions exigées à l'issue du protocole sont diverses :

– certaines données confidentielles ne doivent pas entrer en possession de l'intrus,

– les principaux doivent être en possession des données correctes.

Le théorème à prouver est alors très complexe. [Pau97] prouve des propriétés au prix d'un travail humain important et de stratégies avancées de preuve semi-automatique. [Bol97] propose plutôt de ramener le problème a un cas fini par abstraction.

Des modèles plus faibles sont proposés ; par exemple, [Bol97] considère une seule exécution du protocole en parallèle.

2.2 Moyens de preuve employés

2.2.1 Difficultés de la preuve de theorèmes, en particulier dans le cas de PVS

Le problème de décider si une formule est démontrable ou non est indécidable dans les logiques suffisamment puissantes pour exprimer ce à quoi nous nous intéressons. En conséquence, les prouveurs semi-automatisés de théorèmes [2] sont des systèmes interactifs, où l'utilisateur pilote le programme. Les différences principales, du point de vue de l'utilisateur voulant vérifier des problèmes réels, sont surtout dans le degré d'automatisation, ou, pour parler plus pratiquement, la grosseur de l'ensemble des théorèmes que le système prouve de lui-même. La difficulté réside dans le fait que, tandis que nous basons les raisonnements mathématiques sur des définitions rigoureuses, notre pensée est guidée par l'aspect intuitif des concepts rencontrés ; par exemple, nous raisonnons intuitivement sur les entiers naturels, sans nous ramener à une definition par un type inductif avec une constante « zéro » et un constructeur unaire « successeur ». Le problème est que la machine raisonne comme cela.

Le système PVS est assez performant sur des exemples intéressants en informatique par son utilisation de procédures de décision sur les types couramment rencontrés (entiers...). Un problème est que dès que l'on sort de ces types, même si l'on reste dans des types assez intuitifs pour nous comme les permutations d'un ensemble fini, par exemple, le système devient très maladroit.

La difficulté principale vient du problème d'instancier des variables dont on ne peut connaître *a priori* la valeur. Par exemple, l'axiome $\forall x, A(x) \Rightarrow B$, indique que pour démontrer B, on peut démontrer $A(x)$ pour un certain x. Mais quel x ? PVS a quelques heuristiques à ce sujet, mais cela reste un problème ennuyeux y compris dans des cas où l'instanciation est intuitive pour un humain.

2.2.2 Chaînage avant

Étant donné la simplicité des logiques d'analyse de protocole, j'ai cherché à nous affranchir de ces contraintes en construisant des systèmes de chaînage avant (voir annexe A) dont la théorie engendrée est finie et facilement énumérable.

2.2.3 Abstraction

Pour le cas plus complexe du modèle de protocoles entrelacés (annexe B), j'essaye de trouver une abstraction finie du problème. Le théorème abstrait est alors prouvé — s'il est vrai, bien entendu — par *model-checking*. L'abstraction, bien choisie, fait que cela implique le théorème concret.

L'abstraction dans un système à transitions d'états [3] est une technique par laquelle on regroupe les états concrets sur des états abstraits. On peut considérer une fonction des états concrets vers les états abstraits (par exemple, envoyer $\mathbb{Z}_-$ sur un état abstrait Z-*, 0 sur 0 et $\mathbb{Z}_+$ sur Z+*. Plus généralement, on considère une relation entre les états concrets et les états abstraits. Cela est particulièrement intéressant avec les états abstraits organisés en réseau ; ainsi, on pourrait envoyer 0 sur à la fois

2. Citons, outre PVS, Coq [BBC+97], Isabelle, HOL...

3. Patrick Cousot notamment a développé l'analyse de programmes par interprétation abstraite. Voir [Cou97] pour une introduction.

0, Z-, Z+. Z- est alors la borne supérieure de 0 et Z-*. C'est ce que nous faisons dans B.2 avec un treillis de clôtures.

Je propose une abstraction finie du problème de n protocoles (définis de façon raisonnable) lancés avec entrelacement des transmissions. Le problème de cette abstraction est qu'elle peut être trop grossière pour certains protocoles ; de plus, il reste le problème de la démonstration pour n quelconque.

2.2.4 *Model-checking*

Lorsque le domaine sémantique dans lequel nous analysons le système est fini (après abstraction, si besoin est), et d'un cardinal raisonnable, nous pouvons utiliser des techniques de *model-checking*, qui cherchent à vérifier les formules par exploration systématique de l'espace d'état. Évidemment, ces méthodes nécessitent des optimisations avancées afin de rester en un temps acceptable y compris sur des domaines très grands.

[MCJ97] illustre l'utilisation du *model-checker* SMV pour l'analyse de protocoles cryptographiques. Cependant, il semble que le problème du non-déterminisme infini de l'intrus (cf annexe B) a été négligé dans cet article.

Annexe A

Texte du rapport technique sur la décision des logiques BAN et GNY

A.1 Introduction

A.1.1 The issues

Most real-life networks are bound to be inherently insecure. The communication media might be wiretapped; some of the communicating machines might get compromised, and the data they hold fall into the hand of intruders. We see here two essential tasks: users' data transmitted at all time mustn't be decipherable, and compromission of part of the network architecture or machines at an instant in time shouldn't yield risks later or for other machines.

Cryptographic protocols address those needs. Those protocols are built from cryptographic primitives (symmetric encryption schemes such as DES or RC4, public-key schemes such as RSA, hash functions such as MD5; see [Sch96] for general information on those topics) the security of which can be attacked by cryptanalysis. The methods used here assume the security of the primitives; we shall only deal with attacks on the protocol itself rather than the primitives involved.

Many other concerns interfere with the design of cryptographic communication protocols. Ease of networking in real environments containing hundreds of machines is to be taken care of; failures in machines must be dealt with; operating systems must be taken into account ... and efficiency must be met. Performance concerns deter the protocol designer from using too many computationnally expensive encryption steps[1]. For those reasons, many protocols are proposed. Claims are made about them, such as "at the end of this round, this machine knows that it must use that key to communicate with that other machine".

Many approaches to the problem of analysing those protocols, finding bugs in them or proving some safety properties on them have been proposed. Some of them, as proposed in [Pau97, Bol97], tackle the very general task of ensuring that in no circumstance critical bits of information can leak to an intruder, going as far as considering multiple interleaved runs[2]. Simpler formalisms include modal logics of belief, as outlined in [BAN89] and improved in [GNY90]. (Partially) automated systems for protocol analysis have been proposed [Bra95, Bra96, Pau97, CS96, Sch97, KW96]. Those logics analysis the protocol in an abstract model, on properties such as "after this step, the machine A 'knows' that the key K_{ab} is a secret key to be used between it and B"; reasoning steps are such as "if I get a message encrypted with a secret key only myself and A know, then that messages must be originating from myself or A". We propose here a general method to decide several of those logics.

[1] Some cryptographic primitives, such as RSA encryption (see [Sch96] for details) involve especially long computations such as taking powers in $\mathbb{Z}/n\mathbb{Z}$. In that case, it is really important that the number of encryptions used be minimal, if the protocol is to be used with a high frequency or on a slow computation unit such as a smart card.

[2] A **run** of the protocol is an instance of the use of the protocol. For instance, if the protocol is a safe mail transmission protocol, each sending of a mail constitutes a run of the protocol.

A.2 Knowledge sets

A.2.1 The problem

An idealization of reality

In the general problem of analysis of cryptographic protocols, one has to define the potential knowledge of the intruder. An approach that sticks to reality would be to define encrypted data type as they are in reality (eg as, for instance, 64-bit data). The problem with that approach is that in that a naive formalization of it, the intruder always wins. Those types being finite, the intruder "possesses" all the elements, and analysis will fail for the reason that the intruder can simply guess the correct value. Correct formalizations would include a notion of probability; for instance, while it is true that an intruder may guess a 64-bit value, the probability of that, 2^{-64}, is so small that it is close to zero. Alas, such probabilistic analysis tends to be quite complex.

We therefore use an idealized model. Pieces of data are considered as terms over an algebra of constructors, maybe quotiented by some simplifications (such as encryption then decryption with the same key yields the original piece of data). The universe of data that an intruder possesses is the closure of what it receives by some rules. The intruder is not allowed to guess anything; it can only construct what it can from the pieces of data it received.

An objection to that model is that the intruder is still allowed to "guess" which piece of data is to be sent in the whole universe it can compute. For instance, if the intruder possesses X, and the secret is X concatenated with itself n times, in which n is a secret integer, then our model says that the intruder possesses the secret. That supposes he can somehow guess n. This is not a real problem in the case of cryptographic protocols defined in the usual ways: all the data is represented as "small" terms, and no data is "hidden" in such ways that circumvent the model.

Formal definitions

We want an algorithm that, given a (finite) initial set I of terms, taken on an algebra the signature of which contains:

- pairing $(\bullet, \bullet)$
- encryption $\{\bullet\}_\bullet$

and a term t, computes whether t is in the closure I by the following rules — which we note $I \vdash t$:

- construction rules
  - pairing $\dfrac{X \quad Y}{(X,Y)}$,
  - encryption $\dfrac{X \quad K}{\{X\}_K}$.
- destruction rules
  - splitting $\dfrac{(X,Y)}{X \quad Y}$,
  - decryption $\dfrac{\{X\}_K \quad K}{X}$,

We here consider symmetric[3] encryption; public-key[4] encryption could be considered by extending the approach.

An easy approach is described in [Bol97]. It consists in computing the closure I' of I by the "decomposition" operations (splitting and decryption), which is finite. Then, determining whether

[3] An encryption scheme is said to be secret-key or symmetric if the same key can be used for encryption and decryption. Such systems include DES, RC4 etc ... , see [Sch96]. The main problem with such systems is that they need at least some secure initial transmission of data to take place apart from them: one has to get the key to both ends of the transmission.

[4] An encryption scheme is said to be public-key or asymmetric if keys are organized in couples (p, s). Messages encrypted with p can only be decrypted with s. An user possessing s, the private key, and keeping it secret may publish p, the public key; other people may then send messages to him, encrypted with p, knowing that they can't be decrypted by any other person. The RSA cryptosystem (see [Sch96]) also has the interesting property that the public key decrypts messages encrypted with the private key, but only the private key allows encrypting meaningful messages. Thus the private key may be used as an authentication secret: messages meaningfully[5] encrypted with it, which can be checked by anybody possessing the (published) public key, may then believe they were sent by the owner of the private key.

[5] We mean by meaningfully encrypted that, though an intruder may output any kind of data and pretend it is the encryption of something, the format of the expected data has enough redudancy that it excludes random data that would be the result of the decryption of such.

a term belongs in C by backward-chaining[6] on the "composition" operations (pairing and encryption). That method is based on the following theorem:

$$I \vdash t \iff I \vdash_d I' \vdash_c C$$

where $\vdash$ denotes derivation by all the rules, $\vdash_d$ decomposition and $\vdash_c$ composition (see [Bol97] for a proof — it is basically a cut-free, or normalization, theorem). Sadly, this theorem only holds in the framework of [Bol97], where keys can't have a structure (that is, can't be composite). For instance, if we derive

$$\frac{\{X\}_{(K,K')} \quad \dfrac{K \quad K'}{(K,K')}}{X},$$

we must do one composition step (pairing) before one decomposition step (decryption).

A.2.2 Our approach

We now propose a decision procedure for the more general case where we allow keys to be composite.The idea is that if we reach some data encrypted with some key, we try to build up that key. That form of search can of course be achieved by special tactics; for the sake of ease and clarity of proofs, and implementation in general-purpose forward-chaining[7] systems, we prefer to implement that in the logic itself. We add another symbol, *goal*. $\boxdot X$ means qualitatively that "we would like to prove X". We define $\vdash'$ by the following rules:

- $\dfrac{X \quad Y \quad \boxdot(X,Y)}{(X,Y)}$,
- $\dfrac{(X,Y)}{X \quad Y}$,
- $\dfrac{\{X\}_K \quad K}{X}$,
- $\dfrac{X \quad K \quad \boxdot\{X\}_K}{\{X\}_K}$,
- $\dfrac{\boxdot(X,Y)}{\boxdot X \quad \boxdot Y}$,
- $\dfrac{\{X\}_K}{\boxdot K}$
- $\dfrac{\boxdot\{X\}_K}{\boxdot X \quad \boxdot K}$

We then state

Theorem 1 *For all set Γ of hypotheses and formula t (on the original algebra),*

$$\Gamma \vdash t \iff \Gamma, \boxdot t \vdash' t.$$

The proof is given more generally in the next section.

Here is an example: $K, K', \{X\}_{(K,K')} \vdash X$ is reached by

$$\frac{\{X\}_{(K,K')} \quad \dfrac{\dfrac{\boxdot\{X\}_{(K,K')}}{\boxdot(K,K')} \quad K \quad K'}{(K,K')}}{X}.$$

Theorem 2 *Forward-chaining[8] the rules of $\vdash'$ is a decision procedure.*

It's a consequence of the preceding theorem and the following lemma:

Property 3 *For all rule*

$$\frac{\mathcal{H}_1 \cdots \mathcal{H}_n}{\mathcal{C}}$$

of $\vdash'$,

$$\mathcal{M}(\mathcal{C}) \leq \max\{\mathcal{M}(\mathcal{H}_1), \ldots, \mathcal{M}(\mathcal{H}_n)\} - 1.$$

The measure $\mathcal{M}$ is defined inductively in tab. A.1.

The theorem ensures the correctness and completeness of the method, and the lemma ensures the termination.

[6] We call **backward-chaining** rules rules of the form

$$\frac{\mathcal{H}_1 \ldots \mathcal{H}_n}{\mathcal{C}}$$

in which all the variables in the hypotheses are found in the conclusion. In a backward-chaining system, when one wants to check whether a formula is provable, he tries to match that formula against all such conclusions; if a match succeeds, he tries to prove the hypotheses of the rules he used.

[7] We call **forward-chaining** rules rules of the form

$$\frac{\mathcal{H}_1 \ldots \mathcal{H}_n}{\mathcal{C}}$$

in which all the variables in the conclusion $\mathcal{C}$ are found in the hypotheses $\mathcal{H}_1 \ldots \mathcal{H}_n$. Such systems allow the computation of the whole inferrable theory by repeatedly attempting to unify the hypotheses of a rule with some already proved formulas, and if possible adding its conclusion with the correct variable instanciations.

[8] That is, generating the whole theory by repeatedly trying to apply each rule, till nothing new is produced, then checking whether the desired formula is in that theory.

t	$\mathcal{M}(t)$
α	1
(X,Y)	$\max\{\mathcal{M}(X),\mathcal{M}(Y)\}+1$
$\{X\}_K$	$\max\{\mathcal{M}(X),\mathcal{M}(K)\}+1$
$\boxdot X$	$\mathcal{M}(X)+1$

Table A.1: The measure $\mathcal{M}$ for knowledge set derivations.

A.3 Generalization

We try to generalize that method to more complex systems of rules. We still use the same idea that all destruction may take place before all construction. The only problem with that and composite keys is that these keys may need to be constructed. We trigger the construction of only the keys we need to achieve the destruction.

More generally, we deal with two categories of rules[9]:

- **construction rules**, in which all the variables of the hypothesis are found in the conclusion (those rules are automatically backward-chainable);
- **destruction rules**, in which all the variables of the conclusion found in the hypothesis (those rules are automatically forward-chainable); for those rules, we distinguish the data hypotheses (which can be one or more) and the optional key hypotheses; the variables in the key hypotheses are a subset of those in the data hypotheses.

A constraint for our method to work is that it must always be possible to find a *normal derivation*; that is, for all set of hypotheses Γ and formula t so that $\Gamma \vdash t$, there must be a derivation so that there's no construction rule to be used as the root rule of the sub-derivation for a data hypothesis of a destruction rule.[10]

Informally, that means that you must be able to derive anything that is derivable without having to construct something and destruct it afterwards; for instance, in

$$\frac{\dfrac{(X,Y)}{X}\quad\dfrac{(X,Y)}{Y}}{(X,Y)}$$

we construct a pair just to destruct it afterwards.

Our transformation turns the construction rule

$$\frac{\mathcal{H}_1 \cdots \mathcal{H}_n}{\mathcal{C}}$$

into a pair

$$\frac{\boxdot\mathcal{C}\quad \mathcal{H}_1 \cdots \mathcal{H}_n}{\mathcal{C}}$$

and

$$\frac{\boxdot\mathcal{C}}{\boxdot\mathcal{H}_1 \cdots \boxdot \mathcal{H}_n}$$

and adds to the destruction rule

$$\frac{\mathcal{D}_1 \cdots \mathcal{D}_m \quad \mathcal{K}_1 \cdots \mathcal{K}_n}{\mathcal{C}},$$

where the one or more $\mathcal{D}_i$ are data hypotheses and the zero or more $\mathcal{H}_i$ are key hypotheses, the triggering rule

$$\frac{\mathcal{D}_1 \cdots \mathcal{D}_m}{\boxdot\mathcal{K}_1 \cdots \boxdot \mathcal{K}_n}.$$

See A.2.2 for examples.

Theorem 4 *Such a system verifies for all set of hypotheses* Γ *and formula* t,

$$\Gamma \vdash t \iff \Gamma, \boxdot t \vdash' t;$$

all t *such a* $\Gamma \vdash t$ *can be reached by forward-chaining*[11] *on* $\vdash'$.

The second claim is evident, given the structure of the rules. The right-to-left way of the first claim comes from the rules whose conclusions are non-goal formulas are rules from the original system,

[9] This partition of the set of rules into two such categories is very similar to that of [KW96], where they are called respectively *growing* and *shrinking rules*. We developped our framework independently of the cited work, of which we became aware shortly afterwards.

[10] A stronger constraint is that when a construction rule is used to produce a data hypothesis of a destruction rule, both rules can be removed (i.e. it is never necessary to construct some object to split it up afterwards); doing so until impossible yields a *normal derivation*. This is the case of the knowledge sets and BAN logic. However, this constraint is unecessarily strong, and for instance fails to capture logics where this condition is not true for all construction-destruction pairs, but where the remaining pairs can be transformed into a destruction rule. such as GNY logic (see A.5.2). We call that later condition the **strong normal derivation property**.

[11] Termination of the forward-chaining strategy may be ensured using a step-decreasing measure property, as we do on our examples.

weakened by adding extra hypotheses. The other way is a consequence of the following lemma:

Lemma 5 *If $\Gamma \vdash t$ then $\boxdot t, \Gamma \vdash' t$; we then even have $t \vdash' t$ if there exists a normal derivation $\Gamma \vdash t$ that doesn't end with a construction rule.*

Proof by induction on the structure of a normal derivation of $\Gamma \vdash t$:

- for axioms, it's evident;
- if the bottom rule is a construction rule
$$\frac{\mathcal{H}_1 \cdots \mathcal{H}_n}{t}$$
as we have $\boxdot t$ we can apply the corresponding rule
$$\frac{\boxdot t \quad \mathcal{H}_1 \cdots \mathcal{H}_n}{t}$$
at the bottom of the derivations of $\boxdot \mathcal{H}_i, \Gamma \vdash' \mathcal{H}_i$ of the induction hypothesis; the $\boxdot \mathcal{H}_i$ are derived using
$$\frac{\boxdot \mathcal{C}}{\boxdot \mathcal{H}_1 \cdots \boxdot \mathcal{H}_n};$$
- if the bottom rule is a destruction rule
$$\frac{\mathcal{D}_1 \cdots \mathcal{D}_m \quad \mathcal{K}_1 \cdots \mathcal{K}_n}{t}$$
then for any i the sub-derivation of $\mathcal{D}_i$ is normal and doesn't end with a construction rule (we act on a normal derivation); then we have $\Gamma \vdash' \mathcal{D}_i$; we then use the triggering rule
$$\frac{\mathcal{D}_1 \cdots \mathcal{D}_m}{\boxdot \mathcal{K}_1 \cdots \boxdot \mathcal{K}_n}$$
then the induction hypothesis to get the $\boxdot \mathcal{K}_i, \Gamma \vdash' \mathcal{K}_i$; we apply the original destruction rule at the bottom of those.

A.4 BAN logic

A.4.1 Description

The simple BAN logic, named after its proponents, Burrows, Abadi and Needham (see [BAN89]), though not very powerful, is easy to use. It deals with *beliefs* that *principals* (*i.e.* machines communicating on the network) can hold about the status of the distribution of communication keys.

Formula	Meaning
$P \models X$	P believes X
$P \triangleleft X$	P sees X
$P \mid\sim X$	P once said X
$P \Rightarrow X$	P has jurisdiction over X
$\sharp(X)$	X is fresh
$P \overset{K}{\longleftrightarrow} Q$	K is a symmetric encryption key for P and Q (which commute)
$\overset{+P}{\mapsto} P$	P has K as public key
$P \overset{X}{\longleftrightarrow} Q$	X is a secret shared by by P and Q (which commute)
$\{X\}_K$	X encrypted with K
$\{X\}_{K^{-1}}$	X encrypted with the private key corresponding to the public key K
$\langle X \rangle_Y$	X combined with Y
(X, Y)	pair of X and Y

Table A.2: The meaning of the symbols of BAN logic. P and Q are principals, X and Y formulas, K a key.

Following the works of [CS96], we propose an actual decision procedure for an useful fragment of the logic, and a proof of its adequacy.

The BAN logic (see [BAN89]) rules, on the symbols defined in A.4.1, are the following:

- Message-meaning rules
  - $\dfrac{P \models Q \overset{K}{\longleftrightarrow} P \quad P \triangleleft \{X\}_K}{P \models Q \mid\sim X}$
  - $\dfrac{P \models \overset{+Q}{\mapsto} Q \quad P \triangleleft \{X\}_{K^{-1}}}{P \models Q \mid\sim X}$
  - $\dfrac{P \models Q \overset{Y}{\longleftrightarrow} P \quad P \triangleleft \langle X \rangle_Y}{P \models Q \mid\sim X}$
- Belief production rules
  - $\dfrac{P \models \sharp(X) \quad P \models Q \mid\sim X}{P \models Q \models X}$
  - $\dfrac{P \models Q \Rightarrow X \quad P \models Q \models X}{P \models X}$
- Conjunction of beliefs creating and splitting rules[12]
  - $\dfrac{P \models X \quad P \models Y}{P \models (X, Y)}$

[12] We could extend these rules into a scheme of rules deal-

- $\dfrac{P \models (X,Y)}{P \models X}$ $\qquad$ $\dfrac{P \models (X,Y)}{P \models Y}$
- $\dfrac{P \models Q \models X \quad P \models Q \models Y}{P \models Q \models (X,Y)}$
- $\dfrac{P \models Q \models (X,Y)}{P \models Q \models X}$ $\qquad$ $\dfrac{P \models Q \models (X,Y)}{P \models Q \models Y}$

- Other conjunction splitting rules
  - $\dfrac{P \models Q \mid\sim (X,Y)}{P \models Q \mid\sim X}$ $\qquad$ $\dfrac{P \models Q \mid\sim (X,Y)}{P \models Q \mid\sim Y}$
  - $\dfrac{P \lhd (X,Y)}{P \lhd X}$ $\qquad$ $\dfrac{P \lhd (X,Y)}{P \lhd Y}$

- Message interpretation rules
  - $\dfrac{P \lhd \langle X \rangle_Y}{P \lhd X}$
  - $\dfrac{P \models Q \overset{K}{\longleftrightarrow} P \quad P \lhd \{X\}_K}{P \lhd X}$
  - $\dfrac{P \models \overset{+Q}{\mapsto} Q \quad P \lhd \{X\}_{K^{-1}}}{P \lhd X}$
  - $\dfrac{P \models \overset{+P}{\mapsto} P \quad P \lhd \{X\}_K}{P \lhd X}$

- Freshness extension rule
  $\dfrac{P \models \sharp(X)}{P \models \sharp((X,Y))}$ $\qquad$ $\dfrac{P \models \sharp(Y)}{P \models \sharp((X,Y))}$

A.4.2 Transformation

Apply our technique, we get the following set of rules:

- Message-meaning rules
  - $\dfrac{P \models Q \overset{K}{\longleftrightarrow} P \quad P \lhd \{X\}_K}{P \models Q \mid\sim X}$
  - $\dfrac{P \models \overset{+Q}{\mapsto} Q \quad P \lhd \{X\}_{K^{-1}}}{P \models Q \mid\sim X}$
  - $\dfrac{P \models Q \overset{Y}{\longleftrightarrow} P \quad P \lhd \langle X \rangle_Y}{P \models Q \mid\sim X}$

- Belief production rules
  - $\dfrac{P \models \sharp(X) \quad P \models Q \mid\sim X}{P \models Q \models X}$
  - $\dfrac{P \models Q \mid\sim X}{\boxdot P \models \sharp(X)}$
  - $\dfrac{P \models Q \mapsto X \quad P \models Q \models X}{P \models X}$
  - $\dfrac{P \models Q \mapsto X}{\boxdot P \models Q \models X}$

- Conjunction of beliefs creating and splitting rules
  - $\dfrac{\boxdot P \models (X,Y) \quad P \models X \quad P \models Y}{P \models (X,Y)}$
  - $\dfrac{P \models (X,Y)}{P \models X \quad P \models Y}$
  - $\dfrac{\boxdot P \models (X,Y)}{\boxdot P \models X \quad \boxdot P \models Y}$
  - $\dfrac{\boxdot P \models Q \models (X,Y) \quad P \models Q \models X \quad P \models Q \models Y}{P \models Q \models (X,Y)}$
  - $\dfrac{P \models Q \models (X,Y)}{P \models Q \models X \quad P \models Q \models Y}$
  - $\dfrac{\boxdot P \models Q \models (X,Y)}{\boxdot P \models Q \models X \quad \boxdot P \models Q \models Y}$

- Other conjunction splitting rules
  - $\dfrac{P \models Q \mid\sim (X,Y)}{P \models Q \mid\sim X \quad P \models Q \mid\sim Y}$
  - $\dfrac{P \lhd (X,Y)}{P \lhd X \quad P \lhd Y}$

- Message interpretation rules
  - $\dfrac{P \lhd \langle X \rangle_Y}{P \lhd X}$
  - $\dfrac{P \models Q \overset{K}{\longleftrightarrow} P \quad P \lhd \{X\}_K}{P \lhd X}$
  - $\dfrac{P \models \overset{+Q}{\mapsto} Q \quad P \lhd \{X\}_{K^{-1}}}{P \lhd X}$
  - $\dfrac{P \models \overset{+P}{\mapsto} P \quad P \lhd \{X\}_K}{P \lhd X}$

- Freshness extension rules
  - $\dfrac{\boxdot P \models \sharp((X,Y)) \quad P \models \sharp(X)}{P \models \sharp((X,Y))}$
  - $\dfrac{\boxdot P \models \sharp((X,Y)) \quad P \models \sharp(Y)}{P \models \sharp((X,Y))}$

ing with conjunctions of beliefs through any level of $\bullet \models \bullet$ indirection, as the "rationality" rule of GNY (see [GNY90]) provides.

f	$\mathcal{M}(f)$
$P \models X$	$\mathcal{M}(X)+1$
$P \triangleleft X$	$\mathcal{M}(X)+3$
$P \mid\sim X$	$\mathcal{M}(X)+2$
$P \Rightarrow X$	$\mathcal{M}(X)+3$
$\sharp(X)$	$\mathcal{M}(X)$
$P \overset{K}{\longleftrightarrow} Q$	1
$\overset{+P}{\mapsto} P$	1
$P \overset{X}{\longleftrightarrow} Q$	1
$\{X\}_K$	1
$\{X\}_{K^{-1}}$	$\mathcal{M}(X)+1$
$\langle X \rangle_Y$	$\max\{\mathcal{M}(X),\mathcal{M}(Y)\}+1$
(X,Y)	$\max\{\mathcal{M}(X),\mathcal{M}(Y)\}+1$
$\boxdot X$	$\mathcal{M}(X)+1$

Table A.3: The definition of $\mathcal{M}$ for BAN logic.

$$- \quad \frac{\boxdot P \models \sharp((X,Y))}{\boxdot P \models \sharp(X) \quad \boxdot P \models \sharp(Y)}$$

The same result as for knowledge sets hold, following from the general theorem proved in the precedent section and a well-chosen measure (see tab. A.4.2). Applications of this method on the examples from [BAN89] are given in appendix A.6.

A.4.3 Implementation

We implemented a preprocessor, taking BAN-like code as input and outputting source for LaTeX and for the PVS theorem prover[13]. Then, using a forward-chaining strategy in PVS, we prove automatically all the provable goals. On unprovable ones, the strategy halts with all it could prove, of which the user may get a glimpse of the missing assumption; the user then has to track the error, usually a typing error or too weak assumptions. The "goal" statements forward-chained by the system may be useful in that task, because they are formulae that would be interesting to prove.

For the humor note, our system was actually very good at catching typos in the description of the protocols.

As we noted, we had to instantiate the BAN quantifiers. Most of the time it was obvious (along the lines of $A \overset{K}{\longleftrightarrow} B$ becoming $A \overset{K_{ab}}{\longleftrightarrow} B$). That's why we don't thing the omission of quantifiers is really a problem, according in our experience with the protocols.

[13] Any forward-chaining system will do. We implemented an efficient forward-chaining tactic in PVS for that task.

A.5 GNY logic

A.5.1 Definitions

The GNY logic, named after its authors, Li Gong, Roger Needham and Raphael Yahalom is a logic of belief similar to BAN, but which addresses some deficiencies in that latter one. Again, we won't give it an extensive review and refer the reader to [GNY90] ; see also Li Gong's PhD thesis [Gon90].

The symbols for that logic are summarily described in tab. A.4 and the rules listed in appendix A.7. An additionnal rule is the **rationality rule**: all the rules of the logic may be applied with any preceding chain of belief modality. That is, for all rule

$$\frac{\mathcal{H}_1 \ldots \mathcal{H}_n}{\mathcal{C}}$$

and new principal variable name P, we have a rule

$$\frac{P \models \mathcal{H}_1 \ldots P \models \mathcal{H}_n}{P \models \mathcal{C}}.$$

We consider the closure of the rules by this meta-rule.

The encryptions considered have the following properties:

- $\{\{X\}_K\}_K^{-1} \equiv X$
- $\{\{X\}_{+K}\}_{-K} \equiv X$

A.5.2 Adding some extra rules

The original GNY logic does not match the normal form criterion of A.3 because of the simplifications possible with encryption (see the former paragraph). For instance, we have

$$\frac{\dfrac{P \ni K \quad P \ni \{X\}_K}{P \ni \{\{X\}_K\}_K^{-1}}\ \mathsf{P6}}{P \ni X}\ \text{simplification}$$

That is, we sometimes have to construct the decryption of a piece of data just to apply a (destructive) simplification rule. We add a rule P6'

Symbol	Meaning
Beliefs and conveyance	
$P \models X$	P *believes* X
$P \triangleleft X$	P *is told* X
$P \ni X$	P *possesses* X
$P \mid\sim X$	P *once conveyed* X
Properties of formulae	
$\sharp(X)$	X is *fresh*
$\phi(X)$	X is *recognizable*
Keys and secrets	
$P \overset{S}{\longleftrightarrow} Q$	S is a *secret*[a] shared by P and Q
$\overset{+P}{\mapsto} P$	$+K$ is a *public key* of P's
Message constructors	
$\{X\}_K$	X encrypted with K[b]
$\{X\}_K^{-1}$	X decrypted with K[c]
(X, Y)	the pair of X and Y
Jurisdiction	
$P \Rightarrow X$	P has *jurisdiction* over X
$P \Rightarrow P \models \star$	P is *honest and competent*
$(X \rightsquigarrow C)$	X leading to C
Detection of replays	
$\star X$	X was *not originated here*

[a] Shared secrets are most often shared symmetric keys.

[b] K may here be either a symmetric, a public ($+K$) or a private key ($-K$).

[c] K here must be a symmetric key. Decryption of $\{X\}_{+K}$ is done by encrypting it with $-K$.

Table A.4: The meaning of the symbols of GNY logic, where P and Q range over principals, X and Y over formulas, K over keys; $+X$ and $-X$ are respectively the public and the private keys in a pair.

that stands for the former derivation. We add similar rules in similar other cases, making up the $\vdash'$ derivation system.

Calling $\vdash_{\equiv}$ the original GNY logic modulo the encryption simplifications, $\vdash_+$ our augmented derivation system, and simplified formulae those on which no more left-to-right $\equiv$ rewritings can be made, we have the following:

Property 6 *For all set Γ of simplified hypotheses, all formula t, $\Gamma \vdash_{\equiv} t$ if and only if $\Gamma \vdash_+ t$.*

A.5.3 Transformation into a forward-chaining system

We divide our rules according to the classification laid out in A.3; **equivalence** rules are considered as a special case of destruction rules. We then apply the transformation laid out in A.3, making up the $\vdash'_+$ system.

We first define a measure[14] $\mathcal{M}$ over the formulae of this logic, according to tab. A.5. This measure satisfies the following strict decrease property:

Property 7 *For all rule*

$$\frac{\mathcal{H}_1 \cdots \mathcal{H}_n}{\mathcal{C}}$$

of $\vdash'_+$, except the equivalence rules, we have:

$$\mathcal{M}(\mathcal{C}) \leq \max\{\mathcal{M}(\mathcal{H}_1), \ldots, \mathcal{M}(\mathcal{H}_n)\} - 1.$$

That property implies the following:

Lemma 8 *Throughout the non-equivalence steps of a derivation, the length of the belief modality chain decreases.*

We propose the following strategy to decide quantifier-free GNY logic: forward-chain all rules of $\vdash'_+$ until no new derived formula gets added.

Theorem 9 *This strategy is a decision procedure.*

[14] The definition of that measure was done manually; but one could have automated that step by looking for coefficients α_T defining a measure $\mathcal{M}$ so that for all constructor T, of arity n,

$$\mathcal{M}(T(t_1, \ldots, t_n)) = \alpha_T + \sum_i \mathcal{M}(t_i)$$

and so that property 7 holds. It amounts to generating a set of linear inequalities (one per rule) in the unknown α_T and solving it.

t	$\mathcal{M}(t)$
Beliefs and conveyance	
$P \models X$	$\mathcal{M}(X)$
$P \triangleleft X$	$20 + \mathcal{M}(X)$
$P \ni X$	$\mathcal{M}(X)$
$P \mid\sim X$	$3 + \mathcal{M}(X)$
Properties of formulae	
$\sharp(X)$	$2 + \mathcal{M}(X)$
$\phi(X)$	$\mathcal{M}(X)$
Keys and secrets	
$P \overset{S}{\longleftrightarrow} Q$	0
$\overset{+P}{\mapsto} P$	0
$+K$	0
$-K$	0
Message constructors	
$\{X\}_K$	$3 + \mathcal{M}(C) + \mathcal{M}(K)$
$\{X\}_K^{-1}$	$3 + \mathcal{M}(C) + \mathcal{M}(K)$
(X, Y)	$1 + \mathcal{M}(X) + \mathcal{M}(Y)$
$F(X, Y)$	$2 + \mathcal{M}(X) + \mathcal{M}(Y)$
$H(X)$	$2 + \mathcal{M}(X)$
Jurisdiction	
$P \models\!\Rightarrow X$	$1 + \mathcal{M}(X)$
$P \models\!\Rightarrow P \models \star$	0
$(X \rightsquigarrow C)$	$\mathcal{M}(C)$
Detection of replays	
$\star X$	$1 + \mathcal{M}(X)$
Proof direction	
$\boxdot X$	$1 + \mathcal{M}(X)$

Table A.5: The measure $\mathcal{M}$ used, defined inductively on the formulae.

- it **terminates**, since
  - there exists a finite number of rules that can be applied, since there is a finite number of original rules and there's a upper bound on the depth of the modality chains concerned by the rationality rule, following from lemma 8;
  - the number of rules that can be applied in such a derivation is bounded because:
    - the number of non-equivalence steps is, because of property 7;
    - the number of equivalence steps is, because repeatedly applying equivalence rules from an initial formula yields to a finite set of derived formulae; our system can't derive already derived formulae;
- it's **complete** and **correct**, following from property 6 and theorem 4.

A.5.4 A practical implementation

Any sufficiently fast and convenient forward-chaining engine would do; however, as we didn't have that kind of tool available at the time we did that work, we quickly implemented such an engine for our problem.

Transformations

Internally, we use forward-chaining on the $\vdash'_+$ rules. When attempting to decide whether $\Gamma \vdash t$, the algorithm converts the problem into $\Gamma, \boxdot t \vdash'_+ t$ and tries to solve it.

A side-effect of this transformation is that in the case of failure, the system may give some crude clues as what extra assumptions would have been needed: just collect the formulae f so that $f \neq t$ and $\boxdot f$ but not f have been derived.

Avoiding combinatorial explosion

In this paragraph, we don't consider the rationality rule.

Implementing a working forward-chaining strategy for this problem is not as trivial as it seems. For instance, trying all possible rules in the fashion that to try a n-ary rule you match it against all the

```
do
    for all rule R having d data hypotheses
                          k key hypotheses
    do
        for all d-uple D of "old" formulae
        do
            if D matches the data hypotheses pattern
            then
                    if the key hypotheses are in
                       the "old" formulae
                    then put the conclusions into
                            the "new" formulae
    add the "new" formulae that were not already in
    the "old" formulae to the "old" formulae
until no more formulae are added
```

Figure A.1: Practical GNY logic decision procedure.

n-uples of already derived formulae, until it ends, leads to prohibitive costs (in our case, $n = 6$, which makes the number of matchings grow in D^7, where D is the number of derivable formulae).

A first optimization we tried was based on the fact that one needn't test all possible n-uples, but only ones containing at least one "new" formula; that is, a formula made during the last application of the rules. This is not sufficient, since it reduces only to d^6 ...

Our implementation is based on the fact that to instanciate all the variables in a rule, you needn't consider all the hypotheses; especially, in the modified versions of the construction rules, only one "goal" hypothesis suffices; in the destruction and trigger rules, only the data hypotheses are needed. In the GNY logic, there are at most two data hypotheses. Our algorithm is in fig. A.1.

To refine that algorithm even more, the pattern matching over d-uples is replaced by d successive pattern matchings, and any failure of one the k searches ends the search process with a negative answer.

Such an algorithm is at most in $D^{d+1}(\log D)^k$.

The rationality rule

The rationality rule can be implemented by a recursive call: if a formula $P \models X$ is present, call the forward-chaining algorithm on all the formulae whose root are $P \models \bullet$ or $\boxdot P \models \bullet$.

A translator

Instead of doing all the translation job from rules to program by hand, which would have been very tedious and prone to random and hard to catch typos, we preferred to program a translator from the set of rules to the target programming language to build the inference engine. That way, the only part that is subject to random typos is the relatively small set of rules; errors in the translator would result in systematic errors in the set of effectively applied rules and would get spotted. This translator outputs Objective CAML[15] code, but it would be easily adaptable to any strongly typed functional programming language.

The resulting code, along with some support modules (lexer/parser, pretty-printer ...) is then compiled and makes a protocol analysis tool. A sample run is given in appendix A.8, on which we measured the following computation times:

Machine / CPU		runtime system	
		native	bytecode
SparcStation 20	50 MHz	3.1	11.9
P6	200 MHz	0.6	3.0

The program

As in the sample shown in appendix A.8, our program takes an input of assumptions, then of protocol steps (which are turned into GNY formulas) and of desired conclusions. The program then tests each of the conclusions, and reports a proof in cases of success. In cases when the conclusion is unreachable, it outputs a hint to extra needed assumptions as described in A.5.4.

A.5.5 Other possible implementation

Again, any fast-forward chaining system will do. We considered adding a back-end to the translator into a controlled-rewriting system,[16]taking advan-

[15] Objective CAML is an implementation of the CAML dialect of the ML functional programming language, developped at INRIA. See `http://pauillac.inria.fr/ocaml/` for more information.

[16] For instance, the system Maude (see [CELM96]). We tested the possibility of it on the simpler problem of knowledge sets.

tage of the advanced pattern-matching algorithms of such systems, but adequate performance from our cruder implementation had us conclude that such step wasn't necessary.

A.6 Automatic analyses in BAN logic

A.6.1 The Otway-Rees protocol

Protocol

1	$A \to B$	$\{(N_a, N_c)\}_{K_{as}}$
2	$B \to S$	$\{(N_a, N_c)\}_{K_{as}}, \quad N_b, N_c \quad {}_{K_{bs}}$
3	$S \to B$	$\left(\left\{ N_a, A \overset{K_{ab}}{\longleftrightarrow} B, B \mid\sim N_c \right\}_{K_{as}}, \left\{ N_b, A \overset{K_{ab}}{\longleftrightarrow} B, A \mid\sim N_c \right\}_{K_{bs}}\right)$
4	$B \to A$	$\left\{ N_a, A \overset{K_{ab}}{\longleftrightarrow} B, B \mid\sim N_c \right\}_{K_{as}}$

Assumptions

$A \mid\equiv A \overset{K_{as}}{\longleftrightarrow} S$
$S \mid\equiv A \overset{K_{as}}{\longleftrightarrow} S$
$B \mid\equiv B \overset{K_{bs}}{\longleftrightarrow} S$
$S \mid\equiv B \overset{K_{bs}}{\longleftrightarrow} S$
$S \mid\equiv A \overset{K_{ab}}{\longleftrightarrow} B$
$A \mid\equiv S \mid\Rightarrow A \overset{K_{ab}}{\longleftrightarrow} B$
$B \mid\equiv S \mid\Rightarrow A \overset{K_{ab}}{\longleftrightarrow} B$
$A \mid\equiv S \mid\Rightarrow B \mid\sim N_c$
$B \mid\equiv S \mid\Rightarrow A \mid\sim N_c$
$A \mid\equiv \sharp(N_a)$
$B \mid\equiv \sharp(N_b)$
$A \mid\equiv \sharp(N_c)$

Goals

$A \mid\equiv A \overset{K_{ab}}{\longleftrightarrow} B$
$B \mid\equiv A \overset{K_{ab}}{\longleftrightarrow} B$
$A \mid\equiv B \mid\equiv N_c$
$B \mid\equiv A \mid\sim N_c$

A.6.2 The Needham-Shroeder shared-keys protocol

Protocol

1	$S \to A$	$\left\{ N_a, A \overset{K_{ab}}{\longleftrightarrow} B, \sharp(A \overset{K_{ab}}{\longleftrightarrow} B), \quad A \overset{K_{ab}}{\longleftrightarrow} B \right\}_{K_{as}}$
2	$A \to B$	$A \overset{K_{ab}}{\longleftrightarrow} B \quad {}_{K_{bs}}$
3	$B \to A$	$N_b, A \overset{K_{ab}}{\longleftrightarrow} B \quad {}_{K_{ab}}$
4	$A \to B$	$N_b, A \overset{K_{ab}}{\longleftrightarrow} B \quad {}_{K_{ab}}$

Assumptions

$A \mid\equiv A \overset{K_{as}}{\longleftrightarrow} S$
$B \mid\equiv B \overset{K_{bs}}{\longleftrightarrow} S$
$S \mid\equiv A \overset{K_{as}}{\longleftrightarrow} S$
$S \mid\equiv B \overset{K_{bs}}{\longleftrightarrow} S$
$S \mid\equiv A \overset{K_{ab}}{\longleftrightarrow} B$
$A \mid\equiv S \mid\Rightarrow A \overset{K_{ab}}{\longleftrightarrow} B$
$B \mid\equiv S \mid\Rightarrow A \overset{K_{ab}}{\longleftrightarrow} B$
$A \mid\equiv \sharp(N_a)$
$B \mid\equiv \sharp(N_b)$
$S \mid\equiv \sharp(A \overset{K_{ab}}{\longleftrightarrow} B)$
$B \mid\equiv \sharp(A \overset{K_{ab}}{\longleftrightarrow} B)$

Goals

$A \mid\equiv A \overset{K_{ab}}{\longleftrightarrow} B$
$B \mid\equiv A \overset{K_{ab}}{\longleftrightarrow} B$
$A \mid\equiv B \mid\equiv A \overset{K_{ab}}{\longleftrightarrow} B$
$B \mid\equiv A \mid\equiv A \overset{K_{ab}}{\longleftrightarrow} B$

A.6.3 The Kerberos protocol

Protocol

1	$S \to A$	$\left\{ T_s, A \overset{K_{ab}}{\longleftrightarrow} B, \quad T_s, A \overset{K_{ab}}{\longleftrightarrow} B \quad {}_{K_{bs}} \right\}_{K_{as}}$
2	$A \to B$	$T_s, A \overset{K_{ab}}{\longleftrightarrow} B \quad {}_{K_{bs}}, \quad T_a, A \overset{K_{ab}}{\longleftrightarrow} B \quad {}_{K_{ab}}$
3	$B \to A$	$T_a, A \overset{K_{ab}}{\longleftrightarrow} B \quad {}_{K_{ab}}$

Assumptions

$A \mid\equiv A \overset{K_{as}}{\longleftrightarrow} S$
$B \mid\equiv B \overset{K_{bs}}{\longleftrightarrow} S$
$S \mid\equiv A \overset{K_{as}}{\longleftrightarrow} S$
$S \mid\equiv B \overset{K_{bs}}{\longleftrightarrow} S$
$S \mid\equiv A \overset{K_{ab}}{\longleftrightarrow} B$
$A \mid\equiv S \mid\Rightarrow A \overset{K_{ab}}{\longleftrightarrow} B$
$B \mid\equiv S \mid\Rightarrow A \overset{K_{ab}}{\longleftrightarrow} B$
$A \mid\equiv \sharp(T_s)$
$B \mid\equiv \sharp(T_s)$
$B \mid\equiv \sharp(T_a)$
$A \mid\equiv \sharp(T_a)$

Goals

$A \mid\equiv A \overset{K_{ab}}{\longleftrightarrow} B$
$B \mid\equiv A \overset{K_{ab}}{\longleftrightarrow} B$
$A \mid\equiv B \mid\equiv A \overset{K_{ab}}{\longleftrightarrow} B$
$B \mid\equiv A \mid\equiv A \overset{K_{ab}}{\longleftrightarrow} B$

There's a minor error in [BAN89]. The authors claim to derive $A \models B \models A \overset{K_{ab}}{\longleftrightarrow} B$, which is incorrect with the assumptions they have written. One has to add the very reasonable hypothesis $A \models \sharp(N_a)$ (A should indeed believe that the timestamp it gives, T_a, is correct!) to have it work. This has been detected (independently) in [CS96] and [Sch97].

A.6.4 The wide-mouthed frog protocol

Protocol

1	$A \to S$	$\left\{\left(T_a, A \overset{K_{ab}}{\longleftrightarrow} B\right)\right\}_{K_{as}}$
2	$S \to B$	$\left\{\left(T_s, A \models A \overset{K_{ab}}{\longleftrightarrow} B\right)\right\}_{K_{bs}}$

Assumptions

$$
\begin{array}{l}
A \models A \overset{K_{as}}{\longleftrightarrow} S \\
B \models B \overset{K_{bs}}{\longleftrightarrow} S \\
S \models A \overset{K_{as}}{\longleftrightarrow} S \\
S \models B \overset{K_{bs}}{\longleftrightarrow} S \\
A \models A \overset{K_{ab}}{\longleftrightarrow} B \\
B \models A \models\!\Rightarrow A \overset{K_{ab}}{\longleftrightarrow} B \\
B \models S \models\!\Rightarrow A \models A \overset{K_{ab}}{\longleftrightarrow} B \\
S \models \sharp(T_a) \\
B \models \sharp(T_s)
\end{array}
$$

Goals

$$
\begin{array}{l}
A \models A \overset{K_{ab}}{\longleftrightarrow} B \\
B \models A \overset{K_{ab}}{\longleftrightarrow} B \\
B \models A \models A \overset{K_{ab}}{\longleftrightarrow} B
\end{array}
$$

A.6.5 The Needham-Shroeder public-key protocol

Protocol

1	$S \to A$	$\left\{\overset{+B}{\mapsto} B\right\}_{K_s^{-1}}$
2	$A \to B$	$\{N_a\}_{K_b}$
3	$S \to B$	$\left\{\overset{+A}{\mapsto} A\right\}_{K_s^{-1}}$
4	$B \to A$	$\left\{\left\langle\left(N_a, A \overset{N_b}{\longleftrightarrow} B\right)\right\rangle_{N_a}\right\}_{K_a}$
5	$A \to B$	$\left\{\left\langle\left(N_b, A \overset{N_a}{\longleftrightarrow} B, B \models A \overset{N_b}{\longleftrightarrow} B\right)\right\rangle_{N_b}\right\}_{K_b}$

Assumptions

$$
\begin{array}{l}
A \models \overset{+A}{\mapsto} A \\
A \models \overset{+S}{\mapsto} S \\
B \models \overset{+B}{\mapsto} B \\
B \models \overset{+S}{\mapsto} S \\
S \models \overset{+A}{\mapsto} A \\
S \models \overset{+B}{\mapsto} B \\
S \models \overset{+S}{\mapsto} S \\
A \models S \models\!\Rightarrow \overset{+B}{\mapsto} B \\
B \models S \models\!\Rightarrow \overset{+A}{\mapsto} A \\
A \models \sharp(N_a) \\
B \models \sharp(N_b) \\
A \models A \overset{N_a}{\longleftrightarrow} B \\
B \models A \overset{N_b}{\longleftrightarrow} B \\
A \models \sharp(\overset{+B}{\mapsto} B) \\
B \models \sharp(\overset{+A}{\mapsto} A)
\end{array}
$$

Goals

$$
\begin{array}{l}
A \models \overset{+B}{\mapsto} B \\
B \models \overset{+A}{\mapsto} A \\
A \models B \models A \overset{N_b}{\longleftrightarrow} B \\
B \models A \models A \overset{N_a}{\longleftrightarrow} B \\
B \models A \models B \models A \overset{N_b}{\longleftrightarrow} B
\end{array}
$$

Here, we had to deal with another minor problem. The authors seem to use a rule, correct for the informal semantics of the logic, but unsaid in the specification, that

$$\frac{P \models \langle X \rangle_Y \qquad P \models \sharp(Y)}{P \models \sharp(\langle X \rangle_Y)}.$$

We had to replace instances of $\langle X \rangle_Y$ by $\langle (Y, X) \rangle_Y$,
and we get the correct results.

A.7 The rules of modified GNY logic

The rules with a "primed" name are those added to original GNY logic to match the "normal form" requirement (see section A.3); this transformation is described in A.5.2.

For better readbility, the destruction rules have been typeset in the following fashion:

$$\frac{\boxed{\mathcal{D}_1 \cdots \mathcal{D}_m}\, \mathcal{K}_1 \cdots \mathcal{K}_n}{\mathcal{C}};$$

moreover, rules

$$\frac{\mathcal{H}_1 \cdots \mathcal{H}_m}{\mathcal{C}_1 \cdots \mathcal{C}_n}$$

are short for n rules

$$\frac{\mathcal{H}_1 \cdots \mathcal{H}_m}{\mathcal{C}_i}.$$

A.7.1 Being-told rules

- T1 is a destruction rule:

$$\frac{\boxed{P \triangleleft \star X}}{P \triangleleft X}$$

- T2 is a destruction rule:

$$\frac{\boxed{P \triangleleft (X, Y)}}{P \triangleleft X \quad P \triangleleft Y}$$

- T3 is a destruction rule:

$$\frac{\boxed{P \triangleleft \{X\}_K} \quad P \ni K}{P \triangleleft X}$$

- T4 is a destruction rule:

$$\frac{\boxed{P \triangleleft \{X\}_{+K}} \quad P \ni -K}{P \triangleleft X}$$

- T5 is a destruction rule:

$$\frac{\boxed{P \triangleleft F(X, Y)} \quad P \ni X}{P \triangleleft Y}$$

- TZ is a destruction rule:

$$\frac{\boxed{P \triangleleft (X \rightsquigarrow Y)}}{P \triangleleft X}$$

The TZ rule, though of obviously right informal meaning, is not in the original GNY paper; nevertheless it is necessary to run the logic and obtained the results advertised in that paper.

A.7.2 Possession rules

- P1 is a destruction rule:

$$\frac{\boxed{P \triangleleft X}}{P \ni X}$$

- P2 is a construction rule:

$$\frac{P \ni X \quad P \ni Y}{P \ni (X, Y) \quad P \ni F(X, Y)}$$

- P3 is a destruction rule:

$$\frac{\boxed{P \ni (X, Y)}}{P \ni X \quad P \ni Y}$$

- P4 is a construction rule:

$$\frac{P \ni X}{P \ni H(X)}$$

- P5 is a destruction rule:

$$\frac{\boxed{P \ni F(X, Y)} \quad P \ni X}{P \ni Y}$$

- P6 is a construction rule:

$$\frac{P \ni X \quad P \ni K}{P \ni \{X\}_K \quad P \ni \{X\}_K^{-1}}$$

- P6' is a destruction rule:

$$\frac{\boxed{P \ni \{X\}_K} \quad P \ni K}{P \ni X}$$

- P7 is a construction rule:

$$\frac{P \ni +K \quad P \ni X}{P \ni \{X\}_{+K}}$$

- P8 is a construction rule:
$$\frac{P \ni -K \quad P \ni X}{P \ni \{X\}_{-K}}$$
- P8' is a destruction rule:
$$\frac{\boxed{P \ni \{X\}_{+K}} \quad P \ni -K}{P \ni X}$$

A.7.3 Freshness rules

- F1 is a construction rule:
$$\frac{P \models \sharp(X)}{P \models \sharp((X,Y)) \quad P \models \sharp((Y,X)) \quad P \models \sharp(F(X))}$$
- F2 is a construction rule:
$$\frac{P \models \sharp(X) \quad P \ni K}{P \models \sharp(\{X\}_K) \quad P \models \sharp(\{X\}_K^{-1})}$$
- F2' is a destruction rule:
$$\frac{\boxed{P \models \sharp(\{X\}_K)} \quad P \ni K}{P \models \sharp(X)}$$
- F3 is a construction rule:
$$\frac{P \models \sharp(X) \quad P \ni +K}{P \models \sharp(\{X\}_{+K})}$$
- F4 is a construction rule:
$$\frac{P \models \sharp(X) \quad P \ni -K}{P \models \sharp(\{X\}_{-K})}$$
- F4' is a destruction rule:
$$\frac{\boxed{P \models \sharp(\{X\}_{+K})} \quad P \ni -K}{P \models \sharp(X)}$$
- F5 is an equivalence rule:
$$\frac{\boxed{P \models \sharp(+K)}}{P \models \sharp(-K)}$$
- F6 is an equivalence rule:
$$\frac{\boxed{P \models \sharp(-K)}}{P \models \sharp(+K)}$$
- F7 is a construction rule:
$$\frac{P \models \phi(X) \quad P \models \sharp(K) \quad P \ni K}{P \models \sharp(\{X\}_K) \quad P \models \sharp(\{X\}_K^{-1})}$$
- F8 is a construction rule:
$$\frac{P \models \phi(X) \quad P \models \sharp(+K) \quad P \ni +K}{P \models \sharp(\{X\}_{+K})}$$
- F9 is a construction rule:
$$\frac{P \models \phi(X) \quad P \models \sharp(-K) \quad P \ni -K}{P \models \sharp(\{X\}_{-K})}$$
- F10 is a construction rule:
$$\frac{P \models \sharp(X) \quad P \ni X}{P \models \sharp(H(X))}$$
- F11 is a destruction rule:
$$\frac{\boxed{P \models \sharp(H(X))} \quad P \ni H(X)}{P \models \sharp(X)}$$

A.7.4 Recognizability rules

- R1 is a construction rule:
$$\frac{P \models \phi(X)}{P \models \phi((X,Y)) \quad P \models \phi((Y,X)) \quad P \models \phi(F(X))}$$
- R2 is a construction rule:
$$\frac{P \models \phi(X) \quad P \ni K}{P \models \phi(\{X\}_K) \quad P \models \phi(\{X\}_K^{-1})}$$
- R2' is a destruction rule:
$$\frac{\boxed{P \models \phi(\{X\}_K)} \quad P \ni K}{P \models \phi(X)}$$
- R3 is a construction rule:
$$\frac{P \models \phi(X) \quad P \ni +K}{P \models \phi(\{X\}_{+K})}$$
- R4 is a construction rule:
$$\frac{P \models \phi(X) \quad P \ni -K}{P \models \phi(\{X\}_{-K})}$$
- R4' is a destruction rule:
$$\frac{\boxed{P \models \phi(\{X\}_{+K})} \quad P \ni -K}{P \models \phi(X)}$$
- R5 is a construction rule:
$$\frac{P \models \phi(X) \quad P \ni X}{P \models \phi(H(X))}$$
- R6 is a destruction rule:
$$\frac{\boxed{P \ni H(X)}}{P \models \phi(X)}$$

A.7.5 Message interpretation rules

- I1 is a destruction rule:

$$\frac{\boxed{P \triangleleft \star\{X\}_K \quad P \models P \overset{K}{\longleftrightarrow} Q} \quad P \ni K \quad P \models \phi(X) \quad P \models \sharp((X,K))}{P \models Q \mid\sim X \quad P \models Q \mid\sim \{X\}_K \quad P \models Q \ni K}$$

- I2 is a destruction rule:

$$\frac{\boxed{P \triangleleft \star\{(X,S)\}_{+K} \quad P \models P \overset{S}{\longleftrightarrow} Q} \quad P \ni (-K,S) \quad P \models \overset{+P}{\mapsto} P \quad P \models \phi((X,S)) \quad P \models \sharp((X,S,+K))}{P \models Q \mid\sim (X,S) \quad P \models Q \mid\sim \{(X,S)\}_{+K} \quad P \models Q \ni +K}$$

- I3 is a destruction rule:

$$\frac{\boxed{P \triangleleft \star H(X,S) \quad P \models P \overset{S}{\longleftrightarrow} Q} \quad P \ni (X,S) \quad P \models \sharp((X,S))}{P \models Q \mid\sim (X,S) \quad P \models Q \mid\sim H(X,S)}$$

- I4 is a destruction rule:

$$\frac{\boxed{P \triangleleft \{X\}_{-K} \quad P \models \overset{+Q}{\mapsto} Q} \quad P \ni +K \quad P \models \phi(X)}{P \models Q \mid\sim X \quad P \models Q \mid\sim \{X\}_{-K}}$$

- I5 is a destruction rule:

$$\frac{\boxed{P \triangleleft \{X\}_{-K} \quad P \models \overset{+Q}{\mapsto} Q} \quad P \ni +K \quad P \models \phi(X) \quad P \models \sharp((X,+K))}{P \models Q \ni (-K,X)}$$

- I6 is a destruction rule:

$$\frac{\boxed{P \models Q \mid\sim X} \quad P \models \sharp(X)}{P \models Q \ni X}$$

- I7 is a destruction rule:

$$\frac{\boxed{P \models Q \mid\sim (X,Y)}}{P \models Q \mid\sim X \quad P \models Q \mid\sim Y}$$

A.7.6 Jurisdiction rules

- J1 is a destruction rule:

$$\frac{\boxed{P \models Q \Rightarrow C} \quad P \models Q \models C}{P \models C}$$

- J2 is a destruction rule:

$$\frac{\boxed{P \models Q \mid\sim (X \rightsquigarrow C)} \quad P \models Q \Rightarrow Q \models \star}{P \models Q \models C}$$

- J3 is a destruction rule:

$$\frac{\boxed{P \models Q \models Q \models C} \quad P \models Q \Rightarrow Q \models \star}{P \models Q \models C}$$

A.8 GNY analysis of some properties of the Needham-Shroeder protocol

```
PROTOCOL "Needham-Shroeder";

ASSUMPTIONS
A: P $ Kps;
B: P $ Np;
C: P |= P <-Kps-> S;
D: P |= #Np;
E: P |= %Q;
F: Q $ Kqs;
G: Q $ Nq;
H: Q |= Q <-Kqs-> S;
I: Q |= #Nq;
J: Q |= %Nq;
K: P |= S => (P <-K-> Q);
L: P |= S @@@;
M: P |= Q @@@;
N: Q |= S => (P <-K-> Q);
O: Q |= S @@@;
P: Q |= P @@@;
Q: S $ Kps;
R: S $ Kqs;
S: S $ K;
T: S |= (P <-Kps-> S);
U: S |= (Q <-Kqs-> S);
V: S |= P <-K-> Q;

MESSAGES

P->S: ( *P, *Q, *Np);
S->P: ( *{Np, Q, *K, ( *{K,P} Kqs ~> S |= P <-K-> Q)} Kps ~> S |= P <-K-> Q);
P->Q: ( *{*K, *P} Kqs ~> S |= P <-K-> Q);
Q->P: *{*Nq} K;
P->Q: ( *{*F(Nq)}K ~> P |= P <-K-> Q);

GOALS

P |= P <-K-> Q;
P $ Nq;

END
```

The LaTeX code for this analysis report has been automatically generated, then manually touched up for better page layout.

A.8.1 Assumptions

A $P \ni K_{ps}$

B $P \ni N_p$

C $P \models P \overset{K_{ps}}{\longleftrightarrow} S$

D $P \models \sharp(N_p)$

E $P \models \phi(Q)$

F $Q \ni K_{qs}$

G $Q \ni N_q$

H $Q \models Q \overset{K_{qs}}{\longleftrightarrow} S$

I $Q \models \sharp(N_q)$

J $Q \models \phi(N_q)$

K $P \models S \models\!\Rightarrow P \overset{K}{\longleftrightarrow} Q$

L $P \models S \models\!\Rightarrow S \models \star$

M $P \models Q \models\!\Rightarrow Q \models \star$

N $Q \models S \models\!\Rightarrow P \overset{K}{\longleftrightarrow} Q$

O $Q \models S \models\!\Rightarrow S \models \star$

P $Q \models P \models\!\Rightarrow P \models \star$

Q $S \ni K_{ps}$

R $S \ni K_{qs}$

S $S \ni K$

T $S \models P \overset{K_{ps}}{\longleftrightarrow} S$

U $S \models Q \overset{K_{qs}}{\longleftrightarrow} S$

V $S \models P \overset{K}{\longleftrightarrow} Q$

A.8.2 Protocol steps

1. $P \rightarrow S : (\star P, \star Q, \star N_p)$
2. $S \rightarrow P : \left(\star \left\{ \left(N_p, Q, \star K, \left(\star \{(K, P)\}_{K_{qs}} \rightsquigarrow S \models P \overset{K}{\longleftrightarrow} Q\right)\right)\right\}_{K_{ps}} \rightsquigarrow S \models P \overset{K}{\longleftrightarrow} Q\right)$
3. $P \rightarrow Q : \left(\star \{(\star K, \star P)\}_{K_{qs}} \rightsquigarrow S \models P \overset{K}{\longleftrightarrow} Q\right)$
4. $Q \rightarrow P : \star \{\star N_q\}_K$
5. $P \rightarrow Q : \left(\star \{\star F(N_q)\}_K \rightsquigarrow P \models P \overset{K}{\longleftrightarrow} Q\right)$

A.8.3 Goals

- $P \models P \overset{K}{\longleftrightarrow} Q$,
  applying rule J1 to:
  $P \models S \models\!\Rightarrow P \overset{K}{\longleftrightarrow} Q$,
  from hypothesis **K**.
  $P \models S \models P \overset{K}{\longleftrightarrow} Q$,
  applying rule J3 to:
  $P \models S \models S \models P \overset{K}{\longleftrightarrow} Q$,
  applying rule J2 to:
  $P \models S \vdash \ \star \{(K, P)\}_{K_{qs}} \rightsquigarrow S \models P \overset{K}{\longleftrightarrow} Q$,
  applying rule I7 to:
  $P \models S \vdash \ \star K, \ \star \{(K, P)\}_{K_{qs}} \rightsquigarrow S \models P \overset{K}{\longleftrightarrow} Q$,
  applying rule I7 to:
  $P \models S \vdash \ Q, \star K, \ \star \{(K, P)\}_{K_{qs}} \rightsquigarrow S \models P \overset{K}{\longleftrightarrow} Q$,
  applying rule I7 to:
  $P \models S \vdash \ N_p, Q, \star K, \ \star \{(K, P)\}_{K_{qs}} \rightsquigarrow S \models P \overset{K}{\longleftrightarrow} Q$,
  applying rule I1 to:

$P \triangleleft \star \left\{ \ N_p, Q, \star K, \ \star \{(K,P)\}_{K_{qs}} \rightsquigarrow S \equiv P \overset{K}{\longleftrightarrow} Q \ \right\}_{K_{ps}}$,
applying rule T2Z to:
$P \triangleleft \ \star \left\{ \ N_p, Q, \star K, \ \star \{(K,P)\}_{K_{qs}} \rightsquigarrow S \equiv P \overset{K}{\longleftrightarrow} Q \ \right\}_{K_{ps}} \rightsquigarrow S \equiv P \overset{K}{\longleftrightarrow} Q$,
from hypothesis **Step #2**.
$P \equiv P \overset{K_{ps}}{\longleftrightarrow} S$,
from hypothesis **C**.
$P \ni K_{ps}$,
from hypothesis **A**.
$P \equiv \phi(\ N_p, Q, \star K, \ \star \{(K,P)\}_{K_{qs}} \rightsquigarrow S \equiv P \overset{K}{\longleftrightarrow} Q \ )$,
applying rule R1 to:
$P \equiv \phi(\ Q, \star K, \ \star \{(K,P)\}_{K_{qs}} \rightsquigarrow S \equiv P \overset{K}{\longleftrightarrow} Q \ )$,
applying rule R1 to:
$P \equiv \phi(Q)$,
from hypothesis **E**.
$P \equiv \sharp(\ N_p, Q, \star K, \ \star \{(K,P)\}_{K_{qs}} \rightsquigarrow S \equiv P \overset{K}{\longleftrightarrow} Q \ , K_{ps} \ )$,
applying rule F1 to:
$P \equiv \sharp(\ N_p, Q, \star K, \ \star \{(K,P)\}_{K_{qs}} \rightsquigarrow S \equiv P \overset{K}{\longleftrightarrow} Q \ )$,
applying rule F1 to:
$P \equiv \sharp(N_p)$,
from hypothesis **D**.
$P \equiv S \models\!\Rightarrow S \equiv \star$,
from hypothesis **L**.
$P \equiv S \models\!\Rightarrow S \equiv \star$,
from hypothesis **L**.

- $P \ni N_q$,
  applying rule P1 to:
  $P \triangleleft N_q$,
  applying rule T1 to:
  $P \triangleleft \star N_q$,
  applying rule T3 to:
  $P \triangleleft \{\star N_q\}_K$,
  applying rule T1 to:
  $P \triangleleft \star \{\star N_q\}_K$,
  from hypothesis **Step #4**.
  $P \ni K$,
  applying rule P1 to:
  $P \triangleleft K$,
  applying rule T1 to:
  $P \triangleleft \star K$,
  applying rule T2 to:
  $P \triangleleft \ \star K, \ \star \{(K,P)\}_{K_{qs}} \rightsquigarrow S \equiv P \overset{K}{\longleftrightarrow} Q$,
  applying rule T2 to:
  $P \triangleleft \ Q, \star K, \ \star \{(K,P)\}_{K_{qs}} \rightsquigarrow S \equiv P \overset{K}{\longleftrightarrow} Q$,
  applying rule T2 to:
  $P \triangleleft \ N_p, Q, \star K, \ \star \{(K,P)\}_{K_{qs}} \rightsquigarrow S \equiv P \overset{K}{\longleftrightarrow} Q$,
  applying rule T3 to:
  $P \triangleleft \left\{ \ N_p, Q, \star K, \ \star \{(K,P)\}_{K_{qs}} \rightsquigarrow S \equiv P \overset{K}{\longleftrightarrow} Q \ \right\}_{K_{ps}}$,

applying rule $\mathsf{T1}$ to:

$$P \triangleleft \star \Big\{ \quad N_p, Q, \star K, \quad \star \{(K,P)\}_{K_{qs}} \rightsquigarrow S \models P \overset{K}{\longleftrightarrow} Q \quad \Big\}_{K_{ps}},$$

applying rule TZ to:

$$P \triangleleft \quad \star \Big\{ \quad N_p, Q, \star K, \quad \star \{(K,P)\}_{K_{qs}} \rightsquigarrow S \models P \overset{K}{\longleftrightarrow} Q \quad \Big\}_{K_{ps}} \rightsquigarrow S \models P \overset{K}{\longleftrightarrow} Q \quad ,$$

from hypothesis **Step #2**.

$P \ni K_{ps}$,

from hypothesis **A**.

Annexe B

Quelques idées sur les possibilités d'abstraction dans le problème des lancers de protocoles entrelacés

B.1 Definitions

B.1.1 A strong threat model

We assume that the intruder has full power over the communication network. All communication thus goes through and by the intruder. The intruder may choose to relay or not messages, to delay them, to modify them etc ... The only restriction to its power is that it can't send out data that is not in his knowledge at the moment of the sending — for instance, it can't decrypt data encrypted with a key that it doesn't own.

B.1.2 Protocol runs

Let $\mathcal{D}$ be the set of possible conveyable data. We have a knowledge production function $k : \mathcal{P}(\mathcal{D}) \to \mathcal{P}(\mathcal{D})$ that, informally, computes the knowledge you can extract from a set of data. Such a function must verify

$$\begin{cases} k \circ k = k & \text{(idempotence)} \\ \forall x, y \in \mathcal{P}(\mathcal{D}), \quad x \subset y \Rightarrow k(x) \subset k(y) & \text{(monotonicity)} \\ \forall x \in \mathcal{P}(\mathcal{D}), x \subset k(x) & \text{(expansion)} \end{cases}$$

We call $\bar{\mathcal{D}}$ the image of k; that is, the set of knowledge closures; we put a complete lattice structure over it with

$$x \vee_k y = k(x \cup y)$$

and $x \wedge y = x \cap y$; the bottom element is $\emptyset$ and the top element is $\mathcal{D}$.

Let $\varepsilon \in \mathcal{D}$ be a special value called the empty word; $\forall x \in \bar{\mathcal{D}}, \varepsilon \in x$.

Let P be the set of principals. Each principal $p \in P$ has a local state, taken in the set $\mathcal{S}_p$, runs according to a relation $r_p \subset (\mathcal{S}_p \times \mathcal{D})^2$. Informally, $r_p\left((s, m), (s', m')\right)$ means that when in state s and getting data m, p outputs message m' and moves to state s'.

The global state space is

$$\mathcal{S} = \prod_{p \in P} \mathcal{S}_p \times \bar{\mathcal{D}}.$$

and the stepping relation $\xrightarrow{s}$ is defined as[1]:

$$\begin{array}{ll} & (s_p)_{p\in P}, K_I \xrightarrow{s} (s'_p)_{p\in P}, K'_I \\ \Leftrightarrow & \exists m \in K_I, \exists m' \in \mathcal{D}, \exists p_0 \in P \\ & \quad \begin{cases} (s_{p_0}, m) \xrightarrow{r_p} (s'_{p_0}, m') \wedge \forall p \neq p_0, s'_p = s_p \\ K'_I = K_I \vee_k \{m'\} \end{cases} \end{array}$$

This definition is very broad, and even encompasses nondeterministic principals. Deterministic principals use a function $f_p : (\mathcal{S}_p \times \mathcal{D}) \to (\mathcal{S}_p \times \mathcal{D})$ so that r_p is the graph of f_p.

B.1.3 Common encryption knowledge

Let's consider $\mathcal{D}$ to be the free algebra on the constants in $\mathcal{D}_0$ with the following constructors: pairing $(\bullet, \bullet)$ and symmetric-key encryption $\{\bullet\}_\bullet$. To

[1] For a binary relation R, we'll sometimes use the notation $x \xrightarrow{R} y$ for $(x, y) \in R$.

limit confusion, we'll write singletons built on this algebra between angle brackets. We define $k(X)$ as the least fixpoint of the monotonic function (for the $\subset$ ordering of $+\mathcal{P}(\mathcal{D})$)

$$X \mapsto X \cup \left(\bigcup_{\langle (x,y) \rangle \in X} \{ \langle x \rangle, \langle y \rangle \} \right) \cup \left(\bigcup_{\langle \{x\}_k \rangle \in X \ / \ k \in X} \{ \langle x \rangle \} \right) \cup \left(\bigcup_{x,y \in X} \left\{ \langle (x,y) \rangle, \left\langle \{x\}_y \right\rangle \right\} \right).$$

Informally, $k(X)$ is all that we can get from X through pairing, splitting, encrypting and decrypting with a key we possess.

B.1.4 Common implementations in this framework

Most of the times, we consider local states of the form

$$\mathcal{S}_p = \underbrace{S_p}_{\text{program counter}} \times \underbrace{\mathcal{D}^{m_p}}_{\text{registers}}$$

where S_p is a (finite) set of steps and step relations r_p that so that for all initial local state $s(\sigma, \rho_1, \ldots, \rho_{m_p})$, all incoming message m, all final local state $s(\sigma', \rho'_1, \ldots, \rho'_{m_p})$ and all outgoing message m' so that $r_p((s,m),(s',m'))$, we have $\{m', \rho'_1, \ldots, \rho'_{m_p}\} \subset k(\{m, \rho_1, \ldots, \rho_{m_p}\})$.

B.2 A first abstraction

B.2.1 Informally

Let's first consider a simple example. Let's consider a protocol step in which a principal A receives a piece of data X and outputs the decryption of it with his private key $-K_a$. The intruder, at the beginning of this step, possesses $\{X\}_{+K_a}$ and $\{Y\}_{+K_a}$. The intruder has to make a choice on what piece of data to submit to A for decryption. A problem is the high nondeterminism of that choice: the intruder may try any piece of data in its knowledge.

We abstract that problem into a model where the intruder may "magically" submit simultaneously all data to its knowledge to the principal for decryption. Clearly, the outcome of intruder knowledge, in that abstraction, is always bigger than the possible real outcome. Thus, this abstraction is conservative, in the sense that it overestimates the intruder's knowledge.

B.2.2 Formally

Let's suppose we have abstract relations $\hat{r}_p \subset (\bar{\mathcal{S}}_p \times \bar{\mathcal{D}})^2$ governing the principals that satisfy the abstraction property[2]

$$\begin{array}{ccc} \mathcal{S}_p \times \mathcal{D} & \xrightarrow{\ s\ } & \mathcal{S}_p \times \mathcal{D} \\ \downarrow{\scriptstyle a_p \otimes a_{\mathcal{D}}} & & \downarrow{\scriptstyle a_p \otimes a_{\mathcal{D}}} \\ \hat{\mathcal{S}}_p \times \bar{\mathcal{D}} & \xrightarrow{\ \hat{s}\ } & \hat{\mathcal{S}}_p \times \bar{\mathcal{D}} \end{array}$$

where $a_{\mathcal{D}} = \{(x,y) \in \mathcal{D} \times \bar{\mathcal{D}} / x \in y\}$ and $a_p \subset \mathcal{S}_p \times \hat{\mathcal{S}}_p$.[3]

The abstract global state space is

$$\hat{\mathcal{S}} = \left(\prod_{p \in P} \hat{\mathcal{S}}_p \right) \times \bar{\mathcal{D}},$$

with the abstraction relation

$$a_{\mathcal{S}} = \left(\bigotimes_{p \in P} a_p \right) \otimes =_{\bar{\mathcal{D}}}$$

where $=_X$ denotes the equality on X and the abstract stepping relation $\xrightarrow{\hat{s}}$ is defined as:

$$(\hat{s}_p)_{p \in P}, K_I \xrightarrow{\hat{s}} (\hat{s}'_p)_{p \in P}, K'_I \quad \Leftrightarrow \quad \exists p_0 \in P \begin{cases} (\hat{s}_{p_0}, K_I) \xrightarrow{r_p} (\hat{s}'_{p_0}, M') \\ \forall p \neq p_0, \hat{s}'_p = \hat{s}_p \\ K'_I = K_I \vee_k M' \end{cases}$$

[2] We say that the abstraction property denoted by the diagram

$$\begin{array}{ccc} A & \xrightarrow{\ S\ } & B \\ \downarrow{\scriptstyle a_A} & & \downarrow{\scriptstyle a_B} \\ A' & \xrightarrow{\ S'\ } & B' \end{array}$$

is satisfied when for all $a \in A$, $b \in B$, $a' \in A'$ so that $S(a,b)$ and $a_A(a,a')$ then there exists b' so that $S'(a',b')$ and $a_B(b,b')$.

[3] If we have $r_a \subset A \times A'$ and $r_b \subset B \times B'$, we note $r_a \otimes r_b = \{ \ (a,b),(a',b') \ \in (A \times B) \times (A' \times B') / (a,a') \in r_a \vee (b,b') \in r_b \}$. Similarly we define $\bigotimes$ over indexed products.

Then we have the following property:

$$\begin{array}{ccc} \mathcal{S} & \xrightarrow[s]{} & \mathcal{S} \\ \downarrow a_{\mathcal{S}} & & \downarrow a_{\mathcal{S}} \\ \hat{\mathcal{S}} & \xrightarrow[\hat{s}]{} & \hat{\mathcal{S}} \end{array}$$

This means that all states that map to unreachable abstract states are unreachable. Thus a safety property proved in the abstraction implies a safety property of the real protocol.

B.2.3 Abstraction of the common implementations

We abstract the local states to

$$\hat{\mathcal{S}}_p = \Sigma_p \times \bar{\mathcal{D}}^{m_p}$$

by the abstraction $a_p = D_{\Sigma_p} \otimes {a_{\mathcal{D}}}^{\otimes m_p}$. Then the "smallest" abstract step relation $\hat{r}_p$ is

$$\begin{array}{l} \left((\sigma, \hat{\rho}_1, \ldots, \hat{\rho}_{m_p}), \hat{m}\right) \xrightarrow{\hat{r}_p} \left((\sigma', \hat{\rho}'_1, \ldots, \hat{\rho}'_{m_p}), \hat{m}'\right) \\ \Leftrightarrow \\ \left\{ \begin{array}{l} \forall 1 \leq i \leq m_p, \\ \quad \hat{\rho}'_i = \bigvee_k \quad \{\rho_i\} \\ \qquad \left((\sigma,\rho_1,\ldots,\rho_{m_p}),m\right) \xrightarrow{r_p} (\sigma',\rho'_1,\ldots,\rho'_{m_p}),m' \\ \qquad (\sigma,\rho_1,\ldots,\rho_{m_p}) \xrightarrow{a_p} (\sigma,\hat{\rho}_1,\ldots,\hat{\rho}_{m_p}) \\ \qquad m \xrightarrow{a_{\mathcal{D}}} \hat{m} \\ \hat{m}' = \bigvee_k \quad \{m\}. \\ \qquad \left((\sigma,\rho_1,\ldots,\rho_{m_p}),m\right) \xrightarrow{r_p} (\sigma',\rho'_1,\ldots,\rho'_{m_p}),m' \\ \qquad (\sigma,\rho_1,\ldots,\rho_{m_p}) \xrightarrow{a_p} (\sigma,\hat{\rho}_1,\ldots,\hat{\rho}_{m_p}) \\ \qquad m \xrightarrow{a_{\mathcal{D}}} \hat{m} \end{array} \right. \end{array}$$

B.2.4 A decision procedure for common-case abstraction

Definitions and properties

We consider the cases when the function k is defined as the closure by some formal rules making up the system $\vdash_k$ — as in B.1.3, for instance, though more complex systems can easily be envisioned. We restrict ourselves to the cases where the system of rules matches the strong normal derivation property and a strictly decreasing order property.

We restrict ourselves to the cases where principal actions are of the following form: (r, s) where r is a formula whose leaves are either constants, register names seen as values or register names seen as destinations, to be used as a pattern to which incoming messages are matched; s is a formula whose leaves are either constants or register names seen as values, to be used as the reply.

Given a (finite) sequence l of principals, representing the interleaving of principal execution steps, which we will tag by number n representing their execution ordering and the name P of the principal involved, we construct the $\vdash_{k,l}$ system of rules by prepending the following modalities to the $\vdash_k$ rules:

- $[n : P.R = X]$: at step n, principal P's register R contains data X,
- $[n : K_I \ni X]$: at step n, the intruder knows X.

The rules of the new system are those of $\vdash_k$ with modalities, to which we add rules for the abstract protocol steps: a tagged step $n.P : (r, s)$ is viewed as a rule

$$\frac{[n : K_I \ni r] \quad [n : P.R_1 = X_1] \ldots [n : P.R_m = X_m]}{[n+1 : K_I \ni s] \quad [n+1 : P.R_1 = X_1] \ldots [n+1 : P.R_q = X_q]},$$

where $[n : P.R_1 = X_1] \ldots [n : P.R_m = X_m]$ are constraints arising from the register values in the matches and $[n : P.R_1 = X_1] \ldots [n : P.R_m = X_m]$ are the affectations made by the matching.

We require that for each of those rules, the consequences, taking off the modalities, can be derived in $\vdash_k$ from the hypotheses, taking off the modalities. This informally means that the principals can only do "possible" computations, that is, computations along the lines of the $\vdash_k$ system.

The resulting system, augmented, matches the strong normal derivation property and the strictly decreasing order property. Thus we can use forward-chaining on the derived system as a decision procedure in that abstraction.

If each principal can only make a finite number of steps in a run, for a given number of principals, the number of executable sequences is finite. Therefore, making up all the sequences and, for each, running the above decision procedure yields a decision procedure for the abstract whole problem.

Proofs

For any rule

$$\frac{M_1.H_1 \quad \dots \quad M_n.H_n}{M_C.C} \vdash_{k,l}$$

of $\vdash_{k,l}$ where the M_x are modalities, constraints on the principals' computational power impose that

$$\begin{array}{c} H_1 \quad \dots \quad H_n \\ \vdots \ \vdash_k \\ C \end{array} .$$

We classify the H_i between data hypotheses and key hypotheses, according to their use in the former $\vdash_k$-derivation.

We complete the system into a new system $\vdash_{k,l}$ by considering all the possible "annoying pairs" of construction/destruction rules. For instance, the $\vdash_{k,l}$-rule

$$\frac{[3 : K_I \ni X]}{[4 : P.R_1 = \{X\}^{-1}_{K_P}]}$$

will be supplemented by the rule

$$\frac{[3 : K_I \ni \{X\}_{K_P}]}{[4 : P.R_1 = X]} .$$

If we have a strictly decreasing ordering $\leq_k$ on $\vdash_k$, then we make a strictly decreasing ordering $\leq_{k,l}$ on $\vdash_{k,l}$ by taking the lexicographic product ordering of $\geq$ over the sequence numbers and $\leq_k$.

Bibliographie

[BAN89] Michael Burrows, Martin Abadi, and Roger Needham. A logic of authentication. Technical Report 39, Digital Equipment Corporation, Systems Research Centre, February 1989. http://gatekeeper.dec.com/pub/DEC/SRC/research-reports/abstracts/src-rr-039.html

[BBC+97] Bruno Barras, Samuel Boutin, Cristina Cornes, Judicael Courant, Jean-Christophe Filliatre, Eduardo Gimenez, Hugo Herbelin, Gerard Huet, Cesar Munoz, Chetan Murthy, Catherine Parent, Christine Paulin-Mohring, Amokrane Saibi, and Benjamin Werner. *The Coq Proof Assistant Reference Manual : Version 6.1*. INRIA (Rocquencourt) / CNRS / ENS-Lyon, May 1997. ftp://ftp.inria.fr/INRIA/publication/RT/RT-0203.ps.gz

[Bol97] Dominique Bolignano. Towards a mechanization of cryptographic protocol verification. In Orna Grumberg, editor, *CAV'97: Computer Aided Verification, 9th International Conference, Haifa, Israel*, volume 1254 of *Lecture Notes in Computer Science*, pages 131 – 142. Springer, June 1997.

[Bra95] Stephen H. Brackin. Deciding cryptographic protocol adequacy with HOL. In *IWHOLTP: 8th International Workshop on Higher Order Logic Theorem Proving and Its Applications*, volume 971 of *Lecture Notes in Computer Science*. Springer-Verlag, 1995.

[Bra96] Stephen H. Brackin. Automatic formal analyses of cryptographic protocols. In *NISSC: 19th National Information Systems Security Conference*, pages I.181 – 193, Baltimore, MD, October 1996. National Institute of Standards and Technology and National Computer Security Center.

[CELM96] M. Clavel, S. Eker, P. Lincoln, and J. Meseguer. Principles of Maude. In J. Meseguer, editor, *Rewriting Logic and its Applications, First International Workshop*, volume 4 of *Electronic Notes in Theoretical Computer Science*, pages 65–89. Elsevier Science B.V., September 1996. Only available electronically, to the subscribers of *Theoretical Computer Science*. http://www1.elsevier.nl/mcs/tcs/pc/volume4.htm

[Cou97] Patrick Cousot. Introduction a l'interprétation abstraite. Transparents à l'occasion de l'École Jeunes Chercheurs en Programmation, Mars 1997. http://www.dmi.ens.fr/~cousot/cours/compilation/Cours-9-Compil-4-1.ps.gz

[CS96] Dan Craigen and Mark Saaltink. Using eves to analyze authentication protocols. Technical Report TR-96-5508-05, ORA Canada, Ottawa, March 1996. ftp://ora.on.ca/pub/doc/96-5508-05.dvi.Z

[GNY90] Li Gong, Roger Needham, and Raphael Yahalom. Reasoning about belief in cryptographic protocols. In *IEEE Symposium on Research in Security and Privacy*, pages 234–248, Oakland, California, May 1990. IEEE Computer Society, IEEE Computer Society Press.

http://java.sun.com/people/gong/papers/gny-oakland.ps.gz

[Gon90] Li Gong. *Cryptographic Protocols for Distributed Systems*. PhD thesis, University of Cambridge, Cambridge, England, April 1990. http://java.sun.com/people/gong/papers/phd-thesis.ps.gz

[KW96] D. Kindred and J. M. Wing. Fast, automatic checking of security protocols. In *Second USENIX Workshop on Electronic Commerce*, pages 41–52, Oakland, California, November 1996. USENIX. http://www-cgi.cs.cmu.edu/afs/cs.cmu.edu/project/venari/www/usenix96-submit.html

[MCJ97] W. Marrero, E.M. Clarke, and S. Jha. Model checking for security protocols. Technical Report CMU-SCS-97-139, Carnegie Mellon University, May 1997.

[Pau97] Lawrence C. Paulson. Proving properties of security protocols by induction. In *10th Computer Security Foundations Workshop*, pages 70–83. IEEE Computer Society Press, 1997. http://www.cl.cam.ac.uk/users/lcp/papers/protocols.html

[Sch96] Bruce Schneier. *Applied Cryptography*. John Wiley & Sons, 1996.

[Sch97] Johann Schumann. Automatic verification of cryptographic protocols with SETHEO. In *14th International Conference on Automated Deduction (CADE)*, Lecture Notes in Computer Science. Springer-Verlag, 1997.