Université de Montréal

# Le principe des boîtes noires en cryptologie: application et limites

par

Sophie Laplante

Département d'informatique et de recherche opérationnelle

Faculté des arts et des sciences

Mémoire présenté à la Faculté des études supérieures
en vue de l'obtention du grade de
Maître ès sciences (M.Sc.)
en informatique

Juillet 1992

Université de Montréal
Faculté des études supérieures

Ce mémoire intitulé

# Le principe des boîtes noires en cryptologie: applications et limites

Présenté par
Sophie Laplante

a été évalué par un jury composé des personnes suivantes

| | |
|---|---|
| Pierre McKenzie | président-rapporteur |
| Gilles Brassard | directeur de recherche |
| Anindya Das | membre du jury |

Mémoire accepté le _____

# Sommaire

Les systèmes de signature numérique sont un mécanisme par lequel des individus, ayant choisi et publié une information publique, peuvent apposer une signature à un message de façon à ce que quiconque puisse vérifier cette signature basé sur l'information rendue publique.

Une nomenclature des attaques possibles pour contrefaire les systèmes de signature a été proposée dans un article de Golwasser, Micali et Rivest; on définit ici une nouvelle attaque, à la suite de quoi on prouve que cette attaque est *plus forte* que l'attaque la plus forte selon la nomenclature acceptée. Cette attaque consiste à utiliser le dispositif de signature comme une *boîte noire*, en se permettant de sauvegarder les configurations internes du dispositif et replacer le dispositif dans une de ces configurations ultérieurement.

On prouve également que l'un des systèmes connus qui résiste aux attaques les plus fortes de la nomenclature tombe face à cette nouvelle attaque, et on propose une modification à ce système de signature pour qu'il résiste à la nouvelle attaque.

Ces résultats mettent en lumière bon nombre de questions sur d'autres types de protocoles cryptographiques. L'un d'entre eux est les preuves de connaissance, dû à Feige, Fiat et Shamir. Contrairement à l'attaque décrite ci-haut, le principe des boîtes noires est généralement utilisé en cryptologie comme *technique de preuve*. Dans le cas des preuves de connaissance, l'existence d'un *extracteur*, le joueur fictif

qui incarne le principe des boîtes noires, est à même la définition. On se penche donc sur la question suivante: est-il vrai que le concept intuitif de preuve de connaissance se limite aux protocoles pour lesquels on peut démontrer l'existence d'un extracteur, ou existe-t-il des protocoles qui devraient être considérés comme des preuves de connaissances, mais pour lesquels on peut démontrer que les joueurs concernés peuvent être protégés contre les extracteurs?

La réponse à cette question demeure ouverte, mais se sont dégagés de cette recherche des éléments de réponse ainsi que d'autres questions ouvertes nouvelles et intrigantes.

*Sane ita apparebat. Aderant vestigia, modo inter se occursantia, modo intermixta, sed distincte huc et illuc quattuor paria ungularum concurrentium.*

*Milnei*

# Table des matières

# Introduction

La technique de réduction par "boîtes noires" est une technique de preuve très répandue en cryptologie. Dans ce type de preuve, on introduit un joueur fictif dont le rôle est de produire un résultat lorsqu'on lui donne accès à un des joueurs du protocole "dans une boîte noire". Le joueur fictif a le droit de fournir les entrées à la boîte noire, obtenir les sorties, et sauvegarder la configuration actuelle du joueur de façon à remettre le joueur dans cette configuration ultérieurement. L'existence d'un tel joueur fictif sert à démontrer une propriété "intéressante" du protocole.

Dans ce mémoire, une approche toute nouvelle est prise vis-à-vis du principe des boîtes noires. Plutôt que de considérer la manipulation "en boîte noire" de joueurs d'un protocole comme une technique de preuve, on investigue les conséquences qu'une telle manipulation puisse être faite *en pratique* par un joueur malhonnête tentant d'obtenir des informations sensibles. Comment, si c'est possible, les participants à un protocole cryptographique peuvent-ils se prémunir contre une attaque de la sorte?

Le premier chapitre de ce mémoire est consacré à la présentation de concepts cryptologiques qui seront utiles dans les chapitres subséquents. On y couvre les hypothèses cryptographiques, les preuves interactives, ainsi que les systèmes de camouflage.

Le second chapitre porte sur les *signatures numériques*. On y présente un nouveau type d'attaque, strictement plus forte que les attaques reconnues jusqu'ici comme étant les plus fortes possibles. Cette attaque est basée sur le principe des boîtes noires. Plus spécifiquement, on donne dans ce chapitre la définition de *l'attaque à l'extracteur*; on donne la preuve que cette attaque est strictement plus forte que l'attaque adaptive à message choisi; finalement, on exhibe un système de signature qui résiste à ce nouveau type d'attaque.

Dans le dernier chapitre de ce mémoire, on envisage la question suivante: les joueurs de d'autres types de protocoles cryptographiques peuvent-ils se protéger contre les attaques de type "extracteur"? On examine en particulier le cas des *preuves de connaissance*, en exposant un apparent paradoxe et explorant les ramifications d'une éventuelle solution.

Si la principale contribution de ce mémoire demeure la définition de cette nouvelle attaque, il n'en demeure pas moins que les questions qui restent ouvertes sont d'un intérêt appréciable. Certaines d'entre elles sont présentées en conclusion.

# Chapitre 1

# Concepts de base

Le but de ce chapitre est d'introduire quelques principes de base de la cryptologie, qui seront utilisés au cours de ce mémoire. On introduit ainsi le concept d'*hypothèse cryptographique*, de *preuves interactives* et de *systèmes de camouflage*.

## 1.1 Les hypothèses cryptographiques

Plusieurs des constructions en cryptologie moderne reposent sur des hypothèses encore non-prouvées sur la complexité intrinsèque de certaines fonctions. De plus, il a été démontré pour certains types de protocoles qu'ils ne peuvent exister que si l'une de ces hypothèses se vérifie. C'est le cas des signatures numériques dont il sera question au chapitre 2.

Certaines hypothèses sont plus *faibles* que d'autres, dans le sens où elles sont "plus probables". On discutera de l'hiérarchie des hypothèses les plus courantes par rapport à cet ordonnancement à la section 1.1.3.

### 1.1.1 Fonctions à sens unique

Une des hypothèses les plus courantes est l'existence des fonctions à sens unique. Une fonction à sens unique est une fonction facile à calculer, mais dont il est impossible de calculer l'inverse en temps polynomial. La définition suivante formalise ce concept:

**1.1 Définition** *Une fonction* $f : \mathbb{N} \to \mathbb{N}$ *est une* fonction à sens unique *si*

1. *Il existe un algorithme* $A(n)$ *fonctionnant en temps polynomial tel que* $A(n)$ *calcule correctement* $f(n)$ *pour tout* $n \in \mathbb{N}$.

2. *Il n'existe aucun algorithme probabiliste polynomial* $A(n,r)$ *pour lequel il existe une constante* $c > 0$ *tel que* $(\forall x)\, Prob[A(n,r)=y \text{ et } f(y)=x] > 1-1/|x|^c$, *où la probabilité est prise sur la suite de choix aléatoires* $r$ *de l'algorithme* $A$ *et sur le choix (uniforme) de* $x$.

Un cas particulier des fonctions à sens unique est les *permutations à sens unique*, qui se définit comme les fonctions à sens unique mais où la fonction $f$ est restreinte à être une permutation. Il est clair que l'existence de permutations à sens unique implique l'existence de fonctions à sens unique, mais il n'y a aucune raison de croire que la réciproque soit vraie.

Le candidat le plus souvent cité en exemple est celui de la multiplication d'entiers. En effet, on sait comment multiplier efficacement des entiers de grande taille, mais il n'est pas encore d'algorithme qui puisse *inverser* cette opération efficacement, c'est-à-dire, étant donné un entier de grande taille, en trouver deux facteurs non-triviaux en temps polynomial. L'existence même de fonctions à sens unique n'a pas été démontrée, donc, comme pour tous les candidats au titre de fonction

à sens unique, il se pourrait bien qu'il existe un algorithme de factorisation qui fonctionne en temps polynomial.

### 1.1.2 Fonctions à brèche secrète

Une *fonction à brèche secrète*, comme les fonctions à sens unique, est facile à calculer, mais son inverse est difficile à calculer *en l'absence d'une information secrète*. En général, on parle de *familles de fonctions à brèche secrète*, où un intervenant peut choisir au hasard une "brèche secrète" et construire une fonction à brèche secrète correspondant à cette information, lui permettant ainsi d'inverser cette fonction tout en sachant que calculer l'inverse sera difficile pour qui que ce soit d'autre.

**1.2 Définition** *un* générateur de fonctions à brèche secrète *est un triplet d' algorithmes (probabilistes) polynomiaux* $(G(\cdot), E(\cdot), I(\cdot))$ *tel que*

1. $G(1^k)$ *produit une paire* $(x, y)$, $|x| = |y| = k$[1]*;*
2. *les algorithmes* $E(x, \cdot)$ *("évaluation") et* $I(y, \cdot)$ *("inverse") sont tels que* $I$ *trouve un inverse de* $E$*, c'est-à-dire* $(\forall z \in \{0,1\}^k) E(x, I(y, z)) = z$*.*
3. *il n'existe aucun algorithme* $A(\cdot, \cdot, \cdot, \cdot)$ *qui puisse calculer une fonction "inverse" de* $E(x, \cdot)$ *c'est-à-dire pour lequel il existe une constante* $c > 0$ *tel que* $(\overset{\infty}{\forall}\ k)(\forall c > 0) Prob[E(x, A(1^k, x, z, r) = z] > 1 - 1/|x|^c$ *lorsque* $x$ *est choisi uniformément au hasard et* $r$ *est la suite de choix aléatoires faits par* $A$ *et* $(x, y)$ *est obtenu en exécutant* $G(1^k)$*.*

[1] Il est sous-entendu que cet algorithme est *nécessairement* probabiliste, sans quoi il n'y aurait dans les faits qu'une seule fonction dans la famille.

Comme pour les permutations à sens unique, on définit les *permutations à brèche secrète* comme étant des fonctions à brèche secrète restreintes à être des permutations.

Un exemple de fonction que l'on croit être à brèche secrète est l'exponentiation modulo le produit de deux grands nombres premiers. L'exponentiation se fait aisément, mais il apparaît impossible de trouver le logarithme discret en temps polynomial sans connaître la factorisation du module. (Une fois la factorisation du module connue, par contre, l'extraction du logarithme discret se fait facilement en utilisant le théorème chinois du résidu.)

### 1.1.3 Hiérarchie des hypothèses

La sécurité de plusieurs protocoles cryptographiques repose sur l'admission de l'une des hypothèses ci-haut, ou même sur l'hypothèse plus spécifique qu'un candidat particulier est une fonction à brèche secrète, ou une permutation à sens unique, etc.

Un des objectifs des chercheurs est de tenter de minimiser ces hypothèses, c'est-à-dire de faire reposer la preuve de sécurité des protocoles sur l'hypothèse "la plus vraisemblable". On a également isolé pour certains types de protocoles l'hypothèse minimale pour l'existence de tels protocoles. Par exemple, comme on le verra au chapitre 2, il a été démontré que l'existence de signatures numériques invulnérables est *équivalent* à l'existence de fonctions à sens unique [R90].

L'ordonnancement des hypothèses décrites ici, de la plus forte à la plus faible est: existence de permutations à brèche secrète, existence de fonctions à brèche secrète, existence de permutations à sens unique et finalement, existence de fonctions à sens unique.

## 1.2 Les preuves interactives

### 1.2.1 But

Le concept de "preuves interactives" est un concept encore récent en cryptologie: introduit en 1985 par Goldwasser, Micali et Rackoff [GMR89], il a été à l'origine d'une branche faisant encore l'objet de nombreuses recherches.

Les preuves interactives sont des protocoles cryptographiques dans lesquels un prouveur tente de convaincre un vérificateur de la validité d'un énoncé. Plusieurs variantes ont été proposées, et seront discutées plus loin.

#### Extension de la classe $\mathcal{NP}$

Un des intérêts de la classe des langages admettant une preuve interactive est qu'elle représente une généralisation intéressante du concept derrière la classe $\mathcal{NP}$: celui des preuves efficaces. En effet, la classe $\mathcal{NP}$ peut se définir ainsi:

**1.3 Définition** $\mathcal{NP}$ *est la classe des langages admettant un système de preuves vérifiables en temps polynomial.*

Dans le contexte de cette définition, il suffit pour un prouveur de soumettre un témoin du fait que $x \in L$ , ce qui suffirait pour convaincre n'importe quel vérificateur polynomial, déterministe, de la validité de cet énoncé.

Les preuves interactives permettent de repousser plus loin les limites de cette classe: en admettant

1. que le vérificateur et le prouveur soient *probabilistes*, et par le fait même, en admettant une probabilité d'erreur (qui doit pouvoir être aussi petite qu'on veut) dans la conclusion du vérificateur, et

2. l'échange de messages entre le prouveur et le vérificateur,

on réussit en effet à englober des langages qui ne sont vraisemblablement pas dans $\mathcal{NP}$[2].

**Outil cryptologique**

De façon générale, la cryptologie est l'étude des protocoles permettant à deux ou plusieurs intervenants d'échanger ou de manipuler de l'information, qu'elle soit sous forme de message ou non, et de façon à ce qu'on puisse enforcer certaines contraintes, par exemple sur l'honnêteté des intervenants, ou la confidentialité de l'information qui circule.

Dans le cadre des preuves interactives, l'information faisant l'objet de l'échange est l'appartenance ou non d'une phrase à un langage. Le type de sécurité des deux joueurs dont on peut se préoccuper est l'assurance que le prouveur ne pourra pas démontrer un théorème faux à un vérificateur honnête, ou encore que le vérificateur n'apprendra rien d'autre que la conviction que le théorème est correct.

### 1.2.2 Modèles

Plusieurs modèles ont été proposés pour formaliser le concept de preuves interactives. Les principales différences se situent au niveau de la puissance de calcul des intervenants. On présente dans cette section les deux modèles principaux, soit le modèle original de GMR, ainsi que celui de BCC.

[2]Il est démontré dans [S90] que la classe des langages admettant une preuve interactive est en fait égale à $\mathcal{P}$SPACE, c'est-à-dire la classe des langages pouvant être reconnus en *espace* polynomial.

### Le modèle GMR

Un des premiers modèles formels de preuves interactives comme on les connaît aujourd'hui a été proposé par Goldwasser, Micali et Rackoff [GMR89][3] . Dans ce modèle, le prouveur et le vérificateur ($P$ et $V$, respectivement) sont des "machines de Turing interactives", c'est-à-dire des machines de Turing munies chacune d'un ruban privé et d'un ruban aléatoire, et partageant deux rubans de communication, l'un accessible en écriture par le prouveur et en lecture par le vérificateur, et l'autre, symétriquement, accessible en écriture par le vérificateur et en lecture par le prouveur. On exige dans ce modèle que le vérificateur puisse exécuter sa part du protocole en temps polynomial, mais aucune contrainte n'est imposée sur la puissance du prouveur.

Le but d'une preuve interactive pour un langage $L \subseteq \Sigma^\star$, où $\Sigma = \{0,1\}$ est pour le prouveur de convaincre le vérificateur qu'un $x$ donné appartient à $L$. Le vérificateur acceptera la preuve si la probabilité qu'un prouveur malhonnête réussisse à le convaincre est au plus $1/p(|x|)$, pour tout polynôme $p$ et pour $x$ assez grand.

Un protocole spécifié à l'avance encadre le déroulement de la preuve; celui-ci se fera en un certain nombre d'échanges de messages entre prouveur et vérificateur. À la dernière étape, le vérificateur doit décider s'il accepte ou rejette la preuve.

Plus formellement, on a :

**1.4 Définition** *Une* machine de Turing interactive *est une machine de Turing munie de*

- *un ruban de calcul*

[3]Ce modèle a en fait été précédé par celui de [BM88] dont l'objectif était de capturer la notion de preuve du point de vue de la complexité des langages.

- *un ruban aléatoire*
- *un ruban de communication accessible seulement en lecture*
- *un ruban de communication accessible seulement en écriture.*

**1.5 Définition** *Une paire de machines de Turing interactives* $(P,V)$ *est une* preuve interactive *pour un langage* $L \subseteq \Sigma^*$ *si pour tout polynôme* $p(\cdot)$*, pour toute machine interactive* $\widetilde{P}$[4]

- $P$ *est une machine de Turing interactive à puissance de calcul illimitée*
- $V$ *est une machine de Turing interactive limitée à un temps de calcul polynomial, excluant le temps de calcul pris par* $P$*.*
- $x \in L \Longrightarrow Prob[V \text{ accepte la preuve de } P] \geq 1 - 1/p(|x|)$
- $x \notin L \Longrightarrow Prob[V \text{ accepte la preuve de } \widetilde{P}] < 1/p(|x|)$

### Le modèle BCC

Le second modèle de preuves interactives présenté ici est celui de Brassard, Chaum et Crépeau [BCC88]. Deux caractéristiques le distinguent du modèle de [GMR89]. La première différence se situe au niveau de la puissance de calcul permise au prouveur: alors que [GMR89] n'impose aucune contrainte sur celle-ci, le modèle [BCC88] restreint la puissance de calcul du prouveur à être polynomiale. Cette différence a deux principales conséquences:

- Certaines preuves interactives (au sens de [GMR89]) peuvent exiger du prouveur d'exécuter des calculs irréalisables en temps polynomial. Ces protocoles ne sont pas admis dans le contexte de [BCC88].

[4] $\widetilde{P}$ représente n'importe quel prouveur tricheur

- Le fait de pouvoir supposer que certains calculs ne sont pas réalisables par le prouveur fait en sorte que la sécurité du vérificateur est plus facile à assurer. Considérons, par exemple, le cas d'un protocole pour lequel on peut démontrer que tout prouveur qui réussit à convaincre le vérificateur d'un énoncé faux est également en mesure de résoudre un problème supposé difficile (disons $\mathcal{NP}$-ardu) en temps polynomial. Un tel protocole est acceptable dans le cadre de la définition de BCC, alors qu'il ne le sera pas nécessairement dans celui de GMR.

Une seconde différence, peut-être moins fondamentale au niveau des conséquences qu'elle entraîne, est que dans le modèle de [BCC88], la sécurité du prouveur ne dépend pas de la longueur de l'exemplaire faisant l'objet de la preuve. En effet, dans [BCC88], on n'exige seulement que pour tout $k$

- $x \in L \Longrightarrow$ Prob[$V$ accepte la preuve de $P$ à la fin du protocole] $\geq 1 - 1/2^k$
- $x \notin L \Longrightarrow$ Prob[V accepte la preuve de $\widetilde{p}$ à la fin du protocole] $< 1/2^k$

Cette différence dans la définition est plutôt une question "philosophique", étant donné qu'elle n'affecte pas la classe des langages admettant une preuve interactive. En effet, il est clair que tout protocole avec sécurité $1/p(|x|)$ répond également à la seconde définition; par ailleurs, pour augmenter la sécurité d'un protocole avec sécurité $1/2^k$, il suffit de répéter le protocole un certain nombre de fois et prendre la conclusion majoritaire.

Toutefois, même si cette distinction semble sans importance, il ne semble pas y avoir de raison pour ne pas adopter la définition la plus faible, puisque la plus forte exige des protocoles qu'ils soient plus sécuritaires pour les exemplaires plus longs, alors que la sécurité devrait pouvoir être un paramètre choisi indépendamment de la longueur de l'exemplaire traité.

Afin de bien distinguer les preuves du modèle de [GMR89] de celles de [BCC88], introduisons la terminologie suivante:

**Notation:** on appelle preuve interactive statistiquement convaincante les preuves du modèle de GMR, alors qu'on nomme celles du modèle BCC preuves interactives calculatoirement convaincantes.

## 1.3 Les systèmes de camouflage

### 1.3.1 But

Un outil de base, ou "primitive", utilisé dans plusieurs protocoles cryptographiques est le système de camouflage. Brièvement, il s'agit d'un protocole par lequel un joueur s'engage à la valeur d'un bit sans le révéler immédiatement à l'autre joueur, mais de façon telle que l'autre joueur est convaincu que le premier ne pourra pas changer d'avis sur la valeur de ce bit.

De façon plus anecdotique, on peut imaginer le protocole comme se déroulant ainsi: le premier joueur choisit la valeur d'un bit, et place ce bit dans une enveloppe scellée qu'il met en pleine vue des deux joueurs. Au moment ou il veut révéler la valeur de ce bit, il permet à l'autre joueur d'ouvrir l'enveloppe.

En pratique, les protocoles qui implantent cette notion procéderont en deux étapes, comme ci-haut. Dans la première étape, le premier joueur choisit la valeur de son bit et calcule une valeur (un entier) qu'il envoie au deuxième joueur. Cette valeur se nomme *camouflage* et on dit qu'à cette étape le premier joueur *s'engage* à la valeur de ce bit ou qu'il *met en gage* ou qu'il *met en consigne* ce bit. Dans un deuxième temps, le premier joueur donnera une information supplémentaire, appelée *témoin*, au deuxième joueur qui lui permettra de révéler la valeur du bit

choisie. Le témoin est généralement une chaîne aléatoire qui a servi pour produire le camouflage. Cette étape s'appelle *ouvrir le camouflage*.

### 1.3.2 Caractéristiques

On doit assurer deux propriétés de base pour qu'un tel protocole fonctionne:

- les camouflages doivent être *liants*: il doit être impossible, ou tout au moins irréalisable pour le premier joueur de changer d'avis sur la valeur du bit qu'il a mis en consigne.
- les camouflages doivent être *camouflants*: il doit être impossible, ou encore, comme ci-haut, irréalisable pour le *second* joueur de déduire quelqu'information que ce soit sur la valeur du bit camouflé avant que celle-ci ne soit révélée.

Quand il est impossible, indépendamment de sa puissance de calcul, pour le premier joueur de changer d'avis sur la valeur du bit mis en consigne, on dit du camouflage qu'il est *inconditionnellement liant*. Sinon, on dit qu'il est *calculatoirement liant*, entendant par là qu'il est impossible pour le premier joueur de tricher s'il est limité à un temps de calcul polynomial. De façon analogue, les camouflages peuvent être *inconditionnellement camouflants* ou *calculatoirement camouflants*.

Il est important de noter qu'un système de camouflage ne peut pas produire des camouflages à la fois inconditionnellement liants et inconditionnellement camouflants. Si un système de camouflage donne lieu à des camouflages inconditionnellement liants, c'est que le choix de 0 donne lieu à un ensemble de camouflages disjoint de celui résultant du choix de 1: sans cela il serait possible, avec une puissance de calcul possiblement illimitée, que le premier joueur trouve un camouflage pouvant

à la fois être ouvert comme un 1 et un 0. Or si ces ensembles sont disjoints, il est envisageable que le deuxième joueur, muni d'une puissance de calcul illimitée, puisse distinguer les camouflages représentant un 1 de ceux représentant un 0.

# Chapitre 2

# Les signatures

Introduit par Diffie et Hellman en 1976 [DH76], le concept de signatures numériques est à l'origine de ce qu'on considère maintenant être la cryptologie moderne, celle qui ne requiert pas d'échange secret avant le début des interactions entre les partis impliqués. Par conséquent, les signatures ont bénéficié de beaucoup d'attention de la part des chercheurs.

Malgré cette attention, le problème de trouver un système de signature qui soit "le plus sécuritaire possible" est resté ouvert plusieurs années, d'autant plus qu'on croyait que ceci était impossible à réaliser.

Cette section présente une définition des signatures ainsi qu'un aperçu de ce "paradoxe" et sa solution. On présente ensuite un nouveau type d'attaque pouvant être montée par un faussaire, plus puissante que toutes les autres ayant été proposées, ainsi qu'un système de signature résistant à cette attaque[1].

[1] Une partie de ce travail a été réalisé en collaboration avec Mihir Bellare (alors étudiant au M.I.T.)

## 2.1 Définition du concept

### 2.1.1 But

Le but d'une signature numérique est pour un intervenant, que l'on appellera *signataire* de certifier qu'il est le destinataire d'un message. Deux des caractéristiques souhaitables pour un système de signature sont

- que le signataire ne puisse pas renier sa signature par la suite, et
- qu'il ne soit pas possible pour un "ennemi" de produire des signatures contrefaites, sauf peut-être avec probabilité négligeable.

Ces caractéristiques seront précisées et discutées dans les sections qui suivent.

Loin d'être purement d'intérêt théorique, les signatures numériques revêtent une grande utilité pratique. Avec l'utilisation grandissante de moyens de communication électroniques, il devient de plus en plus facile de faire parvenir des messages dont l'origine est falsifiée. Un système de signature numérique permettrait d'identifier avec certitude l'origine des documents électroniques.

Les signatures peuvent être utilisées dans plusieurs contextes:

- Elles peuvent être envoyées d'un usager à un autre usager, ou à un ensemble d'individus prédéfini, de façon à ce que seuls les membres de ce groupe puissent vérifier la signature.
- Elles peuvent être *vérifiables publiquement*, c'est-à-dire que n'importe qui peut vérifier la validité de la signature.

Dans le premier cas, il faut que les usagers du système échangent (ou partagent) de information au préalable, alors que dans le deuxième cas, il suffit pour un

usager d'avoir accès à une information rendue publique par le signataire (sa *clef publique*) pour pouvoir vérifier une signature. Les sections qui suivent concernent ce deuxième contexte.

### 2.1.2 Définition formelle

Formellement, [GMR88, § 2.1] définit les systèmes de signature comme suit:

**2.1 Définition** *Un* système de signature vérifiable publiquement *est un sextuplet* $\langle k, \mathcal{M}, B, G, \sigma, V \rangle$ *tel que*

- $k$ *est un paramètre de sécurité choisi par le signataire, qui représente le degré de sécurité qu'il exige pour ses signatures (un faussaire polynomial aura une probabilité d'au plus* $1/2^k$ *de produire une fausse signature).*
- $\mathcal{M}$ *est l'espace des messages dans lequel sont choisis les messages à signer (en général,* $\mathcal{M} \subseteq \Sigma^+$*).*
- $B$ *est le nombre maximal de messages pouvant être signés avec une clef donnée.* $B$ *peut dépendre de la valeur de* $k$*.*
- $G$ *est l'algorithme*[2] *de génération de clefs, i.e. un appel à* $G(1^k)$ *produit une paire* $(K_P, K_S)$*, où* $K_P$ *est la clef secrète du signataire, et* $K_S$ *est la clef publique associée à* $K_P$*.*
- $\sigma$ *est l'algorithme de signature, i.e.* $\sigma(m, K_P)$ *produit la signature du message* $m \in \mathcal{M}$ *à l'aide de la clef* $K_P$*. Dans certains cas, une signature peut dépendre de d'autres paramètres, comme la signature du message précédent, ou le nombre de messages déjà signés.*

[2] Tous les algorithmes dans cette définition doivent fonctionner en temps polynomial, et peuvent être probabilistes

- *$V$ est l'algorithme de vérification de signatures, i.e. $V(S, m, K_P)$ retourne "vrai" si et seulement si $S$ est une signature valide pour le message $m$ étant donné la clef publique $K_P$.*

### 2.1.3 Types et niveaux d'attaque

Le degré de sécurité que l'on peut exiger d'un système de signature est très variable. On peut vouloir se protéger de différents moyens d'attaque, et on peut aussi vouloir se prémunir contre différents types de contrefaçon. Les types et niveaux d'attaque habituellement cités dans la littérature sont décrits dans les deux sections qui suivent. (Cette classification est due à [GMR88, §§ 2.2, 2.3].)

#### Types d'attaque

Un type d'attaque décrit les "outils" à la disposition du faussaire[3]. On les classifie de la façon suivante, en ordre croissant de potentiel de dommage:

- *attaque à clef publique seulement:* le faussaire n'a à sa disposition que la clef publique du signataire.
- *attaque à message connu:* en plus de la clef publique, le faussaire dispose d'un certain nombre de paires message–signature.
- *attaque à message choisi fixe:* le faussaire a la possibilité d'obtenir un certain nombre de paires message–signature, pour un ensemble de messages choisis indépendamment de la clef publique du signataire. À noter que le choix des messages doit se faire avant le début de l'attaque (e.g. le choix du 6 $^{\text{ième}}$

[3]On suppose toujours que le faussaire a une puissance de calcul probabiliste, polynomiale.

message ne peut pas être fonction des signatures obtenues pour les 5 messages précédents).

- *attaque à message choisi quelconque:* le faussaire peut obtenir la signature des messages de son choix, cette fois-ci en fonction de la clef publique du signataire. Comme l'attaque à message choisi fixe, le choix des messages est fait avant le début de l'attaque.

- *attaque adaptive à message choisi:* cette attaque est la même que la précédente, sauf qu'ici, le choix des messages peut se faire en fonction des signatures des messages précédents.

### Niveaux d'attaque

Le niveau d'une attaque spécifie le degré de "réussite" du faussaire. Elles sont présentées ici en ordre décroissant de succès:

- *bris total:* le faussaire trouve la clef secrète du signataire.

- *contrefaçon universelle:* le faussaire trouve, à défaut de la clef secrète elle-même, un algorithme permettant de contrefaire n'importe quelle signature.

- *contrefaçon sélective:* le faussaire peut contrefaire une signature *de son choix.*

- *contrefaçon existentielle:* le faussaire réussit à produire une signature, sans aucun contrôle sur le message correspondant.

## 2.2 Signatures invulnérables aux attaques adaptives à message choisi

Comme il s'agit des systèmes jusqu'à maintenant considérés les plus robustes, les systèmes de signature invulnérables aux attaques adaptives à message choisi ont bénéficié d'énormément d'attention de la part de la communauté de chercheurs en cryptologie. Ceci est vrai d'autant plus qu'il a été longtemps cru qu'il était impossible de trouver un tel système pour lequel on pourrait démontrer que la contrefaçon est un problème aussi difficile que, par exemple, la factorisation. Il sera question, dans cette section, de ce "paradoxe" apparent, ainsi que des solutions qui ont été proposées pour le résoudre.

### 2.2.1 Le "paradoxe"

La grande difficulté de trouver un système de signature invulnérable aux attaques adaptives à message choisi est due au fait que, dans la plupart des systèmes proposés (avant 1984), une preuve de sécurité contre les attaques (non-adaptives) à message choisi donnait lieu de façon immédiate à un algorithme pour briser le système avec une attaque adaptive.

**2.1 Théorème (Folklorique)** *Considérons un système de signature dont on peut démontrer qu'il est invulnérable (au moins) aux attaques à clef publique seulement, dans le sens que briser le système est équivalent à briser une hypothèse cryptographique sur laquelle repose la sécurité de la clef secrète du signataire. Alors ce système de signature peut être brisé (en contrefaçon universelle) par une attaque adaptive à message choisi.*

**Preuve:** ([Wi80]) Considérons un système de signature pour lequel on a démontré que la contrefaçon est équivalente à la factorisation[4]. En admettant que la preuve procède de façon constructive et donne un algorithme de factorisation basé sur un algorithme de contrefaçon $\mathcal{A}$, c'est-à-dire que la preuve est basée sur le principe des boîtes noires, on monte une attaque adaptive de la façon suivante: le faussaire exécute l'algorithme de factorisation tel que donné par la preuve, en exécutant $\mathcal{A}$ au besoin. Il est clair que l'attaque résultante est une attaque adaptive à message choisi. Ainsi, à l'aide de cet algorithme de factorisation, le faussaire peut recouvrer la clef secrète du signataire, lui permettant de signer n'importe quel message (contrefaçon universelle).

□

**2.2 Théorème** *Le théorème 2.1 est faux.*

La principale erreur dans le raisonnement de la "preuve" du théorème 2.1 est qu'il néglige le fait que l'algorithme de contrefaçon $\mathcal{A}$ doit être uniforme, c'est-à-dire qu'il doit pouvoir contrefaire une signature *quelle que soit la clef publique du signataire*[5]. Or l'attaque (adaptive) du faussaire sera impossible si, par exemple, l'algorithme de factorisation donné par la preuve requiert que l'on applique $\mathcal{A}$ à des signataires *différents*.

Ce fait est d'ailleurs au centre de la preuve de sécurité de tous les systèmes de signature de la section 2.2.2 qui ont été proposés pour résoudre le "paradoxe". La section suivante donne un aperçu de ces systèmes.

[4]Cet argument tient quelle que soit l'hypothèse cryptographique sur laquelle repose la sécurité de la clef secrète du signataire. On parle ici de factorisation dans le seul but de simplifier la présentation de l'argument.

[5]Sans cette condition, tout algorithme qui "connaît" à priori une clef secrète fixe peut évidemment contrefaire des signatures correspondant à cette clef.

### 2.2.2 Solutions proposées

L'article de Goldwasser, Micali et Rivest [GMR88] présente le premier système de signature invulnérable aux attaques adaptives à message choisi. Cette solution a toutefois été suivie de systèmes plus simples et plus pratiques, et celle à laquelle on s'attardera davantage, surtout à cause de sa simplicité, est celle de Bellare et Micali [BM88]. Parmi les autres solutions qui ont suivi, notons celles de Bellare et Goldwasser [BG89], Naor et Yung [NY89], Brassard [B89] ainsi que celle de Rompel [R90].

#### Le système de Bellare-Micali

Le système de signature proposé par Bellare et Micali [BM88] retient de [GMR88] les idées principales, mais les applique ici d'une manière beaucoup plus simple, donnant lieu à un système nettement plus efficace que le premier. On le décrit ici en plus de détail.

L'idée de base qui sous-tend ce système provient de [La79]. On y suggère, pour signer un bit, de choisir et publier une fonction à brèche secrète $f$ ainsi que deux points $x_0$ et $x_1$ de l'image de $f$, et de garder comme clef secrète $f^{-1}$. Il suffit donc, pour signer le bit 0, de donner $f^{-1}(x_0)$, ou, pour signer le bit 1, de fournir $f^{-1}(x_1)$. Par extension, si on veut signer un message de $l$ bits, il faudrait publier, en plus de $f$, $l$ paires de points $x_0^i, x_1^i$ $(1 \leq i \leq l)$. Comme la sécurité du système repose sur le fait que le faussaire ne dispose d'aucune paire $(f^{-1}(x_0^i), f^{-1}(x_1^i))$, le nombre de bits qui peuvent être signés doit être fixé au moment de publier la clef publique. L'article de [BM88] montre comment généraliser cette approche pour pouvoir signer un nombre illimité de messages avec une seule clef publique de longueur fixe.

L'idée est la suivante: le signataire publie une fonction $f$ ainsi que $k+1$ paires de points de l'image de $f$. Pour signer un bit $b$, il envoie $f^{-1}(x_b^1)$), choisit une nouvelle fonction $f'$ (en faisant un appel à $G(1^k)$) et avec les $k$ paires qui restent, signe chacun des bits de la représentation de $f'$. Pour signer le bit suivant du message, on procède de la même façon, avec la nouvelle fonction $f'$ jouant le rôle de $f$. Une signature complète d'un message est donc constituée d'un inverse pour chaque bit du message, ainsi que *tous les inverses correspondant à toutes les nouvelles fonctions qui ont été utilisées depuis que le système est en opération.*

### Quelques autres systèmes

Pour compléter le tableau, mentionnons trois autres solutions apportées au problème des signatures invulnérables aux attaques adaptives à message choisi. Bellare et Goldwasser [BG89] proposent une approche basée sur l'utilisation d'un autre type de protocole cryptographique, les preuves non-interactives à divulgation nulle ("Non-Interactive Zero-Knowledge") Utilisé avec la construction de protocoles "NIZK" de Feige, Lapidot et Shamir [FLS90], on obtient un système de signature avec les propriétés désirées.

Naor et Yung [NY89] ont démontré que les permutations à sens unique sont suffisantes pour obtenir des signatures invulnérables aux attaques adaptives à message choisi en utilisant des fonctions d'adressage dispersé à sens unique.

Finalement, Rompel [R90] a démontré que l'existence de fonctions à sens unique est nécessaire et suffisante pour l'existence de systèmes de signature invulnérable aux attaques adaptives à message choisi.

Le tableau qui suit résume les hypothèses cryptographiques nécessaires pour chacun de ces systèmes:

| Système de signature | Hypothèse cryptographique |
|---|---|
| [GMR88] | Permutations à brèche secrète sans griffe |
| [BM88] | Permutations à brèche secrète |
| [NY89] | Permutations à sens unique |
| [BG89+FLS90] | (Dépend des hypothèses pour le NIZK) |
| [R90] | Fonctions à sens unique (necessaire et suffisant) |

## 2.3 Un nouveau type d'attaque

Malgré que l'attaque adaptive à message choisi ait été longtemps considérée comme l'attaque naturelle la plus forte qui soit[6], on présente ici un nouveau type d'attaque strictement plus fort que l'attaque adaptive à message choisi, tout en étant réalisable en pratique. On exhibera également un système de signature pouvant résister à ce nouveau type d'attaque.

### 2.3.1 L'attaque à l'extracteur

L'attaque à l'extracteur est inspirée de la définition de *preuves de connaissance* de [FFS88] (décrites au chapitre 3). Pour monter une telle attaque, il suffit au faussaire d'avoir accès au mécanisme de signature en tant que *boîte noire*. Les opérations permises au faussaire sur la boîte noire sont

1. fournir les données d'entrée à la boîte noire (dans ce cas-ci, les messages à signer)

[6] "(...) the adaptive chosen-message attack [is] the most severe natural attack an enemy can mount" [GMR88];
"This notion of security, which was introduced in ([GMR88]), represents the strongest possible natural notion of security for a digital signature scheme" [BG89]

2. obtenir les sorties correspondantes (les signatures)

3. prendre un instantané de la configuration interne de la boîte noire (sans pouvoir en analyser le contenu). Ceci inclut les rubans de calcul et le ruban aléatoire.

4. remettre la boîte noire dans un des états sauvegardés au préalable

5. changer le contenu du ruban aléatoire.[7]

Il est clair que tout système de signature pouvant résister à une attaque à l'extracteur est aussi invulnérable aux attaques adaptives à message choisi; en fait, l'attaque adaptive à message choisi correspond à une attaque à l'extracteur où le faussaire ne fait usage que des deux premières opérations.

On démontrera à la section 2.3.2 que l'ensemble des systèmes de signature invulnérables aux attaques à l'extracteur est un sous-ensemble *strict, non-vide* de l'ensemble des systèmes invulnérables aux attaques adaptives à message choisi. Il s'agit donc d'un type d'attaque *strictement plus fort* que tout autre ayant été proposé dans la littérature.

De plus, cette attaque pourrait être montée *en pratique*. Plusieurs ordinateurs couramment disponibles sur le marché aujourd'hui, notamment les micro-ordinateurs portatifs, disposent d'un système qui enregistre la configuration interne de la machine (par exemple, après une période d'inactivité de 10 minutes) et qui reprennent les opérations après avoir remis l'ordinateur dans la configuration sauvegardée.

[7] Il ne semble pas que l'ajout de cette opération change le potentiel de l'attaque, mais elle a été retenue pour être le plus général possible et pour des raisons d'unité avec le modèle de [FFS88].

Il suffirait donc, pour monter une telle attaque sur un système de signature, que le dispositif de signature soit muni d'un mécanisme semblable. Le faussaire pourra donc, en plus de demander des signatures de messages de son choix, provoquer une sauvegarde de la configuration, copier cette information, et remettre le dispositif dans une de ces configurations au besoin.

### 2.3.2 L'attaque à l'extracteur est non-triviale

L'attaque à l'extracteur n'a bien entendu d'intérêt que s'il est démontré qu'elle n'est pas équivalente à un autre type d'attaque déjà connu. On montrera donc que le système de Bellare-Micali, tout en étant invulnérable aux attaques adaptives à message choisi, tombe très facilement lorsque soumis à une attaque à l'extracteur. De plus, on exhibe un système de signature, basé d'ailleurs sur celui de Bellare-Micali, et qui résiste aux attaques à l'extracteur.

#### L'attaque à l'extracteur est strictement plus forte que l'attaque adaptive

La section 2.2.2 présentait les principaux systèmes de signature qui résiste à une attaque adaptive à message choisi. Il est intéressant, toutefois, de remarquer que chacun de ces systèmes est très facile à contrefaire à l'aide d'une attaque à l'extracteur.

Considérons le cas du système de Bellare-Micali, décrit à la section 2.2.2. Notons d'abord qu'il suffit, pour contrefaire un message de longueur $l$, de connaître la signature de 0 et de 1 en "positions" $1, 2, \ldots, l$. Cette information est facile à obtenir lorsqu'on a la possibilité de faire revenir en arrière le mécanisme de signature: il suffit de faire signer le message $\underbrace{000\cdots0}_{l}$, de remettre le mécanisme de signature dans son état initial, puis de faire signer le message $\underbrace{111\cdots1}_{l}$. Cela

fait, le faussaire peut signer n'importe quel message de longueur $l$.

### Un système de signature qui résiste aux attaques à l'extracteur

Un outil indispensable pour les constructions de cette section est le concept de familles de fonctions polynomialement aléatoires présenté dans [GGM84]. Intuitivement, une famille de fonctions polynomialement aléatoire est une classe de fonctions indicées de façon à ce que l'indice de la fonction suffise pour construire un algorithme efficace qui calcule la fonction correspondante, mais qu'il soit impossible de distinguer un membre choisi aléatoirement de cette famille d'une fonction réellement aléatoire.

**2.2 Définition** *[GGM84] Deux ensembles* $X = \bigcup_{k=1}^{\infty} X_k$ *et* $Y = \bigcup_{k=1}^{\infty} Y_k$ *sont* polynomialement indistingables *si pour tout algorithme probabiliste polynomial* $A$ *admettant un oracle, pour tout polynôme* $p$ *et pour* $k$ *suffisamment grand,*

$$Prob[A^{X_k} \text{ retourne } 0] \text{ - } Prob[A^{Y_k} \text{ retourne } 1] < 1/p(k).^{8}$$

**2.3 Définition** *Soit* $\Sigma = \{0,1\}$*; soit* $F_k \subseteq \{f : \Sigma^k \rightarrow \Sigma^k\}$ *pour chaque* $k \in \mathbb{N}$*. Alors* $F = \bigcup_{k=1}^{\infty} F_k$ *est une* famille de fonctions polynomialement aléatoire *si*

- *pour chaque* $k$*, à chaque* $f \in F_k$ *est associé un indice (unique) de* $k$ *bits;*
- *pour chaque* $k$*, il existe un algorithme polynomial, déterministe qui, étant donné un indice de fonction* $i$ *et une valeur* $x \in \Sigma^k$ *retourne* $f_i(x)$*;*
- *les ensembles* $\{f : \Sigma^k \rightarrow \Sigma^k | k \in \mathbb{N}\}$ *et* $F$ *sont polynomialement indistingables.*

[8] $A^O$ dénote l'algorithme $A$ qui accède à $O$ comme oracle.

Une propriété à la base de la construction d'un système de signature à l'abri des attaques à l'extracteur est *l'indépendance historique.*

**2.4 Définition** *Un système de signature est* indépendant de son histoire *si la signature d'un message est totalement indépendante*

1. *des signatures des messages précédents*
2. *du nombre de messages déjà signés*

**2.1 Lemme** *Tout système de signature invulnérable aux attaques adaptives à message choisi et qui est déterministe et indépendant de son histoire est invulnérable aux attaques à l'extracteur.*

**Preuve:**

Il suffit de montrer que parmi les opérations permises au faussaire-"extracteur", seules les deux premières (section 2.3.1) peuvent lui être utiles. Un tel faussaire serait donc réduit à ne pouvoir dans les faits qu'effectuer une attaque adaptive, qui, par hypothèse, ne pourra pas réussir mieux qu'avec probabilité négligeable.

Remarquons d'abord que les seules opérations permises dans une attaque à extracteur qui ne le sont pas dans une attaque adaptive sont les suivantes:

- après avoir demandé la séquence de messages $m_1, m_2, \ldots, m_k$, le faussaire peut remettre le signataire dans sa configuration précédant sa signature de $m_i$ (c'est-à-dire, suivant la séquence de signatures $m_1, m_2, \ldots, m_{i-1}$ pour $1 \leq i \leq k$; si le ruban aléatoire n'a pas été changé, ceci aura eu aussi pour conséquence de forcer le signataire à ré-utiliser la même séquence aléatoire;
- changer le contenu du ruban aléatoire du signataire avant de demander une signature.

Mais comme le système est à la fois indépendant de son histoire et déterministe, ni l'une ni l'autre de ces opérations ne peut donner au faussaire-"extracteur" un avantage sur le faussaire qui monte une attaque adaptive.

□

Il est clair que le système de Bellare-Micali tel que décrit à la section 2.2.2 n'est ni indépendant de son histoire, ni déterministe. On présente maintenant comment modifier ce système pour qu'il ait ces deux propriétés[9].

**2.3 Théorème** *Il existe au moins un système de signature qui résiste aux attaques à l'extracteur.*

**Preuve:**

Pour rendre le système de Bellare-Micali indépendant de son histoire, on applique une idée due à Goldreich et Levin [G86, GMR88]. On procède en fait en deux étapes: on rend d'abord le système indépendant des signatures précédentes, puis on le rend indépendant du *nombre* de messages ayant déjà été signés dans le passé.

Pour rendre la $i$ ième signature indépendante des $i-1$ signatures qui l'ont précédé, il faut, plutôt que de choisir et signer une nouvelle fonction $f$ après chaque signature, rendre le choix de ces fonctions *déterministe*, sans pour autant compromettre la sécurité du système.

On procède en construisant un arbre binaire, dont chacun des noeuds représente une nouvelle fonction, semblable à celle qui permettait de signer un bit de message et une nouvelle fonction dans le système original. On procède toutefois ici de façon

[9] Les techniques utilisées ici peuvent également être adaptées à tous les autres systèmes présentés à la section 2.2.2.

un peu différente: plutôt que de signer un bit de message et une fonction à chaque noeud, on valide un bit de message et *deux* fonctions. Ces deux fonctions, à leur tour, sont les fils du noeud courant dans l'arbre.

Dans ce nouveau contexte, la signature du $i^{\text{ième}}$ message consiste à donner les signatures de toutes le fonctions sur le chemin parcouru entre la racine et le noeud $i$ (les noeuds étant énumérés par niveaux). Si le choix des fonctions le long de ce chemin est déterministe, alors la signature d'un message ne dépend d'aucun message qui l'ait précédé. On montre maintenant comment rendre cet arbre déterministe.

Soit $F$ une famille de fonctions polynomialement aléatoire et $k$ le paramètre de sécurité du protocole. Soit $p$ un polynôme en $k$ qui représente le temps d'exécution du générateur $G$ sur entrée $1^k$. Le signataire choisit $\rho_1$ aléatoirement dans $F_{p(k)}$ et met l'indice correspondant à cette fonction dans sa clef secrète. La fonction qui apparaît sur le noeud $i$ sera choisie en exécutant $G(1^k)$, mais avec $\rho_1(i)$ sur le ruban aléatoire. Or, comme $\rho_1$ provient d'une famille de fonctions polynomialement aléatoire, le système de signature résultant sera aussi sécuritaire que le système original, car sinon, l'existence d'un faussaire impliquerait qu'on puisse distinguer $\rho_1$ d'une véritable fonction aléatoire, ce qui contredirait le choix de $F$.

Dans le système résultant, la signature du $i^{\text{ième}}$ message ne dépend donc plus des $i-1$ signatures précédentes, mais seulement de $i$. Pour éviter ceci, il suffit de choisir $i$ comme une chaîne aléatoire de $|I|$ bits (où $I$ est le nombre maximal de signatures produits par le système $= 2^k$). De plus, comme on veut un système déterministe, il suffit pour le signataire de choisir une seconde fonction $\rho_2$ choisie aléatoirement dans $F_k$, et de choisir $i$ comme étant $\rho_2(m)$, où $m$ est le message à signer. Comme ci-haut, le signataire garde l'indice de $\rho_2$ dans sa clef secrète.

□

### 2.3.3 Conséquences de la nouvelle attaque

En résumé, on a présenté dans cette section une nouvelle sorte d'attaque qui n'avait jusqu'à maintenant jamais été considérée. En plus d'avoir démontré que les systèmes de signature proposés dans la littérature ne résistent pas à ce type d'attaque, on a exhibé une variante d'un de ces systèmes qui est invulnérable face à ce type d'attaque.

Une conclusion qui s'impose suite à ces résultat est qu'il ne faut jamais supposer que les ressources d'un éventuel adversaire sont limitées à celles qu'on a prévues. Il faut se méfier d'expressions comme "le type d'attaque le plus fort possible" et toujours être à l'affût de faiblesses dans les modèles employés.

De plus, il semble que ce type d'attaque pourrait être appliqué à d'autres types de protocoles cryptographiques. Il serait intéressant de voir quels dommages, s'il en est, cette attaque peut causer dans d'autres contextes. On entreprend dans le chapitre qui suit l'examen de cette question dans le contexte d'un de ces types de protocoles: les preuves de connaissance.

# Chapitre 3

# Preuves de connaissance

Les preuves de connaissance sont une variante des preuves interactives où le but du prouveur n'est plus seulement de démontrer qu'une phrase est dans un langage donné, mais de convaincre le vérificateur qu'il *connaît* en fait un argument court et convaincant de ce fait. La nuance, en apparence subtile, soulève en fait plusieurs questions d'importance, sur lesquelles on tentera de jeter un peu de lumière dans ce chapitre.

L'objet original de la recherche menant à ce mémoire était de trouver une preuve de connaissance pour laquelle il est possible de démontrer que le prouveur bénéficie du même type de sécurité que le signataire dans un système de signature résistant aux attaques à l'extracteur. Malheureusement, ce problème s'est avéré bien plus ardu que les laissaient croire les apparences, et demeure la principale question ouverte de ce mémoire.

## 3.1 But

Les preuves de connaissance est un concept dû à Feige, Fiat et Shamir [FFS88]. La différence fondamentale entre les preuves interactives telles que décrites à la section 1.2 et les preuves de connaissance est que le vérificateur ne sera pas contenté d'être convaincu qu'une phrase est dans un langage: il veut retirer du protocole la conviction que le prouveur possède une preuve concise (dans le sens de la définition de la classe $\mathcal{NP}$) de l'appartenance de cette phrase au langage. Cette différence dans le niveau de connaissance peut être très importante: par exemple, c'est une chose que d'être convaincu qu'un nombre est composé, mais c'en est tout une autre que de connaître ses facteurs! En effet, une preuve interactive qu'un nombre est composé ne serait pas utile puisqu'il existe un algorithme polynomial qui permet au vérificateur de se convaincre de ce fait par lui-même [R80, SS77]. Par contre, il est intéressant, dans certains protocoles cryptographiques, que le prouveur montre qu'il connaît en fait la factorisation complète d'un nombre spécifique.

Ce type preuve peut être utilisé dans le contexte de plusieurs protocoles cryptographiques. Par exemple, il se peut qu'un intervenant veuille être assuré que l'information qu'il s'apprête à donner à son interlocuteur *ne lui apprendra rien*. Pour être convaincu de ce fait, il voudra que cet interlocuteur lui fasse la preuve qu'il connaît déja quelqu'information pertinente, et ce par le biais d'une preuve de connaissance.

**3.1 Définition** *Un* prédicat polynomial $\Pi(I, S)$ *est un prédicat pour lequel il existe*

1. *un polynôme* $p$ *tel que* $\Pi(I, S) \Longrightarrow (|S| \leq p(|I|)) \vee (|I| \leq p(|S|))$
2. *une machine de Turing polynomiale permettant de décider de la valeur de* $\Pi(I, S)$ *quels que soient* $I$ *et* $S$.

Plus précisément, donc, une preuve de connaissance pour un prédicat polynomial $\Pi$ est un protocole dans lequel le prouveur tente de convaincre le vérificateur qu'il "connaît" une chaîne $S$ telle que $\Pi(I, S)$ est vrai pour un $I$ donné. En particulier, le prédicat $\Pi$ pour un langage $L$ peut représenter le fait que $S$ est une preuve concise (dans le sens de la définition de la classe $\mathcal{NP}$) du fait que $I$ est dans $L$.

Cette définition restreint donc l'ensemble des langages à posséder un preuve de connaissance à être un sous-ensemble de $\mathcal{NP}$. Ceci est à contraster avec le fait que l'ensemble des langages possédant une preuve interactive (à la [GMR89]) est en fait la classe $\mathcal{P}$SPACE [S90].

Il est important de noter, en premier lieu, que ce concept se réduit aux preuves interactives si on considère un prouveur à puissance de calcul illimitée. En effet, dans ce contexte, la seule existence de $S$ satisfaisant $\Pi(I, S)$ implique qu'un prouveur à puissance de calcul illimitée "connaît" ce $S$, puisqu'une fouille exhaustive lui permettrait de le trouver. Ce modèle ne met donc en jeu que des prouveurs limités à un temps de calcul polynomial.

## 3.2 Que veut dire "connaître"?

La principale question qui se pose avant de pouvoir formaliser le concept de preuve de connaissance est: "qu'entend-on par *connaître*?". En fait, la question est loin d'être simple, surtout lorsqu'on fait face à des intervenants que l'on peut soupçonner d'être malhonnêtes.

Une définition possible de "connaître" $S$ tel que $\Pi(I, S)$ pourrait être d'exiger que $S$ soit inscrit en clair sur le ruban de calcul privé de la machine de Turing. Il est vrai que toute machine de Turing ayant ce secret en clair sur son ruban

"connaît" ce secret, mais l'inverse est-il vrai? Par exemple, peut-on dire qu'une machine ayant un encodage différent de cette information ne la "connaît" pas? Que dire également des machines qui possèdent une information équivalente dans le sens où elles pourraient reconstituer ce secret en temps polynomial? Ou encore celle qui l'a "à même son code", sans qu'il n'apparaisse sur son ruban privé?

Par ailleurs, on ne peut pas simplement dire qu'une machine $M$ "connaît" $S$ s'il existe une autre machine capable de produire $S$ lorsqu'elle a accès à $M$ (par exemple, avec $M$ comme boîte noire), puisqu'une telle machine existe toujours: celle qui, sur n'importe quelle entrée, produit $S$ en sortie!

La notion de connaissance due à [FFS88] part toutefois de cette idée, mais exige que la machine ayant le contrôle soit uniforme: une machine qui prétend connaître un $S$ tel que $\Pi(I, S)$ doit être dans une classe d'algorithmes qui ensemble, connaissent des $S$ satisfaisants pour tous les exemplaires $I$ satisfaisables. On définit la connaissance en fonction d'une classe d'algorithmes en exigeant l'existence d'un extracteur qui puisse extraire chacun des secrets $S$.

La définition qui suit formalise ce concept (introduit dans [FFS88]). Comme dans la section 2.3.1, les opérations permises à l'extracteur $E$ sur la machine $M$ qu'il contrôle sont de fournir les entrées et recevoir les sorties de $M$, ainsi que de prendre des instantanés de la configuration de $M$ entre les étapes du protocole et de remettre $M$ dans une des configurations sauvegardées. Pour plus de généralité, on pourrait aussi laisser à l'extracteur la possibilité de contrôler le ruban aléatoire du prouveur.

**3.2 Définition** *Soit $\mathcal{C}$ une famille d'algorithmes et $\varphi : \mathcal{C} \rightarrow 2^{\Sigma^*}$ avec*

$$\bigcup_{M \in \mathcal{C}} \varphi(M) = \{I | \exists S \ \Pi(I, S)\}.$$

*$\mathcal{C}$ connaît la relation $\Pi$ aux points définis par $\varphi$ si*

($\exists E$ *une machine de Turing polynomiale*) ($\forall\ M\ \in \mathcal{C}\ $) ($\forall$ *polynôme* $p$)
($\forall\ x\ \in \varphi(M)$) *si* $|x|$ *est suffisamment grand alors*

$$Prob[E(M,x) = y \text{ tel que } \Pi(x,y)] >\ 1 - 1/p(|x|).$$

*La probabilité est prise sur les choix aléatoires de l'extracteur (qui a le contrôle des choix aléatoires de* $M$*).*

On appelle "extracteur" la machine $E$, puisque son rôle est d'extraire de $P$ le secret qu'il prétend connaître.

**3.3 Définition** *Une machine de Turing interactive* $M_0$ connaît $S$ tel que $\Pi(I,S)$ *s'il existe une classe d'algorithmes* $\mathcal{C}$ *avec* $M_0 \in \mathcal{C}$ *et* $\varphi$ *avec* $I\ \in \varphi(M_0)$*, tels que* $\mathcal{C}$ *connaît* $\Pi$ *aux point définis par* $\varphi$ *et en particulier*

$$Prob[E(M_0, I) = S] > 1 - 1/p(|x|)$$

Il est clair que tout algorithme satisfaisant à cette définition "connaît" le secret qu'il prétend avoir comme on l'entend intuitivement, puisqu'il lui suffirait d'exécuter le programme de l'extracteur (en coopérant avec celui-ci quand il s'agit d'enregistrer sa configuration actuelle ou de se remettre dans une ancienne configuration) pour obtenir l'information qu'il prétend connaître en clair. Toutefois, il est moins évident que tous les algorithmes qui intuitivement "connaissent" un secret tomberont sous le chef de cette définition. En particulier, cette définition exclut toute possibilité d'un algorithme qui "connaîtrait" un secret mais qui saurait le protéger contre un extracteur. Cette question sera abordée à la section 3.4.

## 3.3 Les modèles formels

Les principaux modèles formels de preuves de connaissance sont le modèle original de [FFS88], ainsi que celui de [BCLL91]. Le dernier est en fait une généralisation de [FFS88], et diffère sur le plan des hypothèses cryptographiques requises pour la sécurité du vérificateur. Ils sont tous deux basés sur le l'existence d'extracteurs pour assurer la connaissance du prouveur.

### 3.3.1 Le modèle de FFS

Dans une preuve de connaissance tel que défini par [FFS88], les deux intervenants sont des machines de Turing interactives, mais en plus des rubans aléatoire et secret, ils disposent d'un ruban "de connaissance". Dans le cas du prouveur honnête, ce ruban contient, au début du protocole, le secret $S$ qui fait l'objet du protocole.

Un protocole de preuve de connaissance spécifie donc le comportement du prouveur et du vérificateur honnêtes. Il faut noter, cependant, qu'un prouveur dont l'information qui fait l'objet du protocole n'est pas sur le ruban de connaissance est considéré comme *malhonnête*, malgré qu'on doive accepter ses preuves s'il connaît, dans le sens de la définition ci-haut, le secret pour l'exemplaire donné.

**Notation:**

1. On dénotera par $\overline{X}$ le joueur $X$ honnête ("droit"), et $\widetilde{X}$ le joueur $X$ malhonnête ("croche"). $X$ représentera n'importe quel joueur $X$, honnête ou non.

2. Soit X une machine de Turing interactive ayant pour ruban aléatoire $R_X$ et ruban de connaissance $C_X$; on dénote par $E(X, R_X, C_X)$ le résultat de l'exécution de $E$ qui contrôle $X$ comme une boîte noire. Les opérations sur $X$ permises à $E$ sont:

- fournir les données d'entrée à la boîte noire
- obtenir les sorties correspondantes
- prendre un instantané de la configuration interne de la boîte noire (sans pouvoir en analyser le contenu). Ceci inclut les rubans de calcul et le ruban aléatoire.
- remettre la boîte noire dans un des états sauvegardés au préalable.

**3.4 Définition** *Une paire de machines interactives (P,V) est une* preuve de connaissance *si elle satisfait aux contraintes suivantes: (N.B. toutes les M.T. considérées sont polynomiales)*

**Complétude** $(\forall a)\,(\overset{\infty}{\forall}\, I)$

*si* $\overline{P}$ *a* $S$ *sur son ruban de connaissances alors*

$$Prob[(\overline{P},\overline{V}) \text{ accepte } I] > 1 - 1/|I|^a.$$

*La probabilité est prise sur les choix aléatoires de* $\overline{P}$ *et* $\overline{V}$.

**Consistance** $\exists \mathcal{C}$ , $\varphi$ *tels que* $\mathcal{C}$ *connaît* $\Pi$ *aux points définis par* $\varphi$ *et tels que*

$$(\forall a)\,(\forall P)$$
$$Prob[(P,\overline{V}) \text{ accepte } I] > 1/|I|^a \Longrightarrow P \in \mathcal{C} \quad et\ I \in \varphi(P).$$

*La probabilité est prise sur les choix de* $\overline{V}$.

### 3.3.2 Les autres modèles

Un autre modèle formel de preuves de connaissance a été proposés pour admettre les protocoles basés sur certaines hypothèses cryptographiques. Dans le modèle de [FFS88], une preuve reposant sur une hypothèse cryptographique n'est admise

*que si l'hypothèse est vraie*, alors que dans le modèle de [BCLL91], elle demeure une preuve de connaissance même si l'hypothèse s'avère fausse. Nous décrivons brièvement ce modèle ici, en plus de proposer une nouvelle variante du modèle de [FFS88].

### Le modèle de BCLL

Dans le contexte des preuves interactives du modèle de [BCC88], la confiance du vérificateur repose souvent spécifiquement sur l'impossibilité pour le prouveur de briser une hypothèse cryptographique. Le modèle de preuves de connaissance de [BCLL91] rend explicite cette hypothèse.

Supposons que l'hypothèse cryptographique appliquée au prouveur soit "aucun algorithme polynomial ne peut résoudre $\mathcal{T}(x)$ pour tout $x$". Dans le modèle de [BCLL91], l'extracteur doit prendre en paramètre $x$ et $I$, et trouver $S$ tel que $\Pi(I, S)$ ou résoudre $\mathcal{T}(x)$.

Cette définition est en fait équivalente à la proposition suivante: "soit que $(P, V)$ est une preuve de connaissances au sens de [FFS88], ou bien il existe un algorithme polynomial pour briser l'hypothèse cryptographique".

### Nouvelle variante du modèle de FFS

Une des faiblesses du modèle de [FFS88] est qu'il ne considère que les prouveurs limités à un temps de calcul polynomial. Par ce fait, il admet des protocoles dans lequel un prouveur ayant une puissance de calcul plus que polynomiale (sans nécessairement être illimitée) pourrait déjouer le vérificateur, sans connaître le secret qu'il prétend savoir.

Par exemple, on peut très bien imaginer l'existence d'un protocole de con-

naissance des facteurs d'un très grand nombre dans lequel il suffirait de savoir calculer des résidus quadratiques pour berner le vérificateur. En effet, il n'est pas impensable que la résiduosité quadratique soit plus facile à calculer que la factorisation, malgré que ni l'un ni l'autre ne soit calculable en temps polynomial. Un tel protocole serait acceptable dans le cadre de la définition de [FFS88], ce qui semble compromettre inutilement la sécurité du vérificateur, puisqu'on ne connaît pas *a priori* la puissance de calcul du prouveur auquel on fait face.

La variante suivante sur le modèle permettrait de contourner cette faiblesse: on exige que le protocole puisse être réalisé par un prouveur polynomial, mais le vérificateur doit être à l'abri d'un prouveur, aussi puissant soit-il, qui ne connaisse pas le secret. Bien entendu, cette condition ne s'applique qu'aux prouveurs ayant une puissance de calcul strictement inférieure à la difficulté du problème en question, sans quoi ils pourraient calculer la preuve d'eux-même, réduisant ainsi ce cas aux preuves interactives traditionnelles.

## 3.4 Critique du modèle de FFS

À la lumière des résultats obtenus au chapitre précédent, la question suivante vient mettre en doute la validité de la base même des définitions de "preuves de connaissance": *peut-on concevoir une preuve interactive qui démontre la connaissance d'une preuve d'un énoncé, mais qui soit telle que le prouveur puisse se protéger contre des attaques à l'extracteur?* La réponse à cette question demeure ouverte, mais nous examinerons dans cette sections les conséquences d'une réponse (positive ou négative), ainsi que les rapprochements à faire entre cette question et d'autres à portée semblable.

### 3.4.1 Sécurité du prouveur contre un extracteur

Dans le chapitre précédent, il a été démontré qu'il est possible de concevoir des systèmes de signature qui puissent résister à des attaques à l'extracteur. Il est donc naturel de se demander si ce type de sécurité peut être assuré à des intervenants dans des protocoles cryptographiques autres que les signatures numériques.

Toutefois, cette question pose des problèmes lorsqu'on examine le cas des preuves de connaissance telles que définies dans [FFS88, BCLL91]. En effet, l'existence d'un extracteur est à la base même de ces définitions. Il serait donc contradictoire d'imaginer un protocole pour lequel on démontre à la fois

- que c'est une preuve de connaissance dans le sens de [FFS88] (ou [BCLL91]), et
- que le prouveur est protégé contre des attaques à l'extracteur.

Ainsi, si on admet que ces définitions sont correctes à la base, on est forcé d'accepter que ce niveau de sécurité est inaccessible pour les prouveurs dans les preuves de connaissance, alors qu'on peut assurer ce type de sécurité pour les signataires dans les protocoles de signature numérique. Dans ce cas, il serait très intéressant de déterminer ce qui différencie les systèmes de signature des preuves de connaissance en termes de la sécurité qui peut être assurée au prouveur ou au signataire, selon le cas, dans le but de caractériser les protocoles qui admettent ce type de sécurité pour au moins un des joueurs.

Un certain nombre d'hypothèses pourrait être émises pour expliquer cette différence. L'interaction, présente dans les preuves interactives mais pas dans les signatures, pourrait être à l'origine de cette distinction. Si c'était le cas, que pourrait-on dire des *signatures incontestables*, ce concept récemment introduit dans [CvA89]?

Un autre point sur lequel les preuves de connaissance diffère des signatures est la "confiance" qu'on accorde au prouveur ou au signataire. En effet, alors qu'on n'accorde aucune confiance au prouveur qui tente de convaincre son vis-à-vis qu'il connaît effectivement le secret en question, on présume toujours que le signataire, s'il est légitime, a fait le choix de la clef publique de façon *honnête*, et qu'il n'essaie pas de signer des documents à partir d'une clef publique dont il ne connaîtrait pas la clef secrète correspondante. Cette différence entre les deux contextes pourrait être à l'origine du fait que l'un puisse se protéger contre un extracteur et pas l'autre.

Prenant le point de vue contraire, on peut également remettre en question la généralité des définitions basées sur l'existence d'un extracteur. Il faudrait alors exhiber un protocole qui soit une preuve convaincante (au sens intuitif) de la connaissance d'un secret, mais tel que le prouveur soit à l'abri d'un extracteur contrairement à ce qu'exigeraient les définitions proposées par [FFS88] et [BCLL91].

### 3.4.2 FFS est-il inadéquat?

#### Protocole candidat

Le protocole présenté ici est un exemple de protocole qui semble devoir être un preuve de connaissance, et pour lequel il paraît clair qu'il ne peut pas exister d'extracteur. Ceci n'a toutefois pas été démontré, et ne demeure donc qu'une conjecture. Le protocole a toutefois le bénéfice de la simplicité, et comme l'intuition qu'il offre est très forte, il permet d'étayer la thèse stipulant que la définition de [FFS88] est insuffisante.

Ce protocole démontre la connaissance d'une brèche secrète d'une permutation à brèche secrète $\varphi$, qui possède également la propriété suivante: la connaissance

de la brèche secrète est nécessaire et suffisante (au sens des réductions de Turing) à la connaissance d'un algorithme d'inversion pour $\varphi$.

Le protocole de base consiste en cinq échanges de messages: dans la première étape, le prouveur choisit les paramètres d'un système de camouflage de bits. Ensuite, le vérificateur s'engage à un entier $x$, en envoyant un camouflage pour chaque bit dans la représentation de $x$. Le prouveur choisit alors $x$ (au hasard) et l'envoie au vérificateur. Finalement, le vérificateur dévoile $x$, suite à quoi le prouveur réplique par $\varphi^{-1}(x+y)$.

Plus formellement, fixons un système de camouflage inconditionnellement camouflant [BY90] dont les paramètres sont soit publics ou choisis par le receveur[1]. Définissons aussi un opérateur binaire $\oplus$ sur $\mathrm{Dom}(\varphi)$ qui induit la permutation $\rho_a(x) \equiv a \oplus x$.

Le comportement des prouveur et vérificateur *honnêtes* est défini par le protocole suivant:

[1]Un exemple basé sur l'hypothèse du logarithme discret est décrit dans [BCC88, § 6.1.2].

1. $P$ choisit et envoie à $V$ les paramètres du système de camouflage.
2. $V$ choisit $x \in Dom(\varphi)$ au hasard, dont la représentation en binaire est $x_1, x_2, \ldots, x_k$, choisit $k$ témoins $w_1, w_2, \ldots, w_k$ et envoie $c_1, c_2, \ldots, c_k$ à $P$, où chaque $c_i$ est le camouflage de $x_i$ avec comme témoin $w_i$.
3. $P$ choisit $y \in Dom(\varphi)$ au hasard et l'envoie à $V$.
4. $V$ dévoile $c_1, c_2, \ldots, c_k$ en envoyant $w_1, w_2, \ldots, w_k$ à $P$.
5. Si les camouflages sont ouverts correctement, $P$ retourne à $V$ $\varphi^{-1}(x \oplus y)$. $V$ accepte si et seulement si $P$ a répondu correctement.

Il est clair, intuitivement, que le prouveur qui réussit à répondre correctement à tous les défis a nécessairement accès à une fonction qui calcule $\varphi^{-1}$, puisqu'il n'a aucun contrôle sur la valeur finale de $z \equiv x \oplus y$. En effet, au moment où $P$ fixe la valeur de $z$ en choisissant $x$, il n'a aucune information sur $y$, et conséquemment sur $z$, par le fait que le système de camouflage de bits est inconditionnellement camouflant. S'il n'avait pas de tel algorithme à sa disposition, la probabilité qu'il ne se fasse pas prendre serait exponentiellement petite. Mais comme $\varphi$ est tel que la connaissance d'un algorithme pour $\varphi^{-1}$ équivaut à la connaissance d'une brèche secrète pour $\varphi$, le protocole démontre bien que le prouveur connaît une telle brèche secrète.

Par ailleurs, il paraît également clair qu'il ne peut pas exister d'extracteur, puisque celui-ci, pour obtenir la brèche secrète, n'a apparemment aucune autre

ressource que d'utiliser la réduction de algorithme pour $\varphi^{-1}$ vers la connaissance de la brèche secrète. Or un extracteur, pour appliquer cette réduction, doit pouvoir obtenir les valeurs $\varphi^{-1}(z)$ pour les $z$ *de son choix*. Mais dans ce protocole, c'est la réponse $x$ du prouveur qui détermine la valeur finale de $z$.

En particulier, un prouveur (malhonnête) pourrait prendre comme $y$ le résultat d'une fonction à sens unique *sur les camouflages* envoyés par $V$. Dans ce cas, l'extracteur, jouant le rôle de $V$ ne pourra pas contrôler le $z$, et donc ne pourra pas appliquer la réduction qui lui permettrait d'obtenir la brèche secrète de $\varphi$.

Ce problème, quoiqu'offrant une intuition très forte, demeure la principale question ouverte de ce travail. La principale difficulté rencontrée a sans doute été que le problème lui-même remet en question l'application d'une des techniques de preuves les plus répandues en cryptologie: le principe des boîtes noires. De plus, les voies les plus prometteuses se sont avéré infructueuses.

Les deux volets de la preuve, c'est-à-dire montrer que le protocole est convaincant, et montrer que le protocole n'admet pas d'extracteur, semblent aussi difficiles à obtenir l'un que l'autre. D'une part, pour montrer que le protocole est convaincant, il faut d'abord exposer une définition alternative, admettant ce protocole. Pour ce faire, il faut arriver à trouver un concept complètement différent pour définir la connaissance, sans faire appel au principe des boîtes noires. Comme plusieurs chercheurs l'ont constaté [GK89, Or87], ceci paraît très difficile à imaginer.

D'autre part, la recherche d'une preuve qu'il n'existe pas d'extracteur pour un protocole donné se heurte à des difficultés techniques inattendues. On décrit dans ce qui suit certaines des techniques essayées, et les raisons pour lesquelles elles se sont soldé par des échecs. En plus de d'établir des balises pour les recherches à venir, ces observations mènent à nombre de questions ouvertes intéressantes.

### L'utilisation de sous-protocoles

En général, il semble impossible de concevoir des protocoles qui soient des contre-exemples tout en incluant des sous-protocoles déjà existants. La raison est que la grande majorité des sous-protocoles ayant des propriétés "intéressantes", le sont justement parce qu'une preuve utilisant le principe des boîtes noires a été faite pour prouver ces propriétés. Dans tous les cas essayés, cette même preuve donnait lieu de façon plus ou moins immédiate à une construction d'un extracteur! Il faudrait donc utiliser à leur place des protocoles échappant aux preuves par boîtes noires, ce qui nous ramènerait au point de départ de nos recherches.

Ces tentatives ont ainsi mis en évidence que l'utilisation du principe des boîtes noires est très répandue, et il appert que la découverte d'une nouvelle technique permettrait d'ouvrir des portes là où le principe des boîtes noires imposait des limites.

### Preuves par contradiction

L'approche sans doute la plus naturelle de montrer qu'un protocole n'admet pas d'extracteur serait de supposer qu'il en existe un et de montrer que ceci mènerait à l'existence d'un algorithme polynomial pour un problème "difficile". Dans le cas du protocole présenté plus haut, il faudrait montrer que l'existence d'un extracteur permettrait de construire un algorithme pour inverser $\varphi$.

Paradoxalement, la technique la plus courante pour faire ce type de réduction dans le contexte des protocoles est la technique des boîtes noires. Alors que les réductions Turing se font en permettant à un algorithme d'accéder à un oracle (somme toute, à une boîte noire qui calcule une fonction), l'analogue se fait dans le contexte des protocoles en permettant à un algorithme de manipuler un des

joueurs. Dans le cas de l'extracteur, le joueur "dans la boîte" est le prouveur; ici, ce sera l'extracteur lui-même qui serait mis en boîte pour arriver à une contradiction.

Toutefois il devient vite apparent que ce type de scénario ne produira pas la preuve voulue. Supposons pour l'instant qu'il existe un extracteur qui, lorsqu'il "met en boîte" le prouveur, réussit à lui faire cracher son secret. Dans ses interactions avec le prouveur, l'extracteur doit jouer le rôle du vérificateur. Notre but maintenant serait de prouver que l'existence de cet extracteur donne lieu à un algorithme pour inverser $\varphi$. Or pour espérer leurrer l'extracteur pour qu'il produise le secret du prouveur qu'il croit avoir mis en boîte, encore faut-il savoir jouer le rôle du prouveur de façon convaincante. Mais la puissance de calcul à notre disposition est justement polynomiale, alors qu'il doit être irréalisable de jouer le rôle du prouveur sans connaître le secret (sans quoi le protocole original ne saurait être convaincant). Une issue à cet apparent paradoxe semble, du moins pour l'instant, improbable.

### L'approche de Goldreich-Krawczyk

Finalement, il est très intéressant de noter l'existence de travaux indépendants indiquant que le principe des boîtes noires pourrait être insuffisant dans certains cas. En effet, Goldreich et Krawczyk montrent dans [GK89] que des preuves interactives dans le modèle de [GMR89] ne peuvent pas être *black-box zero-knowledge* [Or87] si elles se font en moins de quatre échanges de messages. Ils démontrent également que les protocoles à nombre d'échanges constant ne peuvent pas être *black-box zero-knowledge* si le vérificateur est limité à répondre uniquement par des choix aléatoires.

Il a donc déjà été démontré, dans un contexte différent, qu'il existe certains protocoles pour lesquels le principe des boîtes noires est insuffisant. Toutefois,

les techniques utilisées dans [GK89] semblent inutilisables dans le contexte des preuves de connaissance. La principale raison est que dans le cas du *black-box zero-knowledge*, le joueur "emboîté" est le vérificateur, alors que dans le cas des preuves de connaissance, le joueur "en boîte" est le prouveur. Comme il a été remarqué dans la section précédente, cette technique n'est pas applicable dans ce contexte.

# Conclusion

Le principal mérite de ce mémoire est sans doute d'avoir exposé une problématique originale et intéressante. Les questions ouvertes exposées par ce travail sont nombreuses, intrigantes, et on peut imaginer que les techniques qui devront être développées pour les résoudre auront de nombreuses applications en cryptologie.

En guise de conclusion, on en présente ici quelques-unes. Certaines ont déjà été soulevées dans le texte, mais on en propose également des nouvelles.

## Questions ouvertes

La question la plus générale qui ressort de ce travail est la suivante: *quel est l'effet sur les protocoles cryptographiques lorsqu'on cesse de considérer le principe des boîtes noires comme une technique de preuve, mais qu'à la place, on le considère comme une façon d'extraire de l'information d'un participant à un protocole?* On a donné des éléments de réponse dans le cas des signatures numériques, et on a exploré le cas des preuves de connaissances, mais qu'en est-il pour les autres types de protocoles?

Certains types de protocoles qui seraient d'intérêt comme premier pas dans cette direction sont:

- *les signatures incontestables*[CvA89]: ce type de signature comporte un volet interactif (pour les protocoles de confirmation et réfutation), ce qui les place, pour ainsi dire, à mi-chemin entre les preuves de connaissance (interactives) et les signatures numériques habituelles (non-interactives). L'étude de ces protocoles, pour trouver s'il est possible de trouver des signatures incontestables qui résisteraient à une attaque à l'extracteur, jetterait sans doute de la lumière sur le cas des preuves de connaissance.

- *les protocoles dissimulant les témoins* [FS90] (*witness hiding protocols*): la définition de ce type de protocole, comme les preuves de connaissances, est basée sur l'existence d'extracteurs. Par contre, dans ce cas, le joueur "mis en boîte" est le vérificateur, plutôt que le prouveur. Il devrait donc être plus facile, dans ce contexte, de faire une preuve par contradiction[2] pour prouver la non-existence d'extracteur pour un protocole candidat.

S'il s'avérait impossible de déjouer un extracteur dans certains de ces protocoles, il serait intéressant d'en trouver la raison. Serait-il possible de caractériser simplement les protocoles qui ne peuvent pas offrir la sécurité contre les extracteurs? Quelques hypothèses basées sur la différence entre les signatures numériques et les preuves de connaissance ont été émises à la section 3.4.1.

Si par contre on réussissait à trouver des preuves de connaissance ou des protocoles dissimulant les témoins, il faudrait trouver une nouvelle définition pour ces concepts, une qui ne serait pas basée sur le principe des boîtes noires.

[2]Voir la section 3.4.2

# Références


[BM88] Babai, L. et S. Moran, "Arthur-Merlin games: A randomized proof system, and a hierarchy of complexity classes", *J.Comput. System Sci.*, vol. 36, 1988, pp. 254–276.

[BM88] Bellare, M. et S. Micali, "How to Sign Given Any Trapdoor Function", *Proceedings of the 20th ACM Symposium on Theory of Computing*, 1988, pp. 32–42.

[BG89] Bellare, M. et S. Goldwasser, "New Paradigms for Digital Signatures and Message Authentication Based on Non-Interactive Zero Knowledge Proofs", *Advances in Cryptology — Proccedings of CRYPTO 89*, Springer–Verlag, 1989, pp. 194–211.

[B89] Brassard, G., "How to improve signature schemes", *Advances in Cryptology: Proceedings of Eurocrypt 1989*, Springer–Verlag, 1989, pp. 16–22.

[BCC88] Brassard, G., D. Chaum et C. Crépeau, "Minimum disclosure proofs of knowledge", *Journal of Computer and System Sciences*, Vol. 37, no. 2, 1988, pp. 156–189.

[BCLL91] Brassard, G., C. Crépeau, S. Laplante et C. Léger, "Computationally convincing proofs of knowledge", *Proceedings, Symposium on Theoretical Aspects of Computer Science*, Springer–Verlag, 1991, pp. 251–262.

[BY90] Brassard, G. et M. Yung, "One-way group actions", *Advances in Cryptology: CRYPTO 90 Proceedings*, Springer–Verlag, 1991, pp. 94-107.

[CvA89] Chaum, D. et H. van Antwerpen, "Undeniable signatures", *Advances in Cryptology: CRYPTO 90 Proceedings*, Springer–Verlag, 1991, pp. 212–216.

[DH76] Diffie, W. et M.E. Hellman, "New directions in cryptography", *IEEE Transactions on Information Theory*, vol. IT–22, 1976, pp. 644–654.

[FFS88] Feige, U., A. Fiat et A. Shamir, "Zero knowledge proofs of identity", *Journal of Cryptology*, Vol. 1, no. 2, 1988, pp. 77–94.

[FLS90] Feige, U., D. Lapidot et A. Shamir, "Multiple Non-Interactive Zero Knowledge Proofs Based on a Single Random String", *Proceedings of the 29th IEEE Symposium on Foundations of Computer Science*, 1990, pp. 308–317.

[FS90] Feige, U. et A. Shamir, "Witness Indistinguishable and Witness Hiding Protocols", *Proceedings of the 22nd ACM Symposium on Theory of Computing*, 1990, pp. 416–437.

[G86] Goldreich, O., "Two Remarks Concerning the Goldwasser-Micali-Rivest Signature Scheme", *Advances in Cryptology: CRYPTO '86 Proceedings*, Springer–Verlag, 1986, pp. 104–110.

[GGM84] Goldreich, O., S. Goldwasser et S. Micali, "On the cryptographic applications of random functions", *Advances in Cryptology: Proceedings of Crypto 1984*, Springer–Verlag, 1984, pp. 276–288.

[GK89] Goldreich, O. et H. Krawczyk, "On the composition of zero-knowledge proof systems", *Proceedings of the 17th ICALP*, Springer–Verlag, 1990, pp. 268–282.

[GMR89] Goldwasser, S., S. Micali et C. Rackoff, "The knowledge complexity of interactive proof systems", *SIAM Journal on Computing*, Vol. 18, no. 1, 1989, pp. 186–208.

[GMR88] Goldwasser, S., S. Micali et R. Rivest, "A Digital Signature Scheme Secure against Adaptive Chosen-Message Attacks", *SIAM Journal on Computing*, Vol. 17, No. 2, 1988, pp. 281–308.

[La79] Lamport, L., "Constructing Digital Signatures from a One-Way Function", SRI Intl. CSL98, 1979.

[NY89] Naor, M. et M. Yung, "Universal One-Way Hash Functions and their Cryptographic Applications", *Proceedings of the 21st ACM Symposium on Theory of Computing*, 1989, pp. 33–43.

[Or87] Oren, Y., "On the cunning power of cheating verifiers: Some observations about zero-knowledge proofs", *Proceedings of the 28th IEEE Symposium on Foundations of Computer Science*, 1987, pp. 462–471.

[R80] "Probabilistic algorithm for testing primality", *Journal of Number Theory*, vol. 12, 1980, pp. 128–138.

[R90] Rompel, J., "One-way functions are necessary and sufficient for secure signatures", *Proceedings of the 22nd ACM Symposium on Theory of Computing*, 1990, pp. 387–394.

[S90] Shamir, A., "IP=PSPACE", *Proceedings of the 31st IEEE Symposium on Foundations of Computer Science*, 1990, pp. 11–15.

[SS77] Solovay, R. et V. Strassen, "A fast Monte Carlo Test for Primality", *SIAM Journal on Computing*, vol. 6, 1977, pp. 84–85.

[Wi80] Williams, H.C., "A modification of the RSA public-key cryptosystem", *IEEE Transactions on Information Theory*, IT-26, 1980, pp. 726-729.

# Remerciements

Si j'ai enfin pu terminer ce mémoire, c'est grâce à l'aide (et à la patience!) de plusieurs personnes que je tiens à remercier chaleureusement. Je remercie le Conseil de Recherches en Sciences Naturelles et en Genie et le Fond FCAR, ainsi que Département d'Informatique et de Recherche Opérationnelle pour leur soutien financier.

Un chaleureux merci va aussi à Louis Salvail, qui m'a si gracieusement offert la victoire du concours du vent sur le lac Champlain; à Dominique Balency pour ses nombreux *m-hm* compréhensifs. J'ai également eu grand plaisir à travailler avec Mihir Bellare sur certaines parties de ce travail. Finalement, ma plus grande reconnaissance est dûe à Gilles Brassard, pour son aide constante, sa confiance en moi et ses encouragements répétés.