Université de Versailles Saint-Quentin-en-Yvelines
UFR de Sciences - École doctorale SoFt - Laboratoire PRiSM

Primitives et protocoles cryptographiques à sécurité prouvée

THÈSE

présentée pour l'obtention du grade de

Docteur de l'Université de Versailles Saint-Quentin-en-Yvelines
(Spécialité Informatique)

par

Yannick Seurin

Soutenue publiquement le 1er juillet 2009 devant le jury composé de

David Pointcheval (ENS Paris et CNRS)	*Rapporteur*
Serge Vaudenay (EPFL)	*Rapporteur*
Jacques Patarin (Univ. de Versailles)	*Directeur*
Mihir Bellare (Univ. of California San Diego)	*Examinateur*
Anne Canteaut (INRIA)	*Examinateur*
Pierre-Alain Fouque (ENS Paris)	*Examinateur*
Henri Gilbert (Orange Labs)	*Examinateur*

Travaux effectués au sein du groupe de Cryptographie
de France Télécom R&D, Issy-les-Moulineaux

Remerciements

Pour commencer, je tiens à exprimer ma profonde gratitude aux deux chercheurs et personnes remarquables qui ont présidé à la direction de cette thèse, Jacques Patarin et Henri Gilbert. Merci à Jacques, mon directeur officiel, pour son imagination, son enthousiasme et son audace l'incitant à se confronter à des problèmes qui feraient fuir plus d'un cryptographe chevronné. Merci à Henri, qui a guidé mon travail au quotidien, pour son expertise hors pair, sa disponibilité, son humour et tous les judicieux conseils dispensés durant ces années. Il est difficile de mesurer tout ce que cette thèse leur doit, et ils resteront tous deux pour moi des exemples à suivre dans le monde de la cryptographie.

Je veux également remercier David Pointcheval et Serge Vaudenay, qui m'ont fait le privilège d'accepter d'être les rapporteurs de cette thèse, Mihir Bellare qui m'honore de sa présence à la soutenance malgré un timing serré, ainsi que Anne Canteaut et Pierre-Alain Fouque que j'ai le plaisir de compter parmi les membres de mon jury. Merci par ailleurs à David de m'avoir permis d'organiser la soutenance à l'ENS.

J'ai eu la chance de collaborer avec de nombreux cryptographes expérimentés. Je remercie notamment Matt Robshaw, à l'origine de nombre des travaux auxquels j'ai participé, pour ses précieux conseils et son agréable caractère britannique, Olivier Billet, qui m'étonnera toujours par son originalité créative, sa curiosité sans limites et sa culture dépassant largement le cadre de la cryptographie, Jean-Sébastien Coron dont j'admire l'ingéniosité et la profondeur d'analyse et Thomas Peyrin, avec qui je suis fier d'avoir cosigné mon premier article alors que nous n'étions que deux thésards débutants, et qui est devenu bien plus qu'un simple condisciple. Merci également à tous mes autres coauteurs, Andrei Bogdanov, Lars Knudsen, Gregor Leander, Christof Paar, Alexander Poschmann, Charlotte Vikkelsoe et Yiqun Lisa Yin. Les relations avec certains sont restées épistolaires mais j'ai été très heureux de pouvoir travailler avec eux.

Je remercie chaleureusement les autres membres et ex-membres du groupe de cryptographie des Orange Labs, David Arditti, Ryad Benadjila, Côme Berbain, Jonathan Etrog, Gilles Macario-Rat et Duong-Hieu Phan pour leurs compétences de premier ordre et la formidable ambiance qu'ils ont contribué à faire régner au sein du groupe, ainsi que tous les « Caennais » que j'ai moins eu l'occasion de côtoyer, mais que j'ai toujours pris plaisir à rencontrer en diverses circonstances : Sébastien Canard, Iwen Coisel, Cécile Delerablée, Marc Girault, Émeline Hufschmitt, Amandine Jambert, Jean-François Misarsky et Jacques Traoré.

Je suis également très reconnaissant envers mes manageurs à France Télécom R&D, Sébastien Nguyen Ngoc et Thierry Baritaud, d'avoir toujours veillé à préserver un environnement de travail où les chercheurs puissent s'épanouir et voir leurs travaux appréciés et reconnus.

Je n'ai pas souvent été dans les locaux de l'université de Versailles Saint-Quentin-en-Yvelines. Je tiens cependant à remercier tous les membres de l'équipe de cryptographie que j'ai pu rencontrer en conférence ou lors de mes visites à Jacques, notamment les thésard(e)s Joana Treger, Aurélie Bauer et Vanessa Vitse, ainsi que les professeurs Louis Goubin, Antoine Joux et Michaël Quisquater pour les discussions toujours enrichissantes que j'ai eues avec eux et leur précieuse contribution à la féminisation de notre discipline.

Merci enfin à tous les cryptographes avec lesquels j'ai eu la joie de boire une bière, disputer une partie de bowling, visiter un temple Maya ou une jungle équatoriale, voire, de temps à autre, avoir une conversation sur la cryptographie.

Pour conclure, comment ne pas remercier tous ceux qui ont contribué à me transmettre ce fameux « goût pour la recherche », à commencer par les chercheurs extraordinaires que j'ai croisés sur ma route, notamment Jean Dalibard et Vincent Berger, mais surtout mes parents et ma tante Martine, et plus généralement mon entourage, qui m'ont patiemment, discrètement, mais inlassablement encouragé. Ce n'est qu'aujourd'hui, à la veille de terminer cette longue aventure, que je réalise tout ce que je leur dois.

Et je remercie Myriam de me rappeler tous les jours à l'essentiel, qui n'a rien à voir avec cette thèse...

***

Table des matières

Table des figures

Liste des tableaux

Liste des algorithmes

Introduction générale

La cryptographie a connu au cours du vingtième siècle (en particulier depuis le milieu des années 1970) une révolution paradigmatique profonde. Les cryptographes se sont en effet efforcés de lui donner tous les attributs d'une véritable science (en particulier une approche formelle et un traitement rigoureux), alors qu'elle n'avait auparavant progressé que par tâtonnement, tentatives ratées et améliorations successives.

Imaginer des systèmes résistants à des phénomènes naturels, tels des constructions antisismiques, est notoirement complexe. Dans le cas de la cryptographie, la tâche est d'autant plus difficile qu'il s'agit de concevoir des systèmes capables de résister à des tentatives conscientes et délibérées de les faire dévier de leur comportement normal. C'est probablement ce qui rend cette discipline si ardue mais aussi très excitante.

La première étape de la conception d'un système cryptographique, et sans doute la plus délicate d'entre toutes, est la définition précise et rigoureuse des notions de sécurité adéquates. La formalisation des justes notions de sécurité est une entreprise récente à l'échelle de l'histoire de la cryptographie et en perpétuelle élaboration. Ainsi, bien que la notion de chiffrement symétrique soit connue et utilisée depuis Jules César, ce n'est que depuis l'article fondateur de Shannon, *Communication Theory of Secrecy Systems*, datant de 1949, qu'elle est assise sur des bases mathématiques solides. Il ne faut cependant pas s'en étonner car la formalisation de la cryptographie utilise abondamment les outils de disciplines modernes telles que la théorie de l'information et la théorie de la complexité.

Une fois une définition de sécurité établie (aussi forte et générale que possible), la seconde étape consiste à démontrer que le cryptosystème conçu résiste aux attaques par une *preuve de sécurité*.

On peut établir la hiérarchie suivante dans les preuves de sécurité :

1. Preuves inconditionnelles : on parle de sécurité inconditionnelle lorsque la preuve de sécurité n'utilise aucune hypothèse non démontrée, et n'impose pas de limites à la capacité de calcul de l'attaquant. En général ces preuves ont pour cadre la théorie de l'information. Cela ne veut pas dire pour autant que le cryptosystème est inviolable dans la « réalité », toute preuve reposant sur un certain niveau d'abstraction : ainsi, une telle preuve ne prendra généralement pas en compte les attaques par canaux cachés.
2. Preuves dans le « modèle standard » : une preuve dans le modèle standard considère uniquement des algorithmes polynomiaux, mais s'appuie en général sur des hypothèses de la théorie de la complexité telles que la difficulté de la factorisation ou du logarithme discret. Ce type de preuve utilise une « réduction », c'est-à-dire une démonstration mathématique qu'un attaquant capable de mettre en défaut la sécurité du cryptosystème peut être transformé en un algorithme capable de résoudre le problème conjecturé difficile sous-jacent.

3. Preuves dans des « modèles idéalisés » : une preuve dans un modèle idéalisé procède en remplaçant certaines ressources utilisées par le cryptosystème par une primitive « idéale » (en général parfaitement aléatoire). Les deux modèles idéalisés les plus couramment employés sont le modèle de l'oracle aléatoire et celui du chiffrement par blocs idéal.
4. Enfin, on peut regrouper dans une dernière catégorie les arguments heuristiques et autres preuves de sécurité contre des types d'attaques particuliers. Ainsi, les preuves de résistance à la cryptanalyse linéaire et différentielle de l'algorithme de chiffrement par blocs AES entrent dans ce groupe.

Nos travaux de thèse nous ont donné l'occasion d'aborder les trois dernières catégories de preuves.

Ainsi, nous avons collaboré à la conception d'un algorithme de chiffrement par blocs implantable très efficacement en hardware, PRESENT, et montré qu'il était résistant à la cryptanalyse linéaire et différentielle (référence [2] de la bibliographie personnelle). Nous avons également participé au design d'une fonction de hachage nommée DASH pour laquelle on peut donner des arguments heuristiques de résistance aux collisions [7], ainsi qu'à celui de fonctions de hachage fondées sur PRESENT [8]. Tous ces travaux s'inscrivent dans la quatrième catégorie.

Dans le domaine du modèle standard, nos contributions sont de deux types. Nous avons proposé avec Jacques Patarin une méthode de conception d'algorithmes de chiffrement par blocs dont la sécurité peut être évaluée au regard des meilleures attaques génériques sur les schémas de Feistel [9]. La complexité des attaques génériques peut être vue comme une conjecture du même type que la difficulté supposée de la factorisation ou du logarithme discret (il s'agit néanmoins d'une hypothèse historiquement moins étudiée). Nous avons également considéré, avec Henri Gilbert et Matt Robshaw, les schémas cryptographiques pour l'authentification et le chiffrement symétrique fondés sur la difficulté conjecturée du problème LPN [3, 4, 5]. Ceci fait l'objet de la seconde partie de ce mémoire.

Enfin, dans le domaine des modèles idéalisés, nous avons étudié avec Thomas Peyrin la résistance aux collisions et à la pré-image de fonctions de compression construites à partir d'oracles aléatoires de taille d'entrée finie [1]. Nous avons également montré avec Jean-Sébastien Coron et Jacques Patarin que deux modèles idéalisés, celui de l'oracle aléatoire et celui du chiffrement par blocs idéal, sont équivalents [10]. Ce dernier résultat fait l'objet de la première partie de ce mémoire.

Bibliographie personnelle

Les articles publiés au cours de la présente thèse sont listés ci-dessous dans l'ordre chronologique inverse de publication. Les articles indiqués en gras sont ceux faisant l'objet du présent mémoire.

[11] Ryad Benadjila, Olivier Billet, Henri Gilbert, Gilles Macario-Rat, Thomas Peyrin, Matt Robshaw, and Yannick Seurin. *SHA-3 Proposal : ECHO*. Supporting documentation for the candidate algorithm ECHO for the NIST SHA-3 competition. Available at `http://crypto.rd.francetelecom.com/echo/`.

[10] Jean-Sébastien Coron, Jacques Patarin, and Yannick Seurin. *The Random Oracle Model and the Ideal Cipher Model Are Equivalent*. In David Wagner, editor, *Advances in Cryptology - CRYPTO 2008*, volume 5157 of *Lecture Notes in Computer Science*, pages 1–20. Springer, 2008.

[9] Jacques Patarin and Yannick Seurin. *Building Secure Block Ciphers on Generic Attacks Assumptions*. In *Selected Areas in Cryptography - SAC 2008*. To appear.

[8] Andrey Bogdanov, Gregor Leander, Christof Paar, Axel Poschmann, Matthew J. B. Robshaw, and Yannick Seurin. *Hash Functions and RFID Tags : Mind the Gap*. In Elisabeth Oswald and Pankaj Rohatgi, editors, *Cryptographic Hardware and Embedded Systems - CHES 2008*, volume 5154 of *Lecture Notes in Computer Science*, pages 283–299. Springer, 2008.

[7] Olivier Billet, Matthew J. B. Robshaw, Yannick Seurin, and Yiqun Lisa Yin. *Looking Back at a New Hash Function*. In Yi Mu, Willy Susilo, and Jennifer Seberry, editors, *Information Security and Privacy - ACISP 2008*, volume 5107 of *Lecture Notes in Computer Science*, pages 239–253. Springer, 2008.

[6] Olivier Billet, Jacques Patarin, and Yannick Seurin. *Analysis of Intermediate Field Systems*. In *Symbolic Computation and Cryptography - SCC 2008*, 2008.

[5] Henri Gilbert, Matthew J. B. Robshaw, and Yannick Seurin. *How to Encrypt with the LPN Problem*. In Ivan Damgård, editor, *International Colloquium on Automata, Languages and Programming - ICALP 2008, Track C : Security and Cryptography Foundations*, volume 5126 of *Lecture Notes in Computer Science*, pages 679–690. Springer, 2008.

[4] Henri Gilbert, Matthew J. B. Robshaw, and Yannick Seurin. *$HB^{\#}$: Increasing the Security and Efficiency of HB^{+}*. In Nigel P. Smart, editor, *Advances in Cryptology - EUROCRYPT 2008*, volume 4965 of *Lecture Notes in Computer Science*, pages 361–378. Springer, 2008.

[3] Henri Gilbert, Matthew J. B. Robshaw, and Yannick Seurin. *Good Variants of HB$^+$ Are Hard to Find.* In Gene Tsudik, editor, *Financial Cryptography and Data Security - FC 2008*, volume 5143 of *Lecture Notes in Computer Science*, pages 156–170. Springer, 2008.

[2] Andrey Bogdanov, Lars R. Knudsen, Gregor Leander, Christof Paar, Axel Poschmann, Matthew J. B. Robshaw, Yannick Seurin, and Charlotte Vikkelsoe. *PRESENT : An Ultra-Lightweight Block Cipher*. In Pascal Paillier and Ingrid Verbauwhede, editors, *Cryptographic Hardware and Embedded Systems - CHES 2007*, volume 4727 of *Lecture Notes in Computer Science*, pages 450–466. Springer, 2007.

[1] Yannick Seurin and Thomas Peyrin. *Security Analysis of Constructions Combining FIL Random Oracles*. In Alex Biryukov, editor, *Fast Software Encryption - FSE 2007*, volume 4593 of *Lecture Notes in Computer Science*, pages 119–136. Springer, 2007.

Notations

Notations générales

$\{0,1\}^n$	ensemble des chaines de bits de longueur n
$\{0,1\}^*$	ensemble des chaines de bits de longueur finie quelconque
$\{0,1\}^\infty$	ensemble des chaines de bits de longueur infinie dénombrable
$\boldsymbol{x}\|\boldsymbol{y}$	concaténation des vecteurs $\boldsymbol{x}$ et $\boldsymbol{y}$
$\boldsymbol{x}[i]$	i-ième composante du vecteur $\boldsymbol{x}$
$\boldsymbol{e}_i$	vecteur dont tous les bits valent 0 sauf le i-ième qui vaut 1
$\mathtt{Hwt}(\boldsymbol{x})$	poids de Hamming du vecteur $\boldsymbol{x}$
$[\![m,n]\!]$	ensemble des entiers compris entre m et n
ln	logarithme népérien
log	logarithme en base 2
$\mathrm{GF}(p^r)$	corps fini à p^r éléments, pour p premier et $r \geq 1$
$\mathcal{M}^G$	machine de Turing $\mathcal{M}$ accédant à un oracle G
Ψ_r^F	construction de Luby-Rackoff à r tours
$\Pi_{\boldsymbol{x},\eta}$	oracle LPN associé au vecteur $\boldsymbol{x}$ et de paramètre de bruit η

Probabilités

$\omega \leftarrow \Omega$	tirage d'un élément ω selon la loi de probabilité Ω
$x \xleftarrow{\$} X$	tirage d'un élément x selon la loi uniforme sur X
U_n	distribution uniforme sur $\{0,1\}^n$
$\mathtt{Ber}_\eta$	loi de Bernoulli de paramètre η
$\mathtt{Ber}_{n,\eta}$	loi sur $\{0,1\}^n$ telle que chaque bit suit indépendamment $\mathtt{Ber}_\eta$
$\begin{cases} \Pr_{\omega\leftarrow\Omega}[E] \\ \Pr[\omega \leftarrow \Omega \,:\, E] \end{cases}$	probabilité de l'événement E lorsque $\omega \leftarrow \Omega$
$\mathbb{E}[X]$	espérance de la variable aléatoire X

Première partie

Sécurité prouvée dans des modèles idéalisés

Introduction

Toute cette première partie se concentre sur un unique résultat, publié avec Jean-Sébastien Coron et Jacques Patarin [CPS08], sur l'équivalence entre le modèle de l'oracle aléatoire et celui du chiffrement par blocs idéal.

L'objet du premier chapitre sera d'introduire le formalisme nécessaire à la démonstration de ce résultat en définissant rigoureusement ce que l'on entend par « modèle » (nous emploierons plutôt le vocable de « primitive idéale »), et ce que signifie l'« équivalence » de deux modèles. Un concept central dans toute cette partie sera celui d'*indifférentiabilité*, qui généralise celui d'*indistinguabilité* (primordial en cryptographie à sécurité prouvée).

Le second chapitre est en grande partie constitué de la démonstration que la construction de Luby-Rackoff à 10 tours est indifférentiable d'une permutation aléatoire inversible. Sans formaliser davantage pour l'instant, ceci signifie que si un cryptosystème utilisant une permutation est prouvé sûr lorsque cette permutation est aléatoire, alors le cryptosystème reste sûr lorsque la permutation est remplacée par une construction de Luby-Rackoff à 10 tours, *même lorsque l'adversaire a accès aux fonctions internes de la construction.* L'immense majorité des précédents résultats sur la construction de Luby-Rackoff concernaient le cas où ces fonctions étaient tenues secrètes.

Le même résultat est en fait vrai dès 6 tours : c'est ce que nous avons démontré dans l'article [CPS08]. Cependant, dans un souci didactique et pour éviter de nombreux problèmes purement techniques, nous avons préféré nous concentrer dans ce mémoire sur la démonstration pour 10 tours. L'esprit de la preuve et la portée du résultat restent les mêmes et nous espérons que la lecture de la démonstration pour 10 tours constituera une première étape profitable avant celle de la preuve pour 6 tours.

Chapitre 1

Modèles idéalisés et indifférentiabilité

Dans ce chapitre, nous introduisons les principaux modèles idéalisés utilisés en cryptographie, ainsi que la notion d'indifférentiabilité qui est l'outil qui nous permettra de comparer les modèles entre eux.

1.1 Utilité des modèles idéalisés en cryptographie

Le but de la cryptographie moderne est de concevoir des cryptosystèmes à la fois efficaces et garantissant des propriétés de sécurité aussi fortes que possible. L'assurance que l'on peut avoir dans la sécurité d'un schéma cryptographique provient en général d'une preuve de sécurité, c'est-à-dire d'une démonstration mathématique qu'une certaine classe d'attaques ne peut accomplir certaines actions qui mettraient en défaut la définition de la sécurité du schéma. La portée d'une preuve de sécurité est directement reliée à la généralité des attaques considérées. C'est pourquoi on essaie toujours de maximiser le pouvoir des attaquants et de minimiser les ressources utilisées par le cryptosystème. On tente en général de conduire les preuves dans le « modèle standard », où seuls des algorithmes efficaces et des ressources raisonnables sont disponibles (cela se traduit mathématiquement par une dépendance polynomiale de certaines grandeurs comme le temps de calcul, la mémoire ou l'aléa en k, le paramètre de sécurité).

Il est notoirement difficile, voire théoriquement impossible, de concevoir des cryptosystèmes à sécurité prouvée efficaces n'utilisant aucune hypothèse non démontrée, et donc sûrs du point de vue de la théorie de l'information [Mau99], à moins de réduire les capacités de l'attaquant, en supposant par exemple sa mémoire limitée [CM97]. On est en général amené à réduire la sécurité d'un schéma à un problème calculatoire conjecturé difficile, comme le problème de la factorisation ou celui du logarithme discret.

Mais ce n'est parfois pas suffisant, et il peut être nécessaire, pour obtenir des schémas plus efficaces ou des réductions plus fines, d'introduire un modèle « idéalisé » pour certains des composants utilisés par le cryptosystème tels qu'une fonction de hachage ou un chiffrement par blocs. On parle également de méthodologie, paradigme, ou encore heuristique : tous ces termes soulignent que ces preuves de sécurité ont une portée limitée par l'idéalisation de certaines ressources utilisées par le cryptosystème.

Nous allons successivement détailler les trois modèles les plus utilisés en cryptographie : le modèle de l'oracle aléatoire, celui du chiffrement par blocs idéal, et celui de la permutation aléatoire inversible.

1.1.1 Le modèle de l'oracle aléatoire

Le modèle de l'oracle aléatoire est le plus utilisé des modèles idéalisés. Formalisé pour la première fois par Bellare et Rogaway [BR93], on trouve en fait sa trace dans des travaux antérieurs, notamment dans l'heuristique de Fiat-Shamir [FS86] permettant de transformer un schéma d'authentification en schéma de signature. Le modèle de l'oracle aléatoire consiste à supposer que tous les participants à un protocole cryptographique, y compris les attaquants potentiels, ont accès à un oracle $\boldsymbol{H} : \{0,1\}^* \to \{0,1\}^n$ (on considère parfois un oracle $\boldsymbol{H} : \{0,1\}^* \to \{0,1\}^\infty$) qui renvoie des réponses indépendantes et uniformément aléatoires dans $\{0,1\}^n$ à chaque nouvelle requête (lorsqu'une même requête est effectuée à plusieurs reprises, la même réponse est renvoyée à chaque fois). La méthodologie de l'oracle aléatoire consiste alors à concevoir un cryptosystème utilisant l'oracle $\boldsymbol{H}$, à prouver sa sécurité, puis à remplacer l'oracle par une fonction de hachage réelle H telle que SHA-1 [Nat95], étape appelée « instanciation ».

Cette méthodologie a été couramment utilisée dans les preuves de sécurité, notamment pour les schémas de chiffrement à clé publique tels qu'OAEP [BR94], les schémas de signature tels que PSS [BR96], le schéma de signature de Schnorr [Sch91], ou encore certaines variantes du schéma d'ElGamal [PS00], ainsi que pour les protocoles d'authentification [GQ88, Oka92]. Il est indéniable que la méthodologie de l'oracle aléatoire a permis, au moins à ses débuts, d'obtenir des schémas bien plus efficaces (et possédant des preuves de sécurité avec une réduction plus fine) que ceux prouvés sûrs dans le modèle standard. Cependant, les cryptographes ont été conscients dès l'introduction du modèle [BR93] que l'instanciation d'un oracle aléatoire était problématique. En effet, une fonction de hachage H est nécessairement décrite par une chaine de bits finie. Il s'agit donc d'un objet contenant incommensurablement moins d'entropie qu'un oracle aléatoire $\boldsymbol{H}$. Il a donc été communément admis qu'une preuve de sécurité dans le modèle de l'oracle aléatoire ne devait être prise que comme une indication que le protocole considéré ne possédait pas de défauts de « structure globale », et non comme une preuve rigoureuse de la sécurité du cryptosystème une fois l'oracle aléatoire instancié. En un sens, la méthodologie de l'oracle aléatoire revient à considérer que, le protocole cryptographique et la fonction de hachage considérés ayant été conçus « indépendamment », il y a une probabilité négligeable que ces deux objets « interagissent » mal. Remarquons que l'expérience aléatoire considérée (l'instanciation de l'oracle) n'étant en général pas répétée mais unique, le concept de probabilité doit être pris dans son acception plus large de confiance accordée par l'utilisateur dans le fait que le cryptosystème va bien fonctionner comme il l'espère... Par ailleurs, certains auteurs arguent qu'une preuve dans le modèle de l'oracle aléatoire prouve au moins que le schéma est sûr contre toute attaque « générique », *i.e.* faisant abstraction de la structure interne de la fonction de hachage utilisée [BR93, Sti01].

Les doutes quant au bien-fondé de la méthodologie de l'oracle aléatoire ont été confirmés de façon théorique par des résultats dits « d'ininstanciabilité », consistant en la construction de schémas cryptographiques prouvés sûrs dans le modèle de l'oracle aléatoire, mais devenant vulnérables dès que l'oracle est instancié par n'importe quelle famille de fonctions de hachage. Le premier de ces résultats est dû à Canetti, Goldreich et Halevi [CGH98]. Parmi d'autres résultats du même type on peut citer [GK03, BBP04, CGH04]. Nielsen [Nie02] a même montré qu'il existe des tâches cryptographiques (le chiffrement sans engagement) réalisables de façon sûre dans le modèle de l'oracle aléatoire, mais pas dans le modèle standard. Cependant, tous les schémas faisant l'objet de résultats d'ininstanciabilité sont « artificiels » et très éloignés d'un schéma réel (même si expliciter ce qui les rend artificiels reste à notre connaissance un problème largement ouvert [KM07]).

Pour toutes ces raisons, les cryptographes se sont efforcés de concevoir des protocoles prouvés sûrs sans avoir besoin de recourir à l'oracle aléatoire. Le plus connu de ces protocoles est le schéma de chiffrement asymétrique de Cramer-Shoup [CS98b], qui utilise une fonction de hachage H pour laquelle seule l'hypothèse de résistance à la seconde pré-image est requise (plus précisément, le schéma utilise une famille de fonctions de hachage universelle à sens unique, *UOWHF* en anglais [NY89]). Moyennant cette hypothèse « standard » sur la fonction de hachage, la sécurité du cryptosystème peut être réduite au problème Diffie-Hellman Décisionnel [Bon98]. Le développement de la cryptographie fondée sur les couplages sur une courbe elliptique [Jou00, BF01, Jou02] a largement contribué à l'obtention de schémas à la fois efficaces et prouvés sûrs dans le modèle standard [BB04b, BB04a, Wat05]. Le prix à payer est que la sécurité est réduite à des hypothèses moins bien étudiées que les problèmes classiques tels que la factorisation ou le logarithme discret.

1.1.2 Le modèle du chiffrement par blocs idéal

Tout comme les fonctions de hachage, les algorithmes de chiffrement par blocs sont un autre ingrédient essentiel de nombreux cryptosystèmes. Dans le modèle du chiffrement par blocs idéal, également dénommé « modèle de Shannon » qui fut le premier à le formaliser dans son article fondateur dès 1949 [Sha49], on suppose l'existence d'un algorithme de chiffrement par blocs « parfait », c'est-à-dire uniformément aléatoire parmi tous les chiffrements par blocs possibles (pour une certaine taille de bloc et de clé). Plus précisément, on suppose l'existence d'une paire d'oracles $(\boldsymbol{E}, \boldsymbol{E}^{-1}) : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$, telle que pour toute clé $K \in \{0,1\}^k$, $\boldsymbol{E}(K,\cdot)$ est une permutation uniformément aléatoire parmi toutes les permutations de $\{0,1\}^n$ et $\boldsymbol{E}^{-1}(K,\cdot)$ son inverse. Tous les participants à un protocole cryptographique utilisant $\boldsymbol{E}$, y compris l'attaquant, peuvent faire des requêtes quelconques au chiffrement par blocs idéal, pour des messages ou chiffrés et des clés de leur choix. Intuitivement, ce modèle considère que la spécification du chiffrement par blocs utilisé est publiquement connue (par exemple, AES [Nat01]), mais que la structure interne de cet algorithme est trop complexe pour être utilisée à son avantage par l'attaquant, ce dernier étant réduit à effectuer des chiffrements et des déchiffrements pour des messages et des clés arbitraires (puisque la spécification de l'algorithme est publique), et à obtenir des réponses lui paraissant aléatoires.

L'hypothèse de sécurité standard pour un algorithme de chiffrement par blocs est qu'il constitue une famille de permutations pseudo-aléatoires inversibles [LR86]. Informellement, cela signifie que la permutation associée à une clé aléatoire secrète est indistinguable d'une permutation aléatoire inversible par tout algorithme efficace. Remarquons que le chiffrement par blocs idéal est un objet bien plus puissant qu'une famille de permutations pseudo-aléatoires inversibles. Ainsi, un chiffrement par blocs idéal ne souffre pas de problèmes d'attaques par clés reliées [Bih94], de clés faibles, ou de propriétés non aléatoires telles que la propriété de complémentation de clé de DES [MvOV96, Chapitre 8].

L'hypothèse qu'un chiffrement par blocs constitue une famille de permutations pseudo-aléatoires inversibles est généralement suffisante dans les applications de chiffrement à clé secrète [BDJR97]. Mais il existe des applications où elle ne l'est plus, notamment la construction de fonctions de hachage fondées sur un algorithme de chiffrement par blocs. Comme l'a montré Simon [Sim98], il est impossible de construire des fonctions de hachage résistantes aux collisions à partir de la simple hypothèse de l'existence d'une famille de permutations pseudo-aléatoires. C'est pourquoi le modèle du chiffrement par blocs idéal a été largement utilisé pour analyser les constructions de fonctions de hachage fondées sur

un algorithme de chiffrement par blocs. On en trouve la première trace dans la preuve de la résistance à la pré-image de la construction de Davies-Meyer par Winternitz [Win84] puis dans [Mer89, BRS02, Hir04, Hir06, Ste07]. Le modèle du chiffrement par blocs idéal est également utilisé de façon sporadique pour analyser des schémas de chiffrement symétriques [KR96, EM97, Des00, JJV02], ou des cryptosystèmes plus complexes tels que les signatures d'anneau [RST01] ou les protocoles d'échange de clé authentifié [BPR00].

Tout comme celle du modèle de l'oracle aléatoire, l'utilisation du modèle du chiffrement par blocs idéal est sujette à de nombreuses réserves. La situation est même plus préoccupante pour ce dernier : en effet, les cryptographes sont plus enclins à admettre que le standard de hachage SHA-2 fournit une « bonne » instanciation d'un oracle aléatoire qu'à modéliser AES comme idéal (même si l'hypothèse qu'AES est une bonne famille de permutations pseudo-aléatoires est généralement admise). La principale raison est qu'AES est susceptible de posséder des propriétés non aléatoires, particulièrement lorsque la clé est connue (mais ne remettant pas en cause le fait qu'il forme une famille de permutations pseudo-aléatoires). Très récemment, Biryukov, Khovratovich et Nikolić ont découvert des propriétés de l'algorithme AES-256 le distinguant d'un chiffrement par blocs idéal, notamment des attaques par clés reliées [BKN09].

Comme pour le modèle de l'oracle aléatoire, il est possible de construire des schémas cryptographiques artificiels qui sont sûrs dans le modèle du chiffrement par blocs idéal, mais qui deviennent vulnérables dès que l'on substitue au chiffrement par blocs idéal un algorithme concret (*i.e.* implémentable avec des ressources polynomiales en k, le paramètre de sécurité). Ainsi, Black [Bla06] a exhibé une fonction de hachage fondée sur un algorithme de chiffrement par blocs qui peut être prouvée résistante aux collisions dans le modèle du chiffrement par blocs idéal, mais pour laquelle il devient trivial de trouver une collision dès que le chiffrement par blocs est instancié par un algorithme réel. Comme pour les exemples d'ininstanciabilité de l'oracle aléatoire, la construction de Black est artificielle et conçue dès le départ pour fournir des collisions lorsque le chiffrement par blocs est instancié.

1.1.3 Le modèle de la permutation aléatoire inversible

Le modèle de la permutation aléatoire inversible est très similaire au modèle du chiffrement par blocs idéal, mais avec un espace de clés de cardinal 1. Au lieu de postuler l'existence d'une famille de permutations aléatoires indexée par un ensemble de clés, on ne suppose l'existence que d'une seule permutation aléatoire inversible $(\boldsymbol{P}, \boldsymbol{P}^{-1})$.

Moins populaire que les deux précédents modèles (certainement du fait que son instanciation est moins immédiate), il est cependant utilisé en cryptographie asymétrique pour prouver la sécurité de certains schémas de signature [Gra02] ou de chiffrement [PP03, CMPP05], souvent comme une première étape avant de remplacer la permutation par une construction similaire à OAEP.

1.2 Indifférentiabilité versus indistinguabilité

1.2.1 Un mot sur les notations et le vocabulaire

Étant donné un ensemble $\mathbb{G}$ fini, nous noterons G un élément quelconque de $\mathbb{G}$ et $\boldsymbol{G}$ la distribution uniforme sur cet ensemble. En général nous désignerons $\mathbb{G}$ sous le terme de *fonctionnalité* pour refléter le fait que les objets considérés seront des fonctions ou des permutations et serviront de brique de base dans un cryptosystème plus complexe. Ainsi, nous serons amenés à considérer la fonctionnalité des fonctions de $\{0,1\}^n$ vers $\{0,1\}^m$,

celle des permutations de $\{0,1\}^n$, ou encore celle des familles de permutations de $\{0,1\}^n$ indexées par l'ensemble $\{0,1\}^k$ (la fonctionnalité des chiffrements par bloc).

Nous considérerons à de nombreuses reprises des machines de Turing $\mathcal{M}$ accédant à un (ou plusieurs) oracle G, ce que nous noterons classiquement $\mathcal{M}^G$. Lorsque la fonctionnalité considérée sera celle des permutations de $\{0,1\}^n$, nous dirons que la permutation est *non inversible* pour signifier que la machine ne peut faire de requêtes qu'à P, et *inversible* pour signifier que la machine peut faire des requêtes à P et P^{-1}. Cependant nous noterons simplement $\mathcal{M}^P$ dans tous les cas ; le fait que l'oracle de permutation soit inversible ou non sera clair d'après le contexte.

À strictement parler, $\boldsymbol{G}$ est une distribution de probabilité sur $\mathbb{G}$, mais il nous arrivera de parler de requêtes d'une machine $\mathcal{M}$ à un oracle $\boldsymbol{G}$ pour signifier que $\mathcal{M}$ a accès à un oracle G, G étant uniformément aléatoire dans $\mathbb{G}$. Ceci reflète également le point de vue appelé *lazy sampling* en anglais (ce que l'on pourrait traduire par « échantillonnage économe »), où l'oracle $\boldsymbol{G}$ est vu comme un objet dynamique tirant ses réponses aléatoirement au fur et à mesure qu'elles lui sont demandées, plutôt qu'un objet statique prédéterminé aléatoirement, et nous permettra de considérer des fonctionnalités de cardinal infini, comme la fonctionnalité $\mathbb{H}$ des fonctions de $\{0,1\}^*$ vers $\{0,1\}^n$, modélisant une fonction de hachage. $\boldsymbol{H}$ correspond alors à un oracle aléatoire.

La notation $\Pr\left[\mathcal{M}^{\boldsymbol{G}}(1^k)=1\right]$ dénotera alors la probabilité que $\mathcal{M}$ retourne 1 lorsqu'elle accède à un oracle G uniformément aléatoire dans $\mathbb{G}$ (la probabilité étant également prise, le cas échéant, sur l'aléa de $\mathcal{M}$). Le point de vue *lazy sampling* nous permettra de conserver la même notation lorsque $\mathbb{G}$ est infini.

$\boldsymbol{G}$ sera appelée la *primitive idéale* correspondant à la fonctionnalité $\mathbb{G}$. Étant donnée une autre fonctionnalité $\mathbb{F}$, on peut s'intéresser à l'implémentation des éléments G de $\mathbb{G}$ à l'aide d'éléments de $\mathbb{F}$. Nous dirons qu'une machine de Turing $\mathcal{C}$ accédant à un oracle F *implémente* la fonctionnalité $\mathbb{G}$ si pour tout $F \in \mathbb{F}$, il existe $G \in \mathbb{G}$ tel que pour tout x dans le domaine de G, $\mathcal{C}^F(x) = G(x)$. Nous appellerons $\mathcal{C}$ une *construction*. On pourra alors interpréter $\mathcal{C}^{\boldsymbol{F}}$ comme la distribution de probabilité induite sur $\mathbb{G}$ par la distribution uniforme $\boldsymbol{F}$ sur $\mathbb{F}$. Les exemples de telles constructions sont nombreux en cryptographie : la construction CBC-MAC [BKR00] fournit une famille de fonctions de $\{0,1\}^*$ vers $\{0,1\}^n$ indexée par $\{0,1\}^k$ à partir d'une famille de permutations de $\{0,1\}^n$ indexée par $\{0,1\}^k$, la construction de Merkle-Damgård [Dam89, Mer89] fournit une fonction de $\{0,1\}^*$ vers $\{0,1\}^n$ à partir d'une fonction de $\{0,1\}^{n+m}$ vers $\{0,1\}^n$, etc. Un exemple sur lequel nous nous attarderons longuement est la construction de Luby-Rackoff, fournissant une permutation de $\{0,1\}^{2n}$ à partir de r fonctions de $\{0,1\}^n$ vers $\{0,1\}^n$.

Dans tout ce qui suit, lorsque nous parlerons de primitive idéale, il s'agira de façon implicite de familles de primitives indexées par un paramètre de sécurité $k \in \mathbb{N}$. Alternativement, on pourra considérer que la primitive prend le paramètre de sécurité comme entrée additionnelle.

Le reste de ce chapitre va nous permettre d'introduire les notions nécessaires pour comparer la sécurité d'un cryptosystème accédant à une fonctionnalité $\mathbb{G}$ lorsqu'il est utilisé avec la primitive idéale $\boldsymbol{G}$ et lorsqu'il est utilisé avec une construction $\mathcal{C}^{\boldsymbol{F}}$ implémentant la fonctionnalité $\mathbb{G}$.

1.2.2 Sécurité des cryptosystèmes

L'approche usuelle pour étudier la sécurité d'un cryptosystème consiste à modéliser l'ensemble des participants au protocole considéré comme des *machines de Turing inter-*

actives (que nous abrégerons par MTI). Nous renvoyons le lecteur à [Gol01, Chapitre 4] pour une définition précise de cette notion. Par la suite, nous noterons Γ l'ensemble des participants honnêtes (*i.e.* ne déviant pas du comportement normal) à un protocole cryptographique et dénommerons la MTI résultante *cryptosystème*. Le cryptosystème interagit avec une MTI $\mathcal{A}$ que nous dénommerons *attaquant* et représentant l'ensemble des entités ne suivant pas nécessairement les spécifications du protocole et ayant un comportement arbitraire et des « intentions » vraisemblablement hostiles. Enfin, Γ et $\mathcal{A}$ interagissent avec une MTI à sortie binaire $\mathcal{E}$ dénommée *environnement*, représentant le reste des entités algorithmiques susceptibles d'exister. La distance statistique entre les distributions de la sortie de $\mathcal{E}$ dans deux situations distinctes servira à « mesurer » à quel point ces deux situations sont éloignées. Par la suite, nous noterons $\langle \mathcal{E}, \Gamma, \mathcal{A} \rangle(1^k)$ la sortie de $\mathcal{E}$ lorsqu'il interagit avec Γ et $\mathcal{A}$, pour le paramètre de sécurité commun 1^k.

En général, un cryptosystème fait appel à une ou plusieurs fonctionnalités cryptographiques telles qu'un générateur de bits, une fonction, une permutation, une fonction de hachage, etc. Afin de prouver la sécurité du cryptosystème, on prouve tout d'abord qu'il est sûr lorsque le cryptosystème accède à la primitive idéale correspondant à la fonctionnalité, puis on montre (en général sous certaines hypothèses de complexité) que la primitive réelle est indistinguable de la primitive idéale. Par indistinguable on entend qu'aucun algorithme $\mathcal{D}$ à sortie binaire accédant à un oracle G et limité en nombre de requêtes à la primitive et/ou en temps de calcul (un tel algorithme est appelé *distingueur*) ne peut distinguer avec une probabilité non négligeable s'il interagit avec la primitive idéale $\boldsymbol{G}$ ou la primitive réelle $\{G_K\}_{K \in \mathcal{K}}$. Ainsi, on peut montrer que le masque jetable (« one-time pad » en anglais) constitue une technique de chiffrement sûre (au sens de la théorie de l'information) lorsque le masque est issu d'un générateur de bits idéal générant des bits uniformément aléatoires et indépendants. Il en découle que lorsque le masque est généré par un générateur de bits pseudo-aléatoire (c'est-à-dire indistinguable d'un générateur de bits idéal par tout algorithme efficace), le chiffrement à flot résultant est également sûr (contre tout attaquant efficace). Considérons plus en détail l'exemple important de la construction de Luby-Rackoff.

1.2.3 Indistinguabilité : exemple de la construction de Luby-Rackoff

Considérons la fonctionnalité $\mathbb{P}$ des permutations inversibles de $\{0,1\}^{2n}$. Cette fonctionnalité peut être implémentée à l'aide de la construction de Luby-Rackoff à r tours, qui utilise r fonctions $F_1, \ldots, F_r$ de $\{0,1\}^n$ vers $\{0,1\}^n$. La construction de Luby-Rackoff à un tour associée à F_1 est la permutation, notée $\Psi_1^{F_1}$, définie pour $L, R \in \{0,1\}^n$ par :

$$\Psi_1^{F_1}(L \| R) = R \| (L \oplus F_1(R)) \ .$$

La construction de Luby-Rackoff à r tours associée à $F = (F_1, \ldots, F_r)$ est la permutation notée Ψ_r^F définie par :

$$\Psi_r^F = \Psi_1^{F_r} \circ \Psi_1^{F_{r-1}} \circ \cdots \circ \Psi_1^{F_1} \ .$$

Remarque 1.1. La construction de Luby-Rackoff est également appelée *schéma de Feistel* [Fei73], cependant une légère différence sémantique existe : le terme schéma de Feistel est plus utilisé pour un chiffrement par blocs réel utilisant des fonctions (présumées) pseudo-aléatoires (comme le DES), alors qu'on préférera parler de construction de Luby-Rackoff dans le cadre théorique où les fonctions internes sont uniformément aléatoires. $*$

L'un des résultats les plus célèbres en cryptographie établit que la construction de Luby-Rackoff à 3 tours associée à des fonctions internes aléatoires $\boldsymbol{F} = (\boldsymbol{F}_1, \boldsymbol{F}_2, \boldsymbol{F}_3)$ est

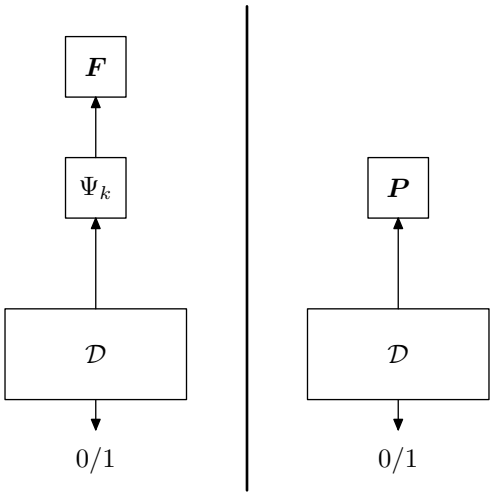

FIGURE 1.1 – La notion d'indistinguabilité pour la construction de Luby-Rackoff.

indistinguable d'une permutation aléatoire (non inversible), et que pour 4 tours elle est indistinguable d'une permutation aléatoire inversible [LR88, Pat90]. Ce résultat repose de façon essentielle sur le fait que les fonctions internes sont secrètes, comme l'illustre la figure 1.1. Plus formellement, on a le théorème suivant :

Théorème 1.1 ([LR88, Pat90]). *Pour tout distingueur $\mathcal{D}$ accédant à un oracle de permutation P (non inversible) auquel il effectue au plus q requêtes, on a :*

$$\left|\Pr\left[\mathcal{D}^{\Psi_3^{\boldsymbol{F}}}(1^k)=1\right]-\Pr\left[\mathcal{D}^{\boldsymbol{P}}(1^k)=1\right]\right|\leq\frac{q^2}{2^n}\ .$$

Pour tout distingueur $\mathcal{D}$ accédant à un oracle de permutation inversible P auquel il effectue au plus q requêtes, on a :

$$\left|\Pr\left[\mathcal{D}^{\Psi_4^{\boldsymbol{F}}}(1^k)=1\right]-\Pr\left[\mathcal{D}^{\boldsymbol{P}}(1^k)=1\right]\right|\leq\frac{q^2}{2^n}\ . \qquad \diamondsuit$$

Remarquons que le distingueur n'a pas accès aux fonctions internes $\boldsymbol{F}_i$ dans le cas où il interagit avec la construction de Luby-Rackoff. On serait d'ailleurs bien en peine de dire par quoi remplacer les fonctions $\boldsymbol{F}_i$ lorsque le distingueur interagit avec une permutation aléatoire $\boldsymbol{P}$ (en particulier, si on lui donne accès à des fonctions $\boldsymbol{F}_i$ indépendantes de $\boldsymbol{P}$, le système $(\boldsymbol{P},\boldsymbol{F})$ est trivialement distinguable de $(\Psi_{3,4}^{\boldsymbol{F}},\boldsymbol{F})$).

D'après ce théorème, prouver la sécurité d'un cryptosystème utilisant une construction de Luby-Rackoff revient donc à montrer qu'il est sûr lorsqu'il est utilisé avec une permutation aléatoire $\boldsymbol{P}$ (éventuellement inversible). En effet, si le cryptosystème était sûr avec $\boldsymbol{P}$ mais pas avec la construction de Luby-Rackoff, on pourrait combiner $(\mathcal{E},\Gamma,\mathcal{A})$ en un distingueur $\mathcal{D}$ capable de distinguer $\Psi_{3,4}^{\boldsymbol{F}}$ de $\boldsymbol{P}$. Ce raisonnement nécessite cependant *que les fonctions internes de la construction de Luby-Rackoff soient secrètes et à l'usage exclusif du cryptosystème.*

Le théorème 1.1 a été redémontré à de nombreuses reprises [Mau92, NR99], et la dépendance de l'avantage du distingueur en fonction du nombre de tours a été largement étudiée [MP03, Pat91, Pat98, Pat03, Pat04, Vau03].

1.2.4 Sécurité des cryptosystèmes utilisant une fonctionnalité publique

La méthodologie exposée ci-dessus, fondée sur l'indistinguabilité d'une construction, suppose que l'attaquant ne connaisse pas les « détails » internes de la primitive réelle (dans les exemples précédents, la clé secrète utilisée pour initialiser le générateur pseudo-aléatoire dans un chiffrement à flot ou les fonctions internes dans la construction de Luby-Rackoff). En revanche, cette méthodologie échoue dans le cas d'une fonction de hachage modélisée par un oracle aléatoire : comme l'ont montré Canetti *et al.* [CGH98], aucune famille de fonctions ne peut implémenter de façon satisfaisante un oracle aléatoire car le paramètre dont dépend la fonction de hachage utilisée (sa description) doit nécessairement être *public* (et donc connu également de l'attaquant) dans de nombreux cas.

De façon plus générale, lorsqu'on s'intéresse au cas où une fonctionnalité $\mathbb{G}$ est implémentée à l'aide d'une autre fonctionnalité $\mathbb{F}$ plus simple et que cette fonctionnalité $\mathbb{F}$ est publique, la notion d'indistinguabilité entre la primitive idéale $\boldsymbol{G}$ et une construction utilisant la primitive idéale $\boldsymbol{F}$ n'a plus de sens car un attaquant peut obtenir de l'information supplémentaire grâce à l'accès à $\boldsymbol{F}$ qu'il n'a pas lorsque la primitive idéale $\boldsymbol{G}$ est utilisée. Il est donc nécessaire d'élargir la notion d'indistinguabilité. Comme nous le verrons par la suite quand nous définirons la notion d'indifférentiabilité, le formalisation du fait que l'accès à $\boldsymbol{F}$ n'apporte pas d'information à un attaquant passe par la description d'un *simulateur* capable de simuler la primitive $\boldsymbol{F}$ à partir de la seule primitive $\boldsymbol{G}$.

Dans ce cadre, une question naturelle est de savoir si une primitive idéale $\boldsymbol{G}$ est strictement plus forte qu'une autre primitive idéale $\boldsymbol{F}$, dans le sens où il existe des fonctions cryptographiques implémentables de façon sûre à l'aide de $\boldsymbol{G}$ mais pas à l'aide de $\boldsymbol{F}$ [1]. La définition suivante, introduite par Maurer *et al.* [MRH04] et inspirée du cadre de la « Composabilité Universelle » (UC pour *Universal Composability* en anglais) de Canetti [Can00, Can01], permet de formaliser le fait qu'un cryptosystème soit aussi sûr lorsqu'il est implémenté avec $\mathcal{C}^{\boldsymbol{F}}$ qu'avec $\boldsymbol{G}$, lorsque la primitive idéale $\boldsymbol{F}$ est publique.

Définition 1.1 (Cryptosystème au moins aussi sûr)
Soit Γ un cryptosystème utilisant une fonctionnalité $\mathbb{G}$. Soit $\boldsymbol{G}$ la primitive idéale correspondant à la fonctionnalité $\mathbb{G}$, et soit $\mathcal{C}$ une construction utilisant une primitive idéale $\boldsymbol{F}$ publique et implémentant la fonctionnalité $\mathbb{G}$. Le cryptosystème $\Gamma^{\mathcal{C}^{\boldsymbol{F}}}$ est dit *au moins aussi sûr* que le cryptosystème $\Gamma^{\boldsymbol{G}}$ si pour tout environnement $\mathcal{E}$, et pour tout attaquant $\mathcal{A}$ interagissant avec le cryptosystème $\Gamma^{\mathcal{C}^{\boldsymbol{F}}}$ et ayant accès à la primitive idéale $\boldsymbol{F}$, à laquelle il fait un nombre de requêtes $q \in \mathtt{poly}(k)$, il existe un attaquant $\mathcal{A}'$, interagissant avec le cryptosystème $\Gamma^{\boldsymbol{G}}$ et ayant accès à la primitive idéale $\boldsymbol{G}$, à laquelle il fait un nombre de requêtes $q' \in \mathtt{poly}(k)$, tel que :

$$\left|\Pr\left[\left\langle \mathcal{E}, \Gamma^{\mathcal{C}^{\boldsymbol{F}}}, \mathcal{A}^{\boldsymbol{F}}\right\rangle (1^k) = 1\right] - \Pr\left[\left\langle \mathcal{E}, \Gamma^{\boldsymbol{G}}, \mathcal{A}'^{\boldsymbol{G}}\right\rangle (1^k) = 1\right]\right| = \mathtt{negl}(k) \ . \qquad \blacklozenge$$

Cette définition signifie que tout attaquant (efficace) contre le cryptosystème $\Gamma^{\mathcal{C}^{\boldsymbol{F}}}$ implique l'existence d'un attaquant (efficace) contre $\Gamma^{\boldsymbol{G}}$ de probabilité de succès quasiment égale. Elle est formulée dans le cadre la théorie de l'information car le temps de calcul des attaquants n'est pas pris en compte. Seul leur nombre de requêtes aux oracles est limité. On obtient la version calculatoire de cette définition en requérant que les temps de calcul de $\mathcal{A}$ et $\mathcal{A}'$ soient également polynomiaux en k.

La notion de cryptosystème au moins aussi sûr implique immédiatement une notion de *réductibilité* entre primitives idéales.

1. Nous soulignons que la question de savoir comment remplacer la primitive idéale $\boldsymbol{F}$ par une implémentation réelle sans perte de sécurité est une question totalement orthogonale.

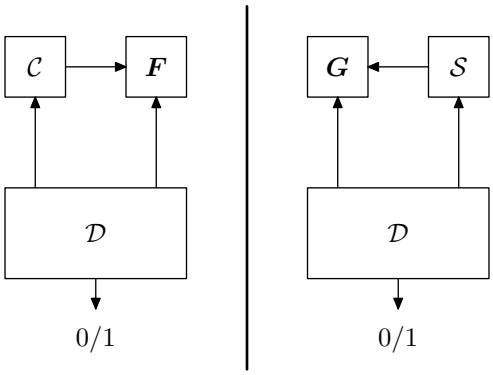

FIGURE 1.2 – La notion d'indifférentiabilité.

Définition 1.2 (Réductibilité)
Une primitive idéale $\boldsymbol{G}$ est dite *réductible* à une primitive idéale $\boldsymbol{F}$ s'il existe une construction $\mathcal{C}$ accédant à la primitive idéale $\boldsymbol{F}$ et implémentant la fonctionnalité $\mathbb{G}$, telle que pour tout cryptosystème Γ, $\Gamma^{\mathcal{C}^{\boldsymbol{F}}}$ est au moins aussi sûr que $\Gamma^{\boldsymbol{G}}$.
Deux primitives idéales $\boldsymbol{F}$ et $\boldsymbol{G}$ sont dites *équivalentes* si $\boldsymbol{G}$ est réductible à $\boldsymbol{F}$ et $\boldsymbol{F}$ est réductible à $\boldsymbol{G}$. ♦

Nous allons maintenant voir comment généraliser la notion d'indistinguabilité lorsqu'une construction utilise une fonctionnalité publique.

1.2.5 Définition générale de l'indifférentiabilité

Afin de généraliser la notion d'indistinguabilité au cas où une construction fait appel à des composants publics, Maurer *et al.* ont introduit la notion d'*indifférentiabilité* [MRH04]. Elle fait appel à un simulateur chargé d'émuler la primitive $\boldsymbol{F}$ lorsque le distingueur interagit avec la primitive idéale $\boldsymbol{G}$ et non la construction $\mathcal{C}^{\boldsymbol{F}}$. La notion de simulation est centrale dans les systèmes de preuves interactives sans transfert de connaissance introduits par Goldwasser *et al.* [GMR89]. Elle consiste à montrer qu'un attaquant n'acquiert aucun avantage notable en interagissant avec un certain objet en prouvant qu'il aurait pu le simuler par lui-même. La définition de Maurer *et al.* est formulée dans le cadre très général des *systèmes aléatoires*, modélisant de façon abstraite les distributions de probabilité des réponses successives d'une MTI. Nous nous contenterons de l'adaptation de ces définitions formulée par [CDMP05].

Définition 1.3 (Indifférentiabilité faible : $\forall\boldsymbol{\mathcal{D}}$, $\exists\boldsymbol{\mathcal{S}}$)
Soit $\boldsymbol{G}$ une primitive idéale correspondant à une fonctionnalité $\mathbb{G}$. Soit $\mathcal{C}$ une construction ayant accès à une primitive idéale $\boldsymbol{F}$ et implémentant la fonctionnalité $\mathbb{G}$. Soient q, $\sigma : \mathbb{N} \to \mathbb{N}$ et $\epsilon : \mathbb{N} \to \mathbb{R}^+$ trois fonctions du paramètre de sécurité k. $\mathcal{C}^{\boldsymbol{F}}$ est dite faiblement (q, σ, ϵ)-indifférentiable de $\boldsymbol{G}$ si pour toute MTI $\mathcal{D}$ ayant accès à deux oracles G et F, faisant au plus q requêtes au total, et à sortie binaire, il existe une MTI $\mathcal{S}$, appelée *simulateur*, ayant accès à la primitive idéale $\boldsymbol{G}$, faisant au plus σ requêtes à $\boldsymbol{G}$ lorsqu'elle interagit avec $\mathcal{D}$, et telle que :

$$\left|\Pr\left[\mathcal{D}^{\mathcal{C}^{\boldsymbol{F}},\boldsymbol{F}}(1^k) = 1\right] - \Pr\left[\mathcal{D}^{\boldsymbol{G},\mathcal{S}^{\boldsymbol{G}}}(1^k) = 1\right]\right| \leq \epsilon \ .$$

$\mathcal{C}^{\boldsymbol{F}}$ est simplement dite faiblement indifférentiable de $\boldsymbol{G}$ si pour tout $q \in \texttt{poly}(k)$, $\mathcal{C}^{\boldsymbol{F}}$ est faiblement (q, σ, ϵ)-indifférentiable de $\boldsymbol{G}$ avec $\sigma \in \texttt{poly}(k)$ et $\epsilon \in \texttt{negl}(k)$. ♦

Définition 1.4 (Indifférentiabilité forte : $\exists \mathcal{S}$, $\forall \mathcal{D}$)
Soit $\boldsymbol{G}$ une primitive idéale correspondant à une fonctionnalité $\mathbb{G}$. Soit $\mathcal{C}$ une construction ayant accès à une primitive idéale $\boldsymbol{F}$ et implémentant la fonctionnalité $\mathbb{G}$. Soient q ,σ : $\mathbb{N} \to \mathbb{N}$ et $\epsilon : \mathbb{N} \to \mathbb{R}^+$ trois fonctions du paramètre de sécurité k. $\mathcal{C}^{\boldsymbol{F}}$ est dite fortement (q, σ, ϵ)-indifférentiable de $\boldsymbol{G}$ s'il existe une MTI $\mathcal{S}$ ayant accès à la primitive idéale $\boldsymbol{G}$, telle que pour toute MTI $\mathcal{D}$ ayant accès à deux oracles G et F, faisant au plus q requêtes au total, et à sortie binaire, $\mathcal{S}$ fait au plus σ requêtes à $\boldsymbol{G}$ lorsqu'elle interagit avec $\mathcal{D}$, et telle que :

$$\left|\Pr\left[\mathcal{D}^{\mathcal{C}^{\boldsymbol{F}},\boldsymbol{F}}(1^k) = 1\right] - \Pr\left[\mathcal{D}^{\boldsymbol{G},\mathcal{S}^{\boldsymbol{G}}}(1^k) = 1\right]\right| \leq \epsilon \ .$$

$\mathcal{C}^{\boldsymbol{F}}$ est simplement dite fortement indifférentiable de $\boldsymbol{G}$ si pour tout $q \in \mathtt{poly}(k)$, $\mathcal{C}^{\boldsymbol{F}}$ est fortement (q, σ, ϵ)-indifférentiable de $\boldsymbol{G}$ avec $\sigma \in \mathtt{poly}(k)$ et $\epsilon \in \mathtt{negl}(k)$. ♦

Dans le cas où le distingueur n'a pas accès à l'oracle $\boldsymbol{F}$, on retrouve bien la notion classique d'indistinguabilité (il n'y a pas besoin d'introduire le simulateur pour remplacer l'oracle $\boldsymbol{F}$ dans le cas où $\mathcal{D}$ interagit avec $\boldsymbol{G}$).

On a évidemment l'implication suivante :

Lemme 1.2. *Si une construction $\mathcal{C}^{\boldsymbol{F}}$ est fortement indifférentiable de $\boldsymbol{G}$, alors $\mathcal{C}^{\boldsymbol{F}}$ est faiblement indifférentiable de $\boldsymbol{G}$.* ▽

Par la suite, lorsque nous parlerons d'indifférentiabilité sans précision, il s'agira d'indifférentiabilité forte.

Les définitions ci-dessus sont formulées dans le cadre la théorie de l'information car le temps de calcul du distingueur et du simulateur n'est pas pris en compte. Seul le nombre de requêtes de $\mathcal{D}$ et $\mathcal{S}$ aux oracles est limité. On peut obtenir une version calculatoire de ces définitions en ne considérant que des algorithmes efficaces, c'est-à-dire en requérant que le distingueur et le simulateur aient une complexité $\mathtt{poly}(k)$.

1.2.6 Application à la sécurité des cryptosystèmes

On peut montrer que l'indifférentiabilité (faible) est exactement la notion requise pour qu'une construction $\mathcal{C}^{\boldsymbol{F}}$ puisse remplacer une primitive idéale $\boldsymbol{G}$ dans tout cryptosystème sans perte de sécurité.

Théorème 1.3 ([MRH04]). *Soit Γ un cryptosystème utilisant une fonctionnalité $\mathbb{G}$. Soit $\mathcal{C}$ une construction ayant accès à une primitive idéale $\boldsymbol{F}$ et implémentant la fonctionnalité $\mathbb{G}$. Alors $\Gamma^{\mathcal{C}^{\boldsymbol{F}}}$ est au moins aussi sûr que $\Gamma^{\boldsymbol{G}}$ pour tout cryptosystème Γ si et seulement si $\mathcal{C}^{\boldsymbol{F}}$ est faiblement indifférentiable de $\boldsymbol{G}$.* ♢

Démonstration. Supposons tout d'abord que $\mathcal{C}^{\boldsymbol{F}}$ est faiblement indifférentiable de $\boldsymbol{G}$. Soit $\mathcal{E}$ un environnement, Γ un cryptosystème, et $\mathcal{A}$ un attaquant contre $\Gamma^{\mathcal{C}^{\boldsymbol{F}}}$ effectuant un nombre $q \in \mathtt{poly}(k)$ de requêtes à $\boldsymbol{F}$. Considérons la combinaison des machines Γ, $\mathcal{A}$ et $\mathcal{E}$ comme un unique distingueur $\mathcal{D}$ (cf. figure 1.3). Par hypothèse, il existe un simulateur $\mathcal{S}$, effectuant un nombre de requêtes $\sigma \in \mathtt{poly}(k)$ à $\boldsymbol{G}$, tel que la distribution de la sortie du distingueur est négligeablement proche lorsqu'il interagit avec $(\mathcal{C}^{\boldsymbol{F}}, \boldsymbol{F})$ et $(\boldsymbol{G}, \mathcal{S}^{\boldsymbol{G}})$. Par conséquent, si l'on considère l'attaquant $\mathcal{A}'$ obtenu en combinant $\mathcal{A}$ et $\mathcal{S}$ (cf. figure 1.3), alors $\mathcal{A}'$ fait $\sigma = \mathtt{poly}(k)$ requêtes à $\boldsymbol{G}$, et la distribution de la sortie de l'environnement est négligeablement proche lorsqu'il interagit avec $(\Gamma^{\mathcal{C}^{\boldsymbol{F}}}, \mathcal{A}^{\boldsymbol{F}})$ et $(\Gamma^{\boldsymbol{G}}, \mathcal{A}'^{\boldsymbol{G}})$, en d'autres termes $\Gamma^{\mathcal{C}^{\boldsymbol{F}}}$ est au moins aussi sûr que $\Gamma^{\boldsymbol{G}}$.

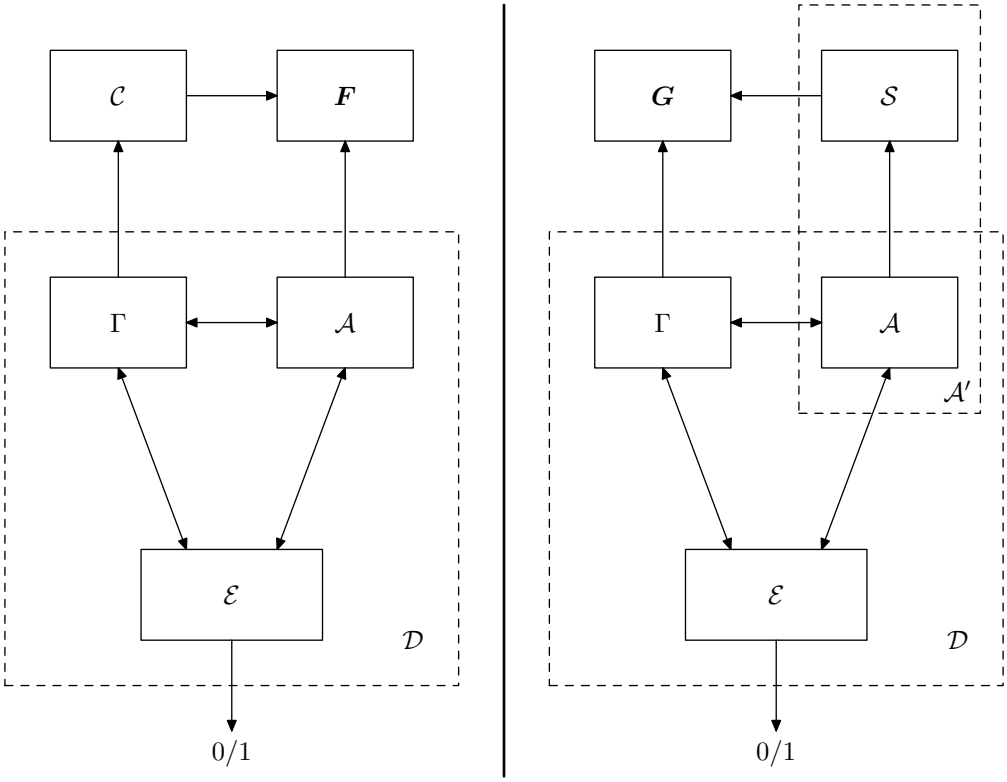

FIGURE 1.3 – Illustration de la preuve du théorème 1.3, première implication.

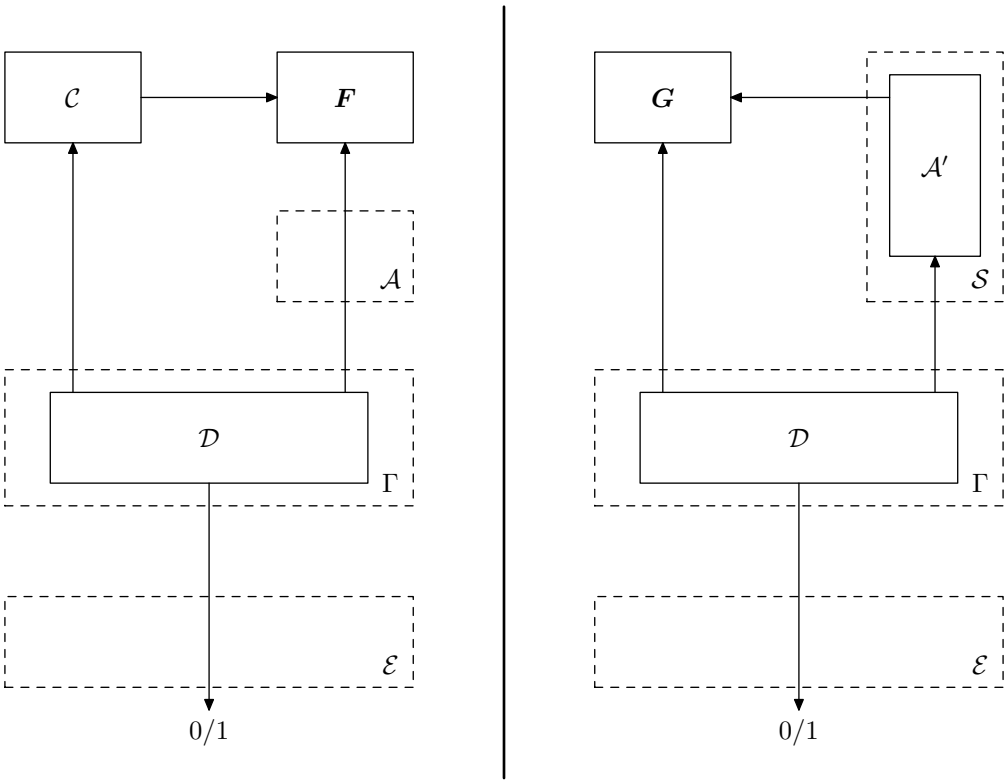

FIGURE 1.4 – Illustration de la preuve du théorème 1.3, seconde implication.

Réciproquement, supposons que pour tout cryptosystème Γ, $\Gamma^{\mathcal{C}^{\boldsymbol{F}}}$ est au moins aussi sûr que $\Gamma^{\boldsymbol{G}}$. Soit $\mathcal{D}$ un distingueur quelconque faisant $q \in \mathtt{poly}(k)$ requêtes. Définissons le cryptosystème Γ comme égal au distingueur, l'attaquant $\mathcal{A}$ comme étant l'algorithme trivial transférant les requêtes de $\mathcal{D}$ à $\boldsymbol{F}$ (et faisant donc un nombre polynomial de requêtes à $\boldsymbol{F}$), et enfin l'environnement $\mathcal{E}$ comme étant l'algorithme trivial ne faisant que reproduire la sortie du distingueur (cf. figure 1.4). Alors il existe un attaquant $\mathcal{A}'$, faisant un nombre de requêtes $q' \in \mathtt{poly}(k)$ à $\boldsymbol{G}$, tel que la distribution de la sortie de l'environnement est négligeablement proche lorsqu'il interagit avec $(\Gamma^{\mathcal{C}^{\boldsymbol{F}}}, \mathcal{A}^{\boldsymbol{F}})$ et $(\Gamma^{\boldsymbol{G}}, \mathcal{A}'^{\boldsymbol{G}})$. Définissons alors le simulateur $\mathcal{S}$ comme étant identique à l'attaquant $\mathcal{A}'$. $\mathcal{S}$ fait $q' \in \mathtt{poly}(k)$ requêtes à $\boldsymbol{G}$. De plus, la sortie du distingueur étant égale à celle de l'environnement par définition de ce dernier, on en déduit que la sortie du distingueur est négligeablement proche lorsqu'il interagit avec $(\mathcal{C}^{\boldsymbol{F}}, \boldsymbol{F})$ et $(\boldsymbol{G}, \mathcal{S}^{\boldsymbol{G}})$, ce qui est bien la définition d'une construction indifférentiable (faiblement car le simulateur dépend du distingueur). ∎

On déduit facilement du théorème ci-dessus que la notion de réductibilité de la définition 1.2 peut être exprimée comme l'existence d'une certaine construction indifférentiable.

Corollaire 1.4. *Une primitive idéale $\boldsymbol{G}$ est réductible à une primitive idéale $\boldsymbol{F}$ si et seulement s'il existe une construction $\mathcal{C}$ accédant à la primitive idéale $\boldsymbol{F}$ et implémentant la fonctionnalité $\mathbb{G}$ telle que $\mathcal{C}^{\boldsymbol{F}}$ est faiblement indifférentiable de $\boldsymbol{G}$.* ▽

Même si l'indifférentiabilité faible suffit pour montrer qu'une construction $\mathcal{C}^{\boldsymbol{F}}$ peut remplacer une primitive idéale $\boldsymbol{G}$ sans perte de sécurité, en pratique on montre que la construction est fortement indifférentiable car il est en général plus simple de construire un seul simulateur indépendant du distingueur.

Insistons sur le fait que la définition de l'indifférentiabilité exige que le simulateur n'ait pas accès aux requêtes du distingueur à la primitive $\boldsymbol{G}$. En effet, même si la construction $\mathcal{C}$ et l'oracle $\boldsymbol{F}$ sont publiquement accessibles, les requêtes du cryptosystème à $\boldsymbol{G}$ peuvent potentiellement dépendre d'une valeur secrète (une clé symétrique ou privée) inconnue de l'attaquant. L'exemple de la section suivante va nous servir à illustrer ce point.

1.2.7 Exemple : MAC et Merkle-Damgård

Considérons l'exemple bien connu de l'attaque par extension de message sur un code d'authentification de message (MAC) fondé sur la construction de Merkle-Damgård. Un MAC est une fonction T qui associe à une clé K et un message M un code $T(K, M)$. Le MAC est considéré sûr s'il est inforgeable sous des attaques à messages choisis, c'est-à-dire qu'un attaquant pouvant demander le MAC de messages de son choix pour une clé K fixée et secrète, ne peut pas produire le MAC $T(K, M')$ d'un message dont il n'a pas demandé le MAC au cryptosystème. Pour simplifier, considérons que le MAC n'accepte en entrée que des clés de n bits et des messages de un ou deux blocs de n bits, et produit une valeur de $2n$ bits. Considérons alors la fonction de MAC suivante, utilisant une fonction $H : \{0,1\}^{2n} \cup \{0,1\}^{3n} \to \{0,1\}^{2n}$, et définie par :

$$T^H(K, M) = H(K \| M) \ .$$

Comparons le cas où T utilise un oracle aléatoire $\boldsymbol{H} : \{0,1\}^{2n} \cup \{0,1\}^{3n} \to \{0,1\}^{2n}$, et le cas où T utilise la construction de Merkle-Damgård (sans le renforcement consistant à suffixer la taille du message) fondée sur une fonction aléatoire publique $\boldsymbol{f} : \{0,1\}^{3n} \to \{0,1\}^{2n}$:

$$\begin{cases} \mathcal{C}^{\boldsymbol{f}}(K, M_1) = \boldsymbol{f}(0_n \| K \| M_1) \\ \mathcal{C}^{\boldsymbol{f}}(K, M_1 \| M_2) = \boldsymbol{f}(\boldsymbol{f}(0_n \| K \| M_1) \| M_2) \ . \end{cases}$$

$T^{\boldsymbol{H}}$ est trivialement sûre, tandis que $T^{\mathcal{C}^{\boldsymbol{f}}}$ est susceptible d'être attaquée de la façon suivante : l'attaquant demande le MAC d'un message d'un bloc quelconque M_1, soit $h_1 = T(K, M_1)$. Il peut alors, pour tout bloc M_2, calculer le MAC de $M_1 \| M_2$ grâce à l'oracle $\boldsymbol{f}$, car $T(K, M_1 \| M_2) = \boldsymbol{f}(h_1 \| M_2)$.

Ceci provient naturellement du fait que $\mathcal{C}^{\boldsymbol{f}}$ n'est pas indifférentiable d'un oracle aléatoire $\boldsymbol{H}$. On peut cependant vérifier que la construction $\mathcal{C}^{\boldsymbol{f}}$ est indifférentiable d'un oracle aléatoire $\boldsymbol{H}$ *lorsque le simulateur connait toutes les requêtes effectuées par le distingueur à* $\boldsymbol{H}$. Le simulateur (qui a accès à $\boldsymbol{H}$ et connait les requêtes effectuées par le distingueur à cet oracle) procède ainsi : lorsqu'il reçoit une requête $\boldsymbol{f}(a\|b\|c)$, il vérifie si $a = 0_n$. Si c'est le cas, il définit $\boldsymbol{f}(a\|b\|c) = \boldsymbol{H}(b\|c)$. Sinon, il vérifie si le distingueur a effectué une requête $a'\|b'$ à $\boldsymbol{H}$ telle que la réponse était $a\|b$. Si c'est le cas, il effectue la requête $\boldsymbol{H}(a'\|b'\|c) = a''\|b''$ et définit $\boldsymbol{f}(a\|b\|c) = a''\|b''$ ainsi que $\boldsymbol{f}(0_n\|a'\|b') = a\|b$, sinon il définit $\boldsymbol{f}(a\|b\|c)$ aléatoirement. On peut facilement se convaincre que les systèmes $(\mathcal{C}^{\boldsymbol{f}}, \boldsymbol{f})$ et $(\boldsymbol{H}, \mathcal{S}^{\boldsymbol{H}})$ sont indistinguables (en particulier, si le distingueur n'a pas encore fait de requête $a'\|b'$ à $\boldsymbol{H}$ telle que la réponse était $a\|b$, la probabilité qu'il en fasse une par la suite est négligeable, ce qui justifie qu'on puisse bien définir $\boldsymbol{f}(a\|b\|c)$ aléatoirement). Pourtant, cela ne permet pas de transformer l'attaque sur $T^{\mathcal{C}^{\boldsymbol{f}}}$ en une attaque sur $T^{\boldsymbol{H}}$ car, lorsque l'attaquant demande le MAC de M_1, il ne connait pas la requête $\boldsymbol{H}(K\|M_1)$ du cryptosystème à $\boldsymbol{H}$ car la clé K est secrète.

Enfin, la technique de simulation ci-dessus échoue si le simulateur ne connait pas les requêtes du distingueur à $\boldsymbol{H}$. L'attaque suivante était déjà décrite dans [CDMP05] : le distingueur demande $\boldsymbol{H}(a'\|b') = a\|b$, puis $f(a\|b\|c) = a''\|b''$ au simulateur puis enfin $\boldsymbol{H}(a'\|b'\|c)$. Cette fois le simulateur n'aura pas pu s'adapter pour que $\boldsymbol{H}(a'\|b'\|c) = a''\|b''$ car il ne connaissait pas la requête $\boldsymbol{H}(a'\|b')$ du distingueur.

La notion d'indifférentiabilité lorsque l'adversaire connait toutes les requêtes à la primitive G a récemment été introduite par Dodis *et al.* sous le nom d'oracle aléatoire *public* (*pub-RO*, pour *public-use Random Oracle*) [DRS09]. Elle est notamment suffisante pour étudier la sécurité de la plupart des schémas de signature.

1.3 Précédents résultats d'indifférentiabilité

1.3.1 Résultats négatifs

Lorsque Maurer *et al.* ont introduit la notion d'indifférentiabilité [MRH04], ils l'ont utilisée pour redémontrer les résultats de Canetti *et al.* [CGH98] quant à l'ininstanciabilité d'un oracle aléatoire, en prouvant qu'un oracle aléatoire n'est pas réductible à une chaine de bits de longueur finie. Ceci implique qu'il existe des cryptosystèmes sûrs lorsqu'ils sont utilisés avec un oracle aléatoire, mais vulnérables dès qu'ils sont utilisés avec une fonction décrite par une chaine de bits publique, *i.e.* une fonction de hachage, ce qui est exactement le résultat de Canetti *et al.* Notons cependant que la preuve de Maurer *et al.* est existentielle (et fondée sur des arguments d'entropie), alors que celle de Canetti *et al.* est constructive (on trouve dans [CGH98] la description d'un schéma de signature sûr avec un oracle aléatoire et vulnérable avec n'importe quelle fonction de hachage).

1.3.2 Réductibilité de l'oracle aléatoire au chiffrement par blocs idéal

Le premier résultat positif d'indifférentiabilité est dû à Coron *et al.* [CDMP05], qui ont montré que l'oracle aléatoire est réductible au chiffrement par blocs idéal. Rappelons qu'un algorithme de chiffrement par blocs $E : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$ peut être utilisé

pour construire une fonction de compression f [PGV93, BRS02], la construction la plus célèbre étant celle de Davies-Meyer :

$$\begin{array}{rl} f : & \{0,1\}^{n+k} \to \{0,1\}^n \\ & (x,y) \mapsto E(y,x) \oplus x \ . \end{array}$$

Pour prouver ce résultat, Coron *et al.* ont montré que quatre constructions, consistant en de légères variantes de la construction de Merkle-Damgård [Dam89, Mer89], utilisées avec une fonction de compression dérivée d'un algorithme de chiffrement par blocs idéal utilisé selon le mode Davies-Meyer, sont indifférentiables d'un oracle aléatoire. Leur analyse a été poursuivie de façon systématique par Chang *et al.* [CLNY06] et Chang et Nandi [CN08].

Ainsi, lorsqu'un cryptosystème est prouvé sûr avec un oracle aléatoire $\boldsymbol{H}$, il reste également sûr lorsqu'il est utilisé avec l'une des constructions mentionnées ci-dessus, utilisée avec un chiffrement par blocs idéal.

1.3.3 Autres résultats

L'indifférentiabilité d'un oracle aléatoire s'impose de plus en plus comme une propriété hautement désirable de toute nouvelle construction permettant d'étendre le domaine d'une fonction ayant une taille d'entrée finie. Bertoni *et al.* ont récemment introduit un nouveau concept pour le design de fonctions de hachage, celui de construction « éponge » [BDPA07], fournissant une fonction de $\{0,1\}^*$ vers $\{0,1\}^{n'}$ à partir d'une fonction ou d'une permutation f de $\{0,1\}^n$ vers $\{0,1\}^n$, $n' \leq n$. Ils ont montré par la suite [BDPA08] que cette construction est indifférentiable d'un oracle aléatoire lorsque la fonction ou la permutation f est aléatoire. Leur résultat montre que l'oracle aléatoire est réductible à une fonction aléatoire $\boldsymbol{f} : \{0,1\}^n \to \{0,1\}^n$ ([CDMP05] avait auparavant montré qu'il était réductible à une fonction de compression aléatoire $\boldsymbol{f} : \{0,1\}^{n+m} \to \{0,1\}^n$), mais également à une permutation aléatoire inversible de $\{0,1\}^n$.

Maurer et Tessaro [MT07] ont donné la première construction, à partir de fonctions de taille d'entrée finie, d'une fonction de hachage indifférentiable d'un oracle aléatoire avec une sécurité au-delà de la borne du paradoxe des anniversaires.

Parmi d'autres travaux étudiant des constructions de fonctions de hachage indifférentiables d'un oracle aléatoire, citons [BR06, DPP08, FLP08].

Terminons ce chapitre par une mise en garde : comme l'ont montré Bellare et Ristenspart [BR06], il faut veiller à ne pas « surinterpréter » les résultats d'indifférentiabilité. En particulier, dans le cas d'une construction $\mathcal{C}$ permettant d'étendre le domaine d'une fonction de compression $f : \{0,1\}^{n+m} \to \{0,1\}^n$ pour obtenir une fonction de hachage $H : \{0,1\}^* \to \{0,1\}^n$, le fait que $\mathcal{C}^{\boldsymbol{f}}$ soit indifférentiable d'un oracle aléatoire $\boldsymbol{H}$ ne garantit en rien que la construction $\mathcal{C}$, utilisée avec une famille de fonctions $\{f_K\}_{K \in \mathcal{K}}$ résistante aux collisions (ou possédant toute autre propriété désirable) fournisse une fonction de hachage résistante aux collisions ! En d'autres termes, l'indifférentiabilité d'une construction garantit une propriété de sécurité lorsque la construction est utilisée avec un composant idéal (ou lui-même indifférentiable de la primitive idéale correspondante), mais ne garantit plus rien lorsque les hypothèses sur la primitive utilisée sont affaiblies.

Chapitre 2

Équivalence entre les modèles de l'oracle aléatoire et du chiffrement par blocs idéal

Ce deuxième chapitre est consacré à la démonstration du principal résultat de cette première partie, à savoir le théorème suivant :

Théorème 2.1. *Le chiffrement par blocs idéal est réductible à l'oracle aléatoire.* ♢

Combiné au résultat de Coron *et al.* [CDMP05] établissant que l'oracle aléatoire est réductible au chiffrement par blocs idéal, on en déduit le corollaire suivant :

Théorème 2.2. *L'oracle aléatoire et le chiffrement par blocs idéal sont deux primitives idéales équivalentes.* ♢

La preuve du théorème 2.1 passe par l'exhibition d'une construction utilisant un oracle aléatoire et indifférentiable d'une permutation aléatoire inversible (nous verrons que le résultat pour un chiffrement par blocs idéal s'ensuit aisément). Dans l'article [CPS08], nous avons, avec Jean-Sébastien Coron et Jacques Patarin, montré le résultat pour la construction de Luby-Rackoff à 6 tours. Dans ce mémoire nous utilisons la construction de Luby-Rackoff à 10 tours car cela simplifie considérablement la preuve. Nous nous contenterons de donner les idées principales pour 6 tours.

2.1 Luby-Rackoff avec des fonctions publiques

Un candidat naturel pour transformer une fonction cryptographiquement sûre en permutation cryptographiquement sûre est la construction de Luby-Rackoff. En effet, comme nous l'avons vu à la section 1.2.3, cette construction, utilisée avec des fonctions internes pseudo-aléatoires, fournit une permutation pseudo-aléatoire pour 3 tours, et une permutation pseudo-aléatoire inversible pour 4 tours [LR88, Pat90]. Il est donc légitime d'étudier la sécurité de la construction de Luby-Rackoff lorsque les fonctions internes sont connues de l'attaquant. En particulier, la construction de Luby-Rackoff, avec un nombre suffisant de tours, est-elle indifférentiable d'une permutation aléatoire inversible ?

La question n'est pas entièrement nouvelle. Elle a déjà été considérée par Ramzan et Reyzin [RR00], qui ont montré que la construction de Luby-Rackoff à 4 tours Ψ_4^F, $F = (F_1, F_2, F_3, F_4)$, reste sûre (*i.e.* indistinguable d'une permutation aléatoire inversible)

lorsque l'attaquant a accès aux deux fonctions centrales F_2 ou F_3, mais devient vulnérable (*i.e.* distinguable d'une permutation aléatoire inversible) dès que l'attaquant a accès à l'une des deux fonctions externes F_1 ou F_4.

Dodis et Puniya ont introduit un modèle d'indifférentiabilité légèrement différent du modèle défini dans le chapitre précédent, appelé indifférentiabilité dans le modèle *honnête mais curieux* [DP06]. Dans ce modèle, qui fait l'objet d'une discussion détaillée à la section 2.5.1, l'attaquant ne peut pas faire de requêtes directement à la primitive idéale $\boldsymbol{F}$ à laquelle la construction $\mathcal{C}$ fait appel. Par contre il peut faire des requêtes à la construction $\mathcal{C}^{\boldsymbol{F}}$ et connaitre toutes les valeurs d'entrée/sortie de $\boldsymbol{F}$ dont a besoin la construction pour calculer la réponse (dans le cas de la construction de Luby-Rackoff, il s'agit des valeurs d'entrée et de sortie des fonction F_i internes). Dodis et Puniya ont montré que la construction de Luby-Rackoff avec un nombre de tours fonction super-logarithmique du paramètre de sécurité est indifférentiable, dans le modèle honnête mais curieux, d'une permutation aléatoire inversible. Ce résultat ne permet cependant pas de conclure pour le modèle d'indifférentiabilité général (cf. section 2.5.1). Dodis et Puniya ont également étudié diverses propriétés de la construction de Luby-Rackoff (telles que l'imprédictibilité et la vérifiabilité) lorsque les fonctions internes sont connues de l'attaquant [DP07].

Enfin, l'indifférentiabilité a des relations avec la notion de « résistance à la corrélation » introduite par Canetti *et al.* dans leur article sur l'ininstanciabilité de l'oracle aléatoire [CGH98], et celle de « distingueur à clé connue » introduite par Knudsen et Rijmen [KR07]. Nous reviendrons sur ce point à la section 2.5.2.

Nous allons voir dans la suite de ce chapitre que la construction de Luby-Rackoff fournit une construction indifférentiable d'un chiffrement par blocs idéal $\boldsymbol{E}$ à partir d'un oracle aléatoire $\boldsymbol{H}$. Remarquons que si l'on est capable d'exhiber une construction indifférentiable d'une permutation aléatoire inversible $\boldsymbol{P}$ à partir d'un oracle aléatoire $\boldsymbol{H} : \{0,1\}^* \to \{0,1\}^n$, on en déduit immédiatement une construction indifférentiable d'un chiffrement par blocs idéal, pour n'importe quel ensemble de clé $\mathcal{K}$: il suffit d'incorporer à chaque requête faite par la construction à l'oracle aléatoire $\boldsymbol{H}$ un préfixe égal à la clé, et on obtient bien des permutations indépendantes pour chaque clé.

Par ailleurs, étant donné un oracle aléatoire $\boldsymbol{H} : \{0,1\}^* \to \{0,1\}^n$, il est possible de construire r fonctions aléatoires indépendantes de $\{0,1\}^n$ vers $\{0,1\}^n$ en posant, pour $i \in [\![1,r]\!]$ et $U \in \{0,1\}^n$:

$$\boldsymbol{F}_i(U) = \boldsymbol{H}(\langle i \rangle \| U) \ ,$$

où $\langle i \rangle$ dénote la représentation binaire de i. Nous noterons simplement $\boldsymbol{F}$ les r fonctions ainsi définies et $\Psi_r^{\boldsymbol{F}}$ la construction de Luby-Rackoff à r tours associée à $\boldsymbol{F} = (\boldsymbol{F}_1, \ldots, \boldsymbol{F}_r)$.

Nous allons donc nous intéresser à la construction de Luby-Rackoff à r tours utilisant r fonctions aléatoires et indépendantes $\boldsymbol{F} = (\boldsymbol{F}_1, \ldots, \boldsymbol{F}_r)$ et à son indifférentiabilité d'une permutation aléatoire inversible en fonction du nombre de tours. Nous considérerons par la suite des distingueurs $\mathcal{D}$ ayant accès à un oracle de permutation inversible que nous noterons simplement P (sous-entendu P/P^{-1}), et à r oracles pour les fonctions $F_1, \ldots, F_r$ que nous noterons comme un oracle unique $F = (F_1, \ldots, F_r)$. Le distingueur doit s'efforcer de distinguer entre deux situations :

- la situation où F est constituée de r fonctions aléatoires et indépendantes et P est la construction de Luby-Rackoff à r tours associée : $\mathcal{D}^{\Psi_r^{\boldsymbol{F}},\boldsymbol{F}}(1^k)$;
- la situation où P est une permutation aléatoire inversible, et les fonctions F sont simulées par un simulateur $\mathcal{S}$ ayant accès à la permutation P : $\mathcal{D}^{\boldsymbol{P},\mathcal{S}^{\boldsymbol{P}}}(1^k)$.

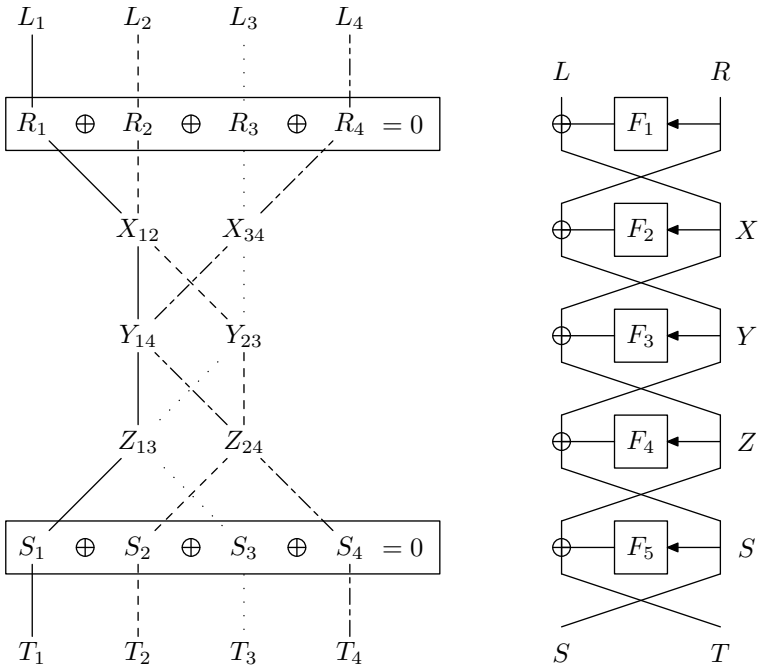

FIGURE 2.1 – Attaque sur la construction de Luby-Rackoff à 5 tours. Les valeurs (L, R, X, Y, Z, S, T) partageant un même indice et reliées par des pointillés de motif identique correspondent à la même valeur d'entrée/sortie de Ψ_5^F.

Les requêtes du distingueur à P/P^{-1} et F seront respectivement appelées P-requêtes et F-requêtes. Nous supposerons que le nombre total de requêtes du distingueur (P-requêtes et F-requêtes confondues) est inférieur à q.

Nous allons commencer par montrer que la construction de Luby-Rackoff à 5 tours n'est pas indifférentiable d'une permutation aléatoire inversible en exhibant une attaque due à Coron *et al.* [CJP02]. Puis nous montrerons que pour 10 tours, la construction est indifférentiable d'une permutation aléatoire inversible. Nous expliquerons ensuite les principales idées permettant de montrer que 6 tours sont en fait suffisants (et nécessaires d'après l'attaque pour 5 tours). Ces résultats contrastent avec les résultats classiques de Luby et Rackoff, rappelés dans le théorème 1.1, énonçant que 3 tours (resp. 4 tours) sont nécessaires et suffisants pour obtenir une permutation pseudo-aléatoire (resp. une permutation pseudo-aléatoire inversible) [LR88, Pat90] à partir de fonctions pseudo-aléatoires.

2.2 Attaque de la construction de Luby-Rackoff à 5 tours

Nous allons montrer que la construction de Luby-Rackoff à 5 tours $\Psi_5^{\boldsymbol{F}}$ n'est pas indifférentiable d'une permutation aléatoire inversible $\boldsymbol{P}$, en exhibant un distingueur possédant un avantage non négligeable quel que soit le simulateur.

Le distingueur $\mathcal{D}$ procède de la façon suivante (la figure 2.1 aidera à comprendre l'attaque). Il choisit une valeur Z_{13} arbitraire, deux valeurs Y_{14} et Y_{23} arbitraires, et effectue les requêtes $F_3(Y_{14})$ et $F_3(Y_{23})$. Puis il calcule les deux valeurs :

$$\begin{cases} X_{12} = Z_{13} \oplus F_3(Y_{14}) \\ X_{34} = Z_{13} \oplus F_3(Y_{23}) \end{cases} .$$

Les notations sont choisies de telle sorte que les valeurs partageant un même indice correspondent à la même entrée/sortie de Ψ_5^F. On dira aussi qu'elles « appartiennent à une même chaine » : ainsi, (X_{12}, Y_{14}, Z_{13}) appartiennent à une même chaine car $X_{12} = Z_{13} \oplus F_3(Y_{14})$.

Le distingueur effectue ensuite les requêtes $F_2(X_{12})$ et $F_2(X_{34})$ et calcule :

$$\begin{cases} R_1 = Y_{14} \oplus F_2(X_{12}) \\ R_2 = Y_{23} \oplus F_2(X_{12}) \\ R_3 = Y_{23} \oplus F_2(X_{34}) \\ R_4 = Y_{14} \oplus F_2(X_{34}) \end{cases} .$$

Remarquons tout de suite que par construction $R_1 \oplus R_2 \oplus R_3 \oplus R_4 = 0$. Puis le distingueur effectue les requêtes $F_1(R_1)$, $F_1(R_2)$, $F_1(R_3)$ et $F_1(R_4)$ et calcule :

$$\begin{cases} L_1 = X_{12} \oplus F_1(R_1) \\ L_2 = X_{12} \oplus F_1(R_2) \\ L_3 = X_{34} \oplus F_1(R_3) \\ L_4 = X_{34} \oplus F_1(R_4) \end{cases} .$$

Enfin, le distingueur effectue les requêtes $S_1 \| T_1 = P(L_1 \| R_1)$, $S_2 \| T_2 = P(L_2 \| R_2)$, $S_3 \| T_3 = P(L_3 \| R_3)$ et $S_4 \| T_4 = P(L_4 \| R_4)$. Si $S_1 \oplus S_2 \oplus S_3 \oplus S_4 = 0$, le distingueur retourne 1, sinon il retourne 0.

Tout d'abord, on peut facilement vérifier que la probabilité que $\mathcal{D}$ retourne 1 lorsqu'il interagit avec le système $(\Psi_5^{\boldsymbol{F}}, \boldsymbol{F})$ vaut 1. En effet, dans ce cas, soit $Z_{24} = X_{12} \oplus F_3(Y_{23})$ la valeur d'entrée de F_4 associée à $L_2 \| R_2$. Comme $X_{12} \oplus F_3(Y_{14}) = X_{34} \oplus F_3(Y_{23}) = Z_{13}$, on a $Z_{24} = X_{34} \oplus F_3(Y_{14})$, si bien que Z_{24} est également la valeur d'entrée de F_4 associée à $L_4 \| R_4$. Par conséquent, on a :

$$\begin{cases} S_1 = Y_{14} \oplus F_4(Z_{13}) \\ S_2 = Y_{23} \oplus F_4(Z_{24}) \\ S_3 = Y_{23} \oplus F_4(Z_{13}) \\ S_4 = Y_{14} \oplus F_4(Z_{24}) \end{cases} ,$$

si bien que la relation $S_1 \oplus S_2 \oplus S_3 \oplus S_4 = 0$ est toujours vérifiée. Ainsi, le distingueur « force une collision » en Z_{13} et obtient quatre chaines qui collisionnent au niveau des entrées de F_2, F_3 et F_4, entrainant des corrélations entre les valeurs de R et S correspondantes.

Pour borner la probabilité p que le distingueur retourne 1 lorsqu'il interagit avec $(\boldsymbol{P}, \mathcal{S}^{\boldsymbol{P}})$, on peut remarquer que si le simulateur effectue au plus $\sigma(q)$ requêtes à la permutation lorsqu'il interagit avec un distingueur lui soumettant q requêtes, on peut combiner $\mathcal{D}$ et $\mathcal{S}$ en un algorithme $\mathcal{A}$ qui effectue au plus $(4 + \sigma(8))$ requêtes à $\boldsymbol{P}$ et trouve 4 couples $S_1 \| T_1 = \boldsymbol{P}(L_1 \| R_1)$, $S_2 \| T_2 = \boldsymbol{P}(L_2 \| R_2)$, $S_3 \| T_3 = \boldsymbol{P}(L_3 \| R_3)$ et $S_4 \| T_4 = \boldsymbol{P}(L_4 \| R_4)$ tels que $R_1 \oplus R_2 \oplus R_3 \oplus R_4 = 0$ et $S_1 \oplus S_2 \oplus S_3 \oplus S_4 = 0$ avec probabilité p. On peut facilement se convaincre que cette probabilité est une quantité négligeable en n, donc en k pour $n \in \mathtt{poly}(k)$. Par conséquent

$$\left| \Pr\left[\mathcal{D}^{\Psi_5^{\boldsymbol{F}}, \boldsymbol{F}}(1^k) = 1\right] - \Pr\left[\mathcal{D}^{\boldsymbol{P}, \mathcal{S}^{\boldsymbol{P}}}(1^k) = 1\right] \right| \geq 1 - \mathtt{negl}(k) .$$

Il existe donc un distingueur possédant un avantage non négligeable (et même exponentiellement proche de un) contre tout simulateur effectuant un nombre polynomial de requêtes. On a donc prouvé le résultat suivant :

Théorème 2.3. *La construction de Luby-Rackoff à 5 tours $\Psi_5^{\boldsymbol{F}}$ n'est pas indifférentiable (pas même faiblement) d'une permutation aléatoire inversible $\boldsymbol{P}$.* ♢

2.3 Indifférentiabilité de la construction de Luby-Rackoff à 10 tours

Nous allons maintenant montrer que la construction de Luby-Rackoff à 10 tours $\Psi_{10}^{\boldsymbol{F}}$ est indifférentiable d'une permutation aléatoire inversible. Pour cela, nous allons décrire un simulateur $\mathcal{S}$ tel que les systèmes $(\Psi_{10}^{\boldsymbol{F}}, \boldsymbol{F})$ et $(\boldsymbol{P}, \mathcal{S}^{\boldsymbol{P}})$ sont indistinguables.

Le simulateur $\mathcal{S}$ doit s'assurer que ses réponses aux F-requêtes du distingueur $\mathcal{D}$ sont cohérentes avec les réponses aux P-requêtes que le distingueur peut obtenir indépendamment, c'est-à-dire que la permutation Ψ_{10} définie par les réponses du simulateur aux F-requêtes du distingueur est égale à la permutation aléatoire $\boldsymbol{P}$ (pour les valeurs d'entrée/sortie de Ψ_{10} calculables à l'aide des F-requêtes du distingueur). La tâche est compliquée par le fait que le simulateur ne connait pas les P-requêtes du distingueur, et doit donc s'efforcer d'être cohérent avec toutes les P-requêtes potentielles. Pour cela, le simulateur est autorisé à faire ses propres requêtes à la permutation aléatoire $\boldsymbol{P}$. De plus, les réponses du simulateur aux F-requêtes du distingueur doivent être statistiquement indistinguables des réponses de fonctions aléatoires et indépendantes.

Dans toute la suite, les probabilités seront prises sur la permutation aléatoire $\boldsymbol{P}$ et l'aléa du simulateur. Le distingueur étant supposé de capacité de calcul illimitée, on le supposera déterministe sans perte de généralité. Avant de décrire le simulateur, il nous faut introduire quelques définitions et notations.

2.3.1 Notations et définitions

Les notations que nous utiliserons pour désigner les entrées des différentes fonctions F_i de Ψ_{10} sont indiquées sur la figure 2.2.

Le simulateur $\mathcal{S}$ maintient un historique des valeurs des oracles F_i déjà définies (soit parce qu'elles ont été demandées lors d'une requête du distingueur, soit parce qu'il les a définies de façon interne, « par anticipation »). Nous noterons F_i, $i \in [\![1, 10]\!]$, l'historique ainsi défini, c'est-à-dire l'ensemble des paires (U, V) telles que F_i a été définie en U par $F_i(U) = V$. Par abus de notation nous noterons $U \in F_i$ s'il existe $(U', V') \in F_i$ tel que $U = U'$. Nous noterons $F_i(U) \leftarrow V$ pour signifier que la paire (U, V) est ajoutée à F_i, et $F_i(U) \xleftarrow{\$} \{0,1\}^n$ pour signifier que la paire (U, V) est ajoutée à F_i, où V est uniformément aléatoire dans $\{0,1\}^n$. Nous noterons $\mathcal{H}$ l'historique complet, *i.e.* $\mathcal{H} = (F_1, F_2, \ldots, F_9, F_{10})$.

Afin que ses réponses soient cohérentes avec la permutation $\boldsymbol{P}$, le simulateur complète en avance les chaines qui se forment dans son historique sous l'effet des F-requêtes du distingueur. Intuitivement, le distingueur peut tenter de former une chaine en commençant « par l'extérieur », c'est-à-dire en faisant d'abord la P-requête $\boldsymbol{P}(L\|R) = S\|T$, puis en faisant les F-requêtes $F_1(R)$, $F_2(W)$ avec $W = L \oplus F_1(R)$, etc. et $F_{10}(S)$, $F_9(D)$ avec $D = T \oplus F_{10}(S)$, etc., ou « par le centre », c'est-à-dire en faisant les F-requêtes $F_5(Z)$ et $F_6(A)$, puis $F_4(Y)$ avec $Y = A \oplus F_5(Z)$, $F_7(B)$ avec $B = Z \oplus F_6(A)$, etc. Les définitions qui suivent vont permettre de formaliser cette idée.

Définition 2.1 (Chaine externe)
Étant donnés un historique $\mathcal{H}$ et une permutation $\boldsymbol{P}$, on dira qu'un quadruplet

$$(W, R, S, D) \in F_2 \times F_1 \times F_{10} \times F_9$$

forme une *chaine externe* si et seulement si $\boldsymbol{P}(L\|R) = S\|T$, avec $L = W \oplus F_1(R)$ et $T = D \oplus F_{10}(S)$.

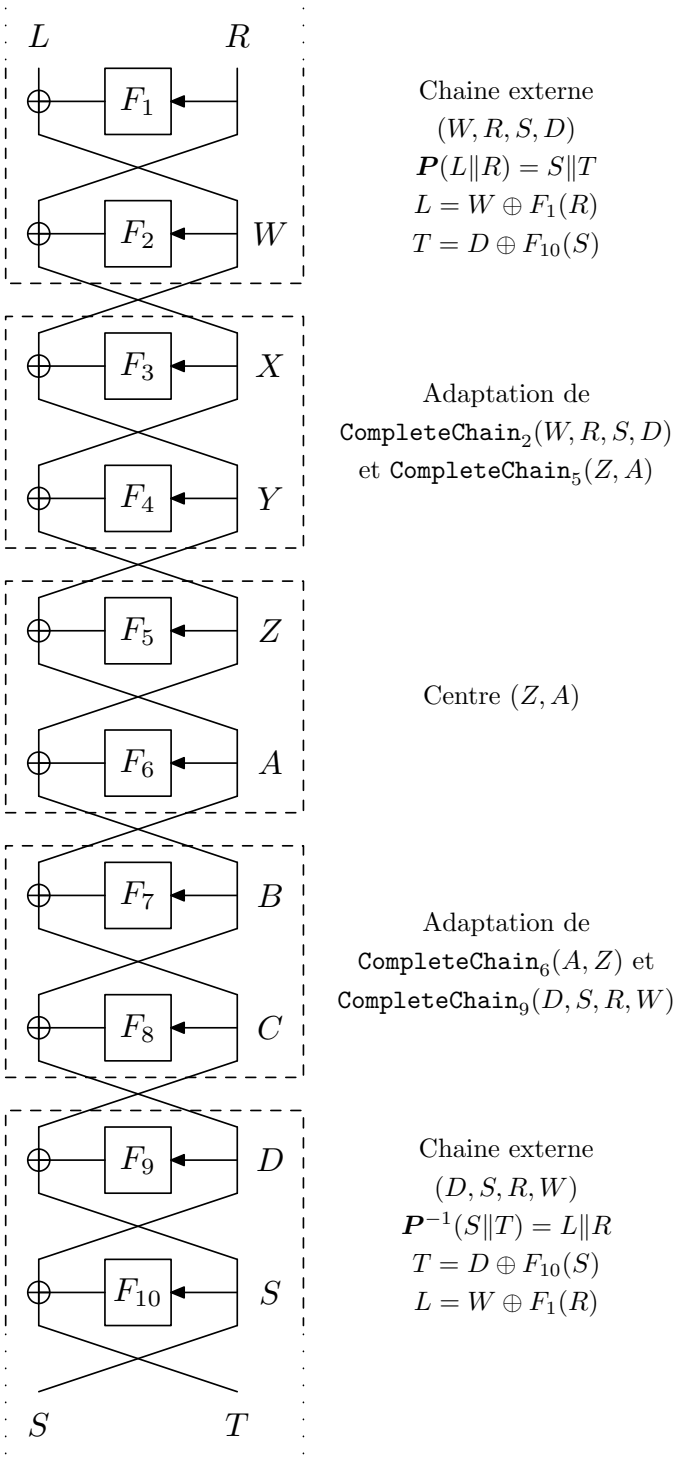

FIGURE 2.2 – Notations pour la construction de Luby-Rackoff à 10 tours.

On dira qu'une chaine externe (W, R, S, D) est *complète* si et seulement si une chaine de valeurs appartenant à $\mathcal{H}$ relie de façon cohérente (R, W) à (D, S), *i.e.*

$$\begin{cases} X = R \oplus F_2(W) \in F_3 \\ Y = W \oplus F_3(X) \in F_4 \\ Z = X \oplus F_4(Y) \in F_5 \\ A = Y \oplus F_5(Z) \in F_6 \\ B = Z \oplus F_6(A) \in F_7 \\ C = A \oplus F_7(B) \in F_8 \\ B \oplus F_8(C) = D \\ C \oplus F_9(D) = S \ , \end{cases}$$

et incomplète sinon.

On dira que $\mathcal{H}$ est *libre de toute chaine externe incomplète* si et seulement toute chaine externe (W, R, S, D) est complète.

Pour tout élément $W \in F_2$, on notera $\texttt{Chain}_2(W)$ l'ensemble des triplets $(R, S, D) \in F_1 \times F_{10} \times F_9$ tels que (W, R, S, D) forme une chaine externe incomplète. De même, pour tout élément $D \in F_9$, on notera $\texttt{Chain}_9(D)$ l'ensemble des triplets $(S, R, W) \in F_{10} \times F_1 \times F_2$ tels que (W, R, S, D) forme une chaine externe incomplète. ⧫

Remarque 2.1. Pour tout $W \in F_2$ (resp. pour tout $D \in F_9$), l'ensemble $\texttt{Chain}_2(W)$ (resp. $\texttt{Chain}_9(D)$) peut être calculé avec au plus $|F_1|$ (resp. $|F_{10}|$) requêtes à $\boldsymbol{P}/\boldsymbol{P}^{-1}$. Nous supposerons par la suite que le simulateur possède une routine interne (que nous ne décrirons pas pour ne pas alourdir la description de $\mathcal{S}$) afin de maintenir à jour la liste des chaines complètes. ∗

Définition 2.2 (Centre)
Étant donnés un historique $\mathcal{H}$ et une permutation $\boldsymbol{P}$, un couple $(Z, A) \in F_5 \times F_6$ est appelé *centre*. On dira qu'un centre est *complet* si et seulement si la chaine correspondant à (Z, A) est complète et cohérente avec $\boldsymbol{P}$, *i.e.*

$$\begin{cases} Y = A \oplus F_5(Z) \in F_4 \\ X = Z \oplus F_4(Y) \in F_3 \\ W = Y \oplus F_3(X) \in F_2 \\ R = X \oplus F_2(W) \in F_1 \ , \end{cases} \qquad \begin{cases} B = Z \oplus F_6(A) \in F_7 \\ C = A \oplus F_7(B) \in F_8 \\ D = B \oplus F_8(C) \in F_9 \\ S = C \oplus F_9(D) \in F_{10} \ , \end{cases}$$

et $\boldsymbol{P}(L \| R) = S \| T$, avec $L = W \oplus F_1(R)$ et $T = D \oplus F_{10}(S)$, et incomplet sinon.

On dira que $\mathcal{H}$ est *libre de tout centre incomplet* si et seulement tout centre $(Z, A) \in F_5 \times F_6$ est complet.

Pour tout élément $Z \in F_5$, on notera $\texttt{Chain}_5(Z)$ l'ensemble des éléments $A \in F_6$ tels que (Z, A) forme un centre incomplet. De même, pour tout élément $A \in F_6$, on notera $\texttt{Chain}_6(A)$ l'ensemble des éléments $Z \in F_5$ tels que (Z, A) forme un centre incomplet. ⧫

Convention. Pour des raisons qui deviendront claires après la description du simulateur, il nous faut éviter de considérer deux fois la même chaine externe incomplète dans deux ensembles $\texttt{Chain}_2(W)$ et $\texttt{Chain}_9(D)$, ou deux fois le même centre incomplet dans deux ensembles $\texttt{Chain}_5(Z)$ et $\texttt{Chain}_6(A)$.

Par convention, pour une chaine externe incomplète (W, R, S, D), on considérera que $(R, S, D) \in \texttt{Chain}_2(W)$ mais $(S, R, W) \notin \texttt{Chain}_9(D)$ si W a été rajouté à l'historique après D, et réciproquement $(S, R, W) \in \texttt{Chain}_9(D)$ mais $(R, S, D) \notin \texttt{Chain}_2(W)$ si D a été rajouté à l'historique après W.

De même, pour un centre incomplet (Z, A), on considérera que $A \in \texttt{Chain}_5(Z)$ mais $Z \notin \texttt{Chain}_6(A)$ si Z a été rajouté à l'historique après A, et réciproquement $Z \in \texttt{Chain}_6(A)$ mais $A \notin \texttt{Chain}_5(Z)$ si A a été rajouté à l'historique après Z.

2.3.2 Description du simulateur

Nous ne donnons ici qu'une description générale du simulateur. Un pseudo-code détaillé se trouve dans l'appendice D.

Comme expliqué précédemment, le simulateur maintient un historique des valeurs des fonctions F_i déjà définies, soit en raison d'une F-requête du distingueur, soit de façon interne. Lorsque le simulateur reçoit une F-requête du distingueur pour $F_i(U)$, si U est déjà dans l'historique de F_i, le simulateur se contente de retourner la valeur $F_i(U)$ correspondante. Sinon, il définit $F_i(U)$ aléatoirement, puis il complète l'historique jusqu'à ce que celui-ci soit libre de toute chaine externe incomplète et de tout centre incomplet. Pour cela, deux procédures sont définies, `CompleteExtCh` et `CompleteCenter`.

`CompleteExtCh` prend en entrée deux ensembles Ω_2 et Ω_9. Ω_2 est l'ensemble des W venant d'être ajoutés à l'historique de F_2 si bien que $\texttt{Chain}_2(W)$ est potentiellement non-vide, et Ω_9 est l'ensemble des D venant d'être ajoutés à l'historique de F_9 si bien que $\texttt{Chain}_9(D)$ est potentiellement non-vide. Ainsi, si l'on suppose l'historique libre de toute chaine externe incomplète et de tout centre incomplet, et si le distingueur effectue une F-requête pour $F_2(W)$, cela va potentiellement créer des chaines externes incomplètes que le simulateur va compléter en appelant $\texttt{CompleteExtCh}(\Omega_2, \Omega_9)$, avec $\Omega_2 = \{W\}$ et $\Omega_9 = \emptyset$.

Pour chacun des éléments $W \in \Omega_2$, et chacun des triplets $(R, S, D) \in \texttt{Chain}_2(W)$, le simulateur appelle la procédure $\texttt{CompleteChain}_2(W, R, S, D)$ qui complète la chaine externe (W, R, S, D) de telle sorte qu'elle soit complète au sens de la définition 2.1. Pour cela, le simulateur prolonge la chaine en calculant :

$$\begin{cases} C = S \oplus F_9(D) \\ B = D \oplus F_8(C) \\ A = C \oplus F_7(B) \\ Z = B \oplus F_6(A) \\ Y = A \oplus F_5(Z) \end{cases},$$

et en définissant aléatoirement les valeurs de $F_8(C)$, $F_7(B)$, $F_6(A)$ et $F_5(Z)$ si celles-ci ne sont pas déjà dans l'historique. Il calcule également $X = R \oplus F_2(W)$, et adapte les valeurs de $F_3(X)$ et $F_4(Y)$ afin de joindre les deux bouts de la chaine en définissant :

$$\begin{cases} F_3(X) = W \oplus Y \\ F_4(Y) = X \oplus Z \end{cases}.$$

Si X ou Y sont déjà dans l'historique de F_3 ou F_4, le simulateur abandonne. Nous parlerons d'abandon avec erreur UNABLE_TO_ADAPT. Un point crucial de la démonstration sera de montrer qu'il est toujours possible d'adapter les valeurs de $F_3(X)$ et $F_4(Y)$ pour que la chaine soit cohérente avec $\boldsymbol{P}$, sauf avec une probabilité négligeable.

De façon symétrique, pour chacun des éléments $D \in \Omega_9$, et chaque triplet $(S, R, W) \in \texttt{Chain}_9(D)$, le simulateur appelle la procédure $\texttt{CompleteChain}_9(D, S, R, W)$ qui complète la chaine externe (W, R, S, D) de façon similaire à $\texttt{CompleteChain}_2$, à la différence près que l'adaptation a lieu en $F_7(B)$ et $F_8(C)$.

La complétion des chaines externes rajoute des éléments aux historiques de F_5 et F_6, ce qui crée potentiellement des centres incomplets. Une fois l'exécution de `CompleteExtCh` terminée, le simulateur appelle donc $\texttt{CompleteCenter}(\Omega_5, \Omega_6)$, où Ω_5 et Ω_6 sont respectivement l'ensemble des valeurs Z rajoutées à l'historique de F_5 et l'ensemble des valeurs A rajoutées à l'historique de F_6 par les exécutions de $\texttt{CompleteChain}_2$ et $\texttt{CompleteChain}_9$. Le simulateur complète alors les centres (Z, A) correspondants à l'aide des procédures $\texttt{CompleteChain}_5(Z, A)$, appelée pour chaque $Z \in \Omega_5$ et chaque $A \in \texttt{Chain}_5(Z)$, ainsi que $\texttt{CompleteChain}_6(A, Z)$, appelée pour chaque $A \in \Omega_6$ et $Z \in \texttt{Chain}_6(A)$. Les procédures $\texttt{CompleteChain}_5$ et $\texttt{CompleteChain}_6$ sont très similaires à $\texttt{CompleteChain}_2$ et $\texttt{CompleteChain}_9$, à la différence près qu'un appel à $\boldsymbol{P}/\boldsymbol{P}^{-1}$ est nécessaire durant l'exécution pour compléter la chaine. $\texttt{CompleteChain}_5$ s'adapte en $F_3(X)$ et $F_4(Y)$, tandis que $\texttt{CompleteChain}_6$ s'adapte en $F_7(B)$ et $F_8(C)$. Nous renvoyons à l'appendice D pour les détails des différentes procédures.

À nouveau, les exécutions de $\texttt{CompleteChain}_5$ et $\texttt{CompleteChain}_6$ ont rajouté des éléments aux historiques de F_2 et F_9, créant potentiellement de nouvelles chaines externes incomplètes. Une fois l'exécution de `CompleteCenter` terminée, le simulateur appelle donc $\texttt{CompleteExtCh}(\Omega_2, \Omega_9)$, où Ω_2 et Ω_9 sont respectivement l'ensemble des valeurs W rajoutées à l'historique de F_2 et l'ensemble des valeurs D rajoutées à l'historique de F_9 par les exécutions de $\texttt{CompleteChain}_5$ et $\texttt{CompleteChain}_6$. Les appels récursifs à `CompleteCenter` et `CompleteExtCh` se poursuivent jusqu'à ce que l'historique soit libre de toute chaine externe incomplète et de tout centre incomplet.

Afin que le simulateur ne « s'emballe » pas lors des appels récursifs à `CompleteExtCh` et `CompleteCenter`, et n'effectue un nombre trop grand de requêtes à $\boldsymbol{P}$ (rappelons que la définition de l'indifférentiabilité exige que le simulateur fasse un nombre de requêtes $\sigma \in \texttt{poly}(k)$ lorsque le distingueur fait un nombre total de requêtes $q \in \texttt{poly}(k)$), le simulateur abandonne lorsque le nombre total d'appels aux procédures $\texttt{CompleteChain}_2$ ou $\texttt{CompleteChain}_9$ (celles qui traitent les chaines externes) dépasse une borne égale à q, le nombre maximal de requêtes du distingueur. Nous parlerons d'abandon avec erreur OUT_OF_BOUND. Comme nous le verrons, la probabilité qu'une chaine externe (W, R, S, D) incomplète soit formée sans que le distingueur ait fait la P-requête correspondante étant négligeable, la probabilité que le simulateur abandonne avec erreur OUT_OF_BOUND l'est également.

Si une F-requête du distingueur entraine l'abandon du simulateur (que ce soit avec erreur OUT_OF_BOUND ou UNABLE_TO_ADAPT), le simulateur retourne le symbole spécial $\perp$ en réponse à la F-requête du distingueur.

Nous allons maintenant montrer successivement à travers une série de lemmes que :

1. la complexité du simulateur est polynomiale en q ;
2. le simulateur abandonne avec erreur OUT_OF_BOUND avec une probabilité négligeable ;
3. le simulateur abandonne avec erreur UNABLE_TO_ADAPT avec une probabilité négligeable ;
4. enfin, le système $(\boldsymbol{P}, \mathcal{S}^{\boldsymbol{P}})$ est indistinguable du système $(\Psi_{10}^{\boldsymbol{F}}, \boldsymbol{F})$.

La combinaison de ces divers lemmes nous permettra d'établir le théorème suivant, dont le théorème 2.1 est une conséquence directe :

Théorème 2.4. *Pour tout $q \in \mathbb{N}$, la construction de Luby-Rackoff à 10 tours $\Psi_{10}^{\boldsymbol{F}}$ est fortement (q, σ, ϵ)-indifférentiable d'une permutation aléatoire inversible $\boldsymbol{P}$, avec :*

$$\sigma(q) = 32q^4 + \mathcal{O}(q^3) \qquad et \qquad \epsilon(q) = \frac{2^{21} \cdot q^4}{2^n} + \mathcal{O}\left(\frac{q^3}{2^n}\right) \ . \qquad \diamondsuit$$

2.3.3 Analyse de la complexité du simulateur

Nous allons montrer tout d'abord que la complexité du simulateur est polynomiale en q. Plus précisément, nous allons voir que la taille des historiques est en $\mathcal{O}(q)$ pour les deux fonctions centrales F_5 et F_6, en $\mathcal{O}(q^2)$ pour les autres, et que le nombre de requêtes à $\boldsymbol{P}/\boldsymbol{P}^{-1}$ effectuées par le simulateur est en $\mathcal{O}(q^4)$.

Lemme 2.5. *Au cours de la simulation, la taille des historiques des fonctions F_i vérifie toujours :*

$$\begin{cases} |F_i| \leq 2q & \text{pour } i \in \{5, 6\} \\ |F_i| \leq q + 4q^2 & \text{pour } i \in \{1, 2, 9, 10\} \\ |F_i| \leq 2q + 4q^2 & \text{pour } i \in \{3, 4, 7, 8\} \ . \end{cases} \qquad \triangledown$$

Démonstration. L'historique des fonctions F_5 et F_6 ne peut augmenter que lors d'une F-requête du distingueur, ou lors d'un appel à $\texttt{CompleteChain}_2$ ou $\texttt{CompleteChain}_9$, chaque appel ajoutant au plus un seul élément à chacun des historiques. Le nombre de requêtes du distingueur est inférieur à q, de même que le nombre total d'appels à $\texttt{CompleteChain}_2$ ou $\texttt{CompleteChain}_9$ par construction du simulateur (sinon il abandonne avec erreur OUT_OF_BOUND), d'où la première inégalité.

L'historique des fonctions F_i, $i \in \{1, 2, 9, 10\}$, ne peut augmenter que lors d'une F-requête du simulateur, ou lors d'un appel à $\texttt{CompleteChain}_5$ ou $\texttt{CompleteChain}_6$. Le nombre total d'appels à $\texttt{CompleteChain}_5$ et $\texttt{CompleteChain}_6$ est inférieur au nombre de centres (Z, A). Le nombre total de centres (Z, A) étant lui-même inférieur à $|F_5| \cdot |F_6| \leq (2q)^2 = 4q^2$, l'historique de ces fonctions est inférieur à $q + 4q^2$.

Enfin, l'historique des fonctions F_i, $i \in \{3, 4, 7, 8\}$, ne peut augmenter que lors d'une F-requête du simulateur, ou lors d'un appel à $\texttt{CompleteChain}_j$, $j \in \{2, 5, 6, 9\}$. L'historique de ces fonctions est donc inférieur à $q + q + 4q^2 = 2q + 4q^2$. ∎

Par la suite nous noterons

$$\begin{cases} |F_i|_{\max} = 2q & \text{pour } i \in \{5, 6\} \\ |F_i|_{\max} = q + 4q^2 & \text{pour } i \in \{1, 2, 9, 10\} \\ |F_i|_{\max} = 2q + 4q^2 & \text{pour } i \in \{3, 4, 7, 8\} \ . \end{cases}$$

Lemme 2.6. *Le nombre $\sigma(q)$ de requêtes à $\boldsymbol{P}/\boldsymbol{P}^{-1}$ effectuées par le simulateur lorsque le distingueur effectue au plus q requêtes vérifie $\sigma(q) = 32q^4 + \mathcal{O}(q^3)$.* $\triangledown$

Démonstration. Le simulateur effectue un appel à $\boldsymbol{P}/\boldsymbol{P}^{-1}$ lors du calcul des ensembles $\texttt{Chain}_2(W)$ et $\texttt{Chain}_9(D)$, et lorsqu'il complète un centre. Le nombre d'appels du premier type est inférieur à $|F_1|_{\max} \cdot |F_2|_{\max} + |F_9|_{\max} \cdot |F_{10}|_{\max} = 2 \cdot (q + 4q^2)^2$, le nombre d'appels du second type est inférieur à $|F_5|_{\max} \cdot |F_6|_{\max} = 4q^2$. Par conséquent $\sigma(q) \leq 2 \cdot (q + 4q^2)^2 + 4q^2$, d'où le résultat. ∎

2.3.4 Probabilité d'abandon avec erreur OUT_OF_BOUND

Soit $\texttt{oob}$ l'événement que le simulateur abandonne avec erreur OUT_OF_BOUND. Rappelons que cela se produit lorsque le nombre d'appels aux procédures $\texttt{CompleteChain}_2$ ou $\texttt{CompleteChain}_9$ est supérieur à q, le nombre total de requêtes du distingueur. Nous allons voir que la probabilité pour que cela arrive est négligeable. L'argument principal est qu'une chaine externe (W, R, S, D) ne se forme pas par hasard : une chaine externe (W, R, S, D) ne se forme avec une probabilité non négligeable que si le distingueur a fait la P-requête correspondante, c'est-à-dire la requête $\boldsymbol{P}((W \oplus F_1(R))\|R)$ ou $\boldsymbol{P}^{-1}(S\|(D \oplus F_{10}(S)))$.

Lemme 2.7. *La probabilité que le simulateur abandonne avec erreur* OUT_OF_BOUND *vérifie :*

$$\Pr\left[\texttt{oob}\right] \leq \frac{16q^4}{2^n} + \mathcal{O}\left(\frac{q^3}{2^n}\right) \quad . \qquad \triangledown$$

Démonstration. Soit $\texttt{Bad}$ l'événement qu'un appel à $\texttt{CompleteChain}_2(W, R, S, D)$ ou à $\texttt{CompleteChain}_9(D, S, R, W)$ soit effectué pour des valeurs (W, R, S, D) telles que le distingueur n'a pas fait la requête $\boldsymbol{P}((W \oplus F_1(R))\|R)$ ou $\boldsymbol{P}^{-1}(S\|(D \oplus F_{10}(S)))$. Sachant $\overline{\texttt{Bad}}$, le nombre total d'appels à $\texttt{CompleteChain}_2$ et $\texttt{CompleteChain}_9$ est inférieur au nombre de P-requêtes du distingueur, donc inférieur à q, si bien que le simulateur n'abandonne pas avec erreur OUT_OF_BOUND.

Nous n'avons donc plus qu'à majorer la probabilité de l'événement $\texttt{Bad}$. Par la suite, nous noterons $(U \rightarrow F_i)$ l'événement que U soit ajouté à l'historique de F_i au cours de la simulation. Fixons l'aléa du simulateur, et considérons l'espace de probabilité résultant (dépendant uniquement de la permutation $\boldsymbol{P}$ puisque le distingueur est supposé déterministe). Soient W_0, R_0, S_0 et D_0 quatre valeurs fixées de $\{0,1\}^n$. L'aléa du simulateur étant fixé, et les valeurs de F_1 et F_{10} ne dépendant que de l'aléa du simulateur et non de la permutation $\boldsymbol{P}$, en particulier $F_1(R_0)$ et $F_{10}(S_0)$ sont fixés. Soient $L_0 = W_0 \oplus F_1(R_0)$ et $T_0 = D_0 \oplus F_{10}(S_0)$. Enfin, soit $\texttt{Bad}_{(W_0,R_0,S_0,D_0)}$ l'événement qu'un appel à $\texttt{CompleteChain}_2(W_0, R_0, S_0, D_0)$ ou à $\texttt{CompleteChain}_9(D_0, S_0, R_0, W_0)$ soit effectué sans que le distingueur ait fait la requête $\boldsymbol{P}(L_0\|R_0)$ ou $\boldsymbol{P}^{-1}(S_0\|T_0)$. Pour que $\texttt{Bad}_{(W_0,R_0,S_0,D_0)}$ se produise, il est clairement nécessaire que $\boldsymbol{P}(L_0\|R_0) = S_0\|T_0$, par conséquent :

$$\Pr\left[\texttt{Bad}_{(W_0,R_0,S_0,D_0)}\right] = \frac{1}{2^{2n}} \cdot \Pr\left[\texttt{Bad}_{(W_0,R_0,S_0,D_0)} \mid \boldsymbol{P}(L_0\|R_0) = S_0\|T_0\right] \quad .$$

Tant que le simulateur n'a pas fait d'appel à $\texttt{CompleteChain}_2(W_0, R_0, S_0, D_0)$ ou à $\texttt{CompleteChain}_9(D_0, S_0, R_0, W_0)$, son historique et ses réponses sont indépendantes de l'événement $\boldsymbol{P}(L_0\|R_0) = S_0\|T_0$. Par ailleurs, le distingueur étant déterministe, sa i-ième requête est déterminée par ses $i-1$ requêtes précédentes et les réponses (de $\boldsymbol{P}$ ou du simulateur) correspondantes. Par conséquent, si le distingueur n'a pas fait de P-requête correspondant aux valeurs $L_0\|R_0$ ou $S_0\|T_0$, et si le simulateur n'a pas encore fait d'appel à $\texttt{CompleteChain}_2(W_0, R_0, S_0, D_0)$ ou à $\texttt{CompleteChain}_9(D_0, S_0, R_0, W_0)$, les événements $(W_0 \rightarrow F_2)$, $(R_0 \rightarrow F_1)$, $(S_0 \rightarrow F_{10})$ et $(D_0 \rightarrow F_9)$ sont indépendants de l'événement $\boldsymbol{P}(L_0\|R_0) = S_0\|T_0$, si bien que :

$$\Pr\left[\texttt{Bad}_{(W_0,R_0,S_0,D_0)} \mid \boldsymbol{P}(L_0\|R_0) = S_0\|T_0\right] \leq \Pr[(W_0 \rightarrow F_2) \wedge (R_0 \rightarrow F_1) \wedge (S_0 \rightarrow F_{10}) \wedge (D_0 \rightarrow F_9)] \quad .$$

En sommant sur $(W_0, R_0, S_0, D_0) \in (\{0,1\}^n)^4$, on obtient (en tenant compte du lemme 2.5) :

$$\Pr[\texttt{Bad}] \leq \frac{|F_1|_{\max} \cdot |F_2|_{\max} \cdot |F_9|_{\max} \cdot |F_{10}|_{\max}}{2^{2n}} = \frac{(q+4q^2)^4}{2^{2n}} \leq \frac{(q+4q^2)^2}{2^n} \quad .$$

La majoration étant valable quel que soit l'aléa du simulateur, elle reste vraie en moyennant sur ce dernier. Le lemme s'ensuit. ∎

2.3.5 Probabilité d'abandon avec erreur UNABLE_TO_ADAPT

Nous allons maintenant analyser la probabilité pour que le simulateur abandonne en raison d'une impossibilité de s'adapter lors d'un appel à $\texttt{CompleteChain}_j$, $j \in \{2,5,6,9\}$. Cela se produit lorsque l'une des deux valeurs où le simulateur s'adapte (X et Y pour $\texttt{CompleteChain}_2$ et $\texttt{CompleteChain}_5$, et B et C pour $\texttt{CompleteChain}_6$ et $\texttt{CompleteChain}_9$) est déjà dans l'historique de la fonction F_i correspondante. Soit `uta` l'événement que le simulateur abandonne avec erreur UNABLE_TO_ADAPT. Nous allons montrer que la probabilité de l'événement `uta` est négligeable.

Le cœur de la preuve repose sur le fait que tant que le simulateur n'a pas abandonné, l'historique juste avant un appel à `CompleteExtCh` est libre de tout centre incomplet. Par conséquent, lorsqu'un appel à $\texttt{CompleteChain}_2(W,R,S,D)$ est effectué, le centre (Z,A) correspondant ne peut pas déjà être dans l'historique (sinon la chaine serait déjà complète), si bien que $F_5(Z)$ est défini aléatoirement. Par conséquent $Y = A \oplus F_5(Z)$ n'est dans l'historique de F_4 qu'avec une probabilité négligeable et le simulateur n'abandonne pas lors de l'adaptation en $F_4(Y)$. Il n'abandonne pas non plus pour l'adaptation en $F_3(X)$ car $X = R \oplus F_2(W)$ et $F_2(W)$ a été défini aléatoirement. Le raisonnement est symétrique pour un appel à $\texttt{CompleteChain}_9(D,S,R,W)$.

De même, tant que le simulateur n'a pas abandonné, l'historique juste avant un appel à `CompleteCenter` est libre de toute chaine externe incomplète. Par conséquent, lorsqu'un appel à $\texttt{CompleteChain}_5(Z,A)$ est effectué, la chaine externe (W,R,S,D) correspondante ne peut pas déjà être dans l'historique (sinon la chaine serait déjà complète), si bien que $F_2(W)$ est défini aléatoirement. Par conséquent $X = R \oplus F_2(W)$ n'est dans l'historique de F_3 qu'avec une probabilité négligeable et le simulateur n'abandonne pas lors de l'adaptation en $F_3(X)$. Il n'abandonne pas non plus pour l'adaptation en $F_4(Y)$ car $Y = A \oplus F_5(Z)$ et $F_5(Z)$ a été défini aléatoirement. Le raisonnement est symétrique pour un appel à $\texttt{CompleteChain}_6(A,Z)$.

En fait, il y a une subtilité supplémentaire due au fait qu'une chaine externe puisse être formée « par malchance » lors de l'ajout d'un élément à l'historique de F_1 ou F_{10} et soit non détectée, donc non complétée par le simulateur. Nous noterons $\texttt{Bad}'$ cet événement, c'est-à-dire l'événement qu'une chaine externe soit formée lors de la définition de $F_1(R)$ pour un R quelconque ou de $F_{10}(S)$ pour un S quelconque, soit lors d'une F-requête du distingueur, soit lors d'un appel à $\texttt{CompleteChain}_5$ ou $\texttt{CompleteChain}_6$. Alors :

Lemme 2.8. *La probabilité de l'événement* $\texttt{Bad}'$ *vérifie :*

$$\Pr[\texttt{Bad}'] \leq \frac{16q^4}{2^n} + \mathcal{O}\left(\frac{q^3}{2^n}\right) \quad .$$

▽

Démonstration. Considérons l'ajout d'un unique élément R à l'historique de F_1. L'événement $\texttt{Bad}'$ se produit si et seulement si $F_1(R)$ est défini tel que $F_1(R) = L \oplus W$, avec $W \in F_2$ et L tel qu'il existe $(S, D) \in F_{10} \times F_9$ avec $\boldsymbol{P}^{-1}(S\|(D \oplus F_{10}(S))) = L\|R$. Soit N_R le nombre de telles paires (S, D). Alors la probabilité que l'ajout de R à l'historique de F_1 provoque $\texttt{Bad}'$ est inférieure à $\frac{N_R \cdot |F_2|_{\max}}{2^n} = \frac{N_R(q+4q^2)}{2^n}$.

De même, considérons l'ajout d'un unique élément S à l'historique de F_{10}. L'événement $\texttt{Bad}'$ se produit si et seulement si $F_{10}(S)$ est défini tel que $F_{10}(S) = T \oplus D$, avec $D \in F_9$ et T tel qu'il existe $(R, W) \in F_1 \times F_2$ avec $\boldsymbol{P}((W \oplus F_1(R))\|R) = S\|T$. Soit N_S le nombre de telles paires (R, W). Alors la probabilité que l'ajout de S à l'historique de F_{10} provoque $\texttt{Bad}'$ est inférieure à $\frac{N_S \cdot |F_9|_{\max}}{2^n} = \frac{N_S(q+4q^2)}{2^n}$.

Par conséquent, la probabilité de $\texttt{Bad}'$ est inférieure à la somme sur les éléments R et S ajoutés respectivement aux historiques de F_1 et F_{10} de $\frac{N_R(q+4q^2)}{2^n} + \frac{N_S(q+4q^2)}{2^n}$. Une majoration brutale de la somme des N_R par $|F_9|_{\max} \cdot |F_{10}|_{\max}$ et de la somme des N_S par $|F_1|_{\max} \cdot |F_2|_{\max}$ donne une probabilité en $\mathcal{O}\left(\frac{q^6}{2^n}\right)$. On peut cependant améliorer cette borne de la façon suivante. D'une façon exactement identique à la démonstration du lemme 2.7, on peut montrer que la probabilité que l'événement $\texttt{Bad}'$ se produise pour des valeurs (W, R, S, D) telles que le distingueur n'a pas fait la requête $\boldsymbol{P}((W \oplus F_1(R))\|R)$ ou $\boldsymbol{P}^{-1}(S\|(D \oplus F_{10}(S)))$ est inférieure à $\frac{16q^4}{2^n} + \mathcal{O}\left(\frac{q^3}{2^n}\right)$. On peut alors se restreindre aux valeurs correspondant aux P-requêtes du distingueur, si bien que $\sum N_R + \sum N_S \leq q$, d'où le résultat. ∎

Il nous reste donc à majorer la probabilité de $\texttt{uta}$ sachant $\overline{\texttt{Bad}'}$. Afin de formaliser la preuve, nous avons besoin d'introduire quelques définitions caractérisant l'état de l'historique au moment d'un appel à `CompleteExtCh` ou `CompleteCenter`.

Définition 2.3 (Historique pseudo-libre)
Étant donnés un historique $\mathcal{H}$ et des ensembles $\Omega_2 \subset F_2$ et $\Omega_9 \subset F_9$, on dira que l'historique est *pseudo-libre* relativement à Ω_2 et Ω_9 s'il est libre de tout centre incomplet, et libre de toute chaine externe incomplète, à l'exception des chaines externes faisant intervenir $W \in \Omega_2$ ou $D \in \Omega_9$.
Étant donnés un historique $\mathcal{H}$ et des ensembles $\Omega_5 \subset F_5$ et $\Omega_6 \subset F_6$, on dira que l'historique est *pseudo-libre* relativement à Ω_5 et Ω_6 s'il est libre de toute chaine externe incomplète, et libre de tout centre incomplet à l'exception des centres faisant intervenir $Z \in \Omega_5$ ou $A \in \Omega_6$. ♦

Lemme 2.9. *Supposons* $\overline{\texttt{Bad}'}$*. Alors :*
- *lors d'un appel à* $\texttt{CompleteExtCh}(\Omega_2, \Omega_9)$*, si l'historique est pseudo-libre relativement à* Ω_2 *et* Ω_9*, alors, si le simulateur n'abandonne pas, lors de l'appel récursif à* $\texttt{CompleteCenter}(\Omega_5, \Omega_6)$*, l'historique est pseudo-libre relativement à* Ω_5 *et* Ω_6 *;*
- *lors d'un appel à* $\texttt{CompleteCenter}(\Omega_5, \Omega_6)$*, si l'historique est pseudo-libre relativement à* Ω_5 *et* Ω_6*, alors, si le simulateur n'abandonne pas, lors de l'appel récursif à* $\texttt{CompleteExtCh}(\Omega_2, \Omega_9)$*, l'historique est pseudo-libre relativement à* Ω_2 *et* Ω_9 *;*
- *dans les 2 cas, s'il n'y a pas d'appel récursif à* `CompleteExtCh` *ou* `CompleteCenter`*, alors l'historique est libre de toute chaine externe incomplète et de tout centre incomplet.* ▽

Démonstration. Dans le cas où le simulateur n'a pas abandonné à l'issue de l'exécution de `CompleteExtCh`, l'historique est libre de toute chaine externe incomplète car toutes les chaines externes incomplètes au moment de l'appel à `CompleteExtCh` ont été correctement complétées, et les seuls centres incomplets font nécessairement intervenir $Z \in \Omega_5$

ou $A \in \Omega_6$ car l'historique était libre de tout centre incomplet au moment de l'appel à `CompleteExtCh`. S'il n'y pas d'appel récursif à `CompleteCenter`, l'historique est nécessairement également libre de tout centre incomplet. L'argument est identique pour le cas d'un appel à `CompleteCenter`, avec la subtilité supplémentaire déjà mentionnée qu'il faut faire intervenir l'hypothèse $\overline{\texttt{Bad}'}$ afin de s'assurer qu'aucune chaine externe non détectée ne se forme lors de l'ajout d'une valeur R à l'historique de F_1 ou d'une valeur S à l'historique de F_{10}. ∎

On en déduit aisément le corollaire suivant :

Corollaire 2.10. *Sachant $\overline{\texttt{Bad}'}$ et tant que le simulateur n'a pas abandonné, alors :*
- *lors d'un appel à* `CompleteExtCh`(Ω_2, Ω_9)*, l'historique est pseudo-libre relativement à Ω_2 et Ω_9 ;*
- *lors d'un appel à* `CompleteCenter`(Ω_5, Ω_5)*, l'historique est pseudo-libre relativement à Ω_5 et Ω_6.* ▽

Nous sommes maintenant prêts à majorer la probabilité de l'événement `uta` sachant $\overline{\texttt{Bad}'}$:

Lemme 2.11.

$$\Pr\left[\texttt{uta} \,|\, \overline{\texttt{Bad}'}\right] \leq \frac{64q^4}{2^n} + \mathcal{O}\left(\frac{q^3}{2^n}\right) \ . \qquad \text{▽}$$

Démonstration. Supposons $\overline{\texttt{Bad}'}$, et considérons un appel à `CompleteExtCh`(Ω_2, Ω_9).

Lors d'un appel à $\texttt{CompleteChain}_2(W, R, S, D)$, $W \in \Omega_2$, le simulateur n'abandonne pas lors de l'adaptation en $F_3(X)$, sauf avec une probabilité inférieure à $\frac{|F_3|_{\max}}{2^n}$. En effet, la valeur de $X = R \oplus F_2(W)$ est uniformément aléatoire par rapport à l'historique de F_3 car $F_2(W)$ a été tiré aléatoirement.

Montrons que le simulateur n'abandonne pas non plus lors de l'adaptation en $F_4(Y)$, sauf avec une probabilité négligeable. Rappelons que Y est déterminé par $Y = A \oplus F_5(Z)$. D'après le corollaire 2.10, le simulateur n'ayant pas encore abandonné, l'historique est pseudo-libre par rapport à Ω_2 et Ω_9, donc libre de tout centre incomplet. Par conséquent, le centre (Z, A) correspondant à la chaine externe (W, R, S, D) ne peut pas déjà être dans l'historique, sinon la chaine serait déjà complète et $\texttt{CompleteChain}_2$ ne serait pas appelé. Distinguons alors deux cas. Si A est déjà dans l'historique de F_6, Z n'est pas dans l'historique de F_5, si bien que $F_5(Z)$ est défini aléatoirement et $Y = A \oplus F_5(Z)$ est dans l'historique de F_4 avec une probabilité inférieure à $\frac{|F_4|_{\max}}{2^n}$. Si A n'est pas dans l'historique de F_6, alors $F_6(A)$ est défini aléatoirement, donc $Z = B \oplus F_6(A)$ est dans l'historique de F_5 avec une probabilité inférieure à $\frac{|F_5|_{\max}}{2^n}$. Dans le cas contraire, on est ramené au cas précédent. La probabilité totale que Y soit dans l'historique de F_4 est donc inférieure à :

$$\frac{|F_4|_{\max}}{2^n} + \frac{|F_5|_{\max}}{2^n} + \left(1 - \frac{|F_5|_{\max}}{2^n}\right)\frac{|F_4|_{\max}}{2^n} \leq \frac{2|F_4|_{\max} + |F_5|_{\max}}{2^n} \ .$$

La probabilité que le simulateur abandonne lors d'un unique appel à $\texttt{CompleteChain}_2$ est donc inférieure à :

$$p_{2,9} = \frac{|F_3|_{\max} + 2|F_4|_{\max} + |F_5|_{\max}}{2^n} \ ,$$

soit, d'après le lemme 2.5 :

$$p_{2,9} = \frac{8q + 12q^2}{2^n} \ .$$

Le raisonnement est symétrique pour un appel à $\texttt{CompleteChain}_9(D, S, R, W)$, $D \in \Omega_9$, si bien que nous l'omettons.

Le raisonnement est similaire pour un appel à $\texttt{CompleteCenter}(\Omega_5, \Omega_6)$. On obtient que lors d'un appel à $\texttt{CompleteChain}_5(Z, A)$, le simulateur abandonne avec une probabilité inférieure à :

$$p_{5,6} = \frac{|F_4|_{\max} + 2|F_3|_{\max} + |F_2|_{\max}}{2^n} = \frac{7q + 16q^2}{2^n} \quad .$$

La même borne est également valable pour un appel à $\texttt{CompleteChain}_6(A, Z)$.

En tenant compte du fait que le nombre total d'appels à $\texttt{CompleteChain}_{2,9}$ est inférieur à q, et que le nombre total d'appels à $\texttt{CompleteChain}_{5,6}$ est inférieur à $4q^2$, la probabilité que le simulateur abandonne avec erreur UNABLE_TO_ADAPT sachant $\overline{\texttt{Bad}'}$ vérifie :

$$\Pr\left[\texttt{uta} \,|\, \overline{\texttt{Bad}'}\right] \leq q \cdot p_{2,9} + 4q^2 \cdot p_{5,6} = \frac{64q^4}{2^n} + \mathcal{O}\left(\frac{q^3}{2^n}\right) \quad . \qquad \blacksquare$$

En combinant les lemmes 2.8 et 2.11, on obtient finalement le résultat suivant :

Lemme 2.12. *La probabilité* $\Pr[\texttt{uta}]$ *pour que le simulateur abandonne avec erreur* UNABLE_TO_ADAPT *vérifie :*

$$\Pr[\texttt{uta}] \leq \frac{80q^4}{2^n} + \mathcal{O}\left(\frac{q^3}{2^n}\right) \quad . \qquad \triangledown$$

Conclusion quant à la probabilité d'abandon totale. Soit $\texttt{Abort} = \texttt{oob} \vee \texttt{uta}$ l'événement que le simulateur abandonne (quelle que soit l'erreur). Alors la combinaison des lemmes 2.7 et 2.12 nous donne :

Lemme 2.13. *La probabilité que le simulateur abandonne vérifie :*

$$\Pr[\texttt{Abort}] \leq \frac{96q^4}{2^n} + \mathcal{O}\left(\frac{q^3}{2^n}\right) \quad . \qquad \triangledown$$

2.3.6 Preuve d'indifférentiabilité

Il reste à montrer que les deux systèmes $(\boldsymbol{P}, \mathcal{S}^{\boldsymbol{P}})$ et $(\Psi_{10}^{\boldsymbol{F}}, \boldsymbol{F})$ sont indistinguables. Pour cela, nous allons utiliser un système intermédiaire (cf. figure 2.3). Il s'agit du système où la permutation aléatoire $\boldsymbol{P}$ est remplacée par une construction de Luby-Rackoff Ψ_{10} calculant ses réponses en faisant les requêtes nécessaires au simulateur, ce dernier ayant toujours accès à la permutation aléatoire $\boldsymbol{P}$ (qui est désormais inaccessible directement au distingueur). Nous supposerons que si le simulateur abandonne sur une requête de la construction Ψ_{10}, alors la construction renvoie le symbole spécial $\perp$ en réponse à la P-requête du distingueur.

Nous noterons chacun des trois systèmes considérés dans la preuve respectivement :

$$\begin{cases} \Sigma_1 = (\boldsymbol{P}, \mathcal{S}^{\boldsymbol{P}}) \\ \Sigma_2 = (\Psi_{10}^{\mathcal{S}^{\boldsymbol{P}}}, \mathcal{S}^{\boldsymbol{P}}) \\ \Sigma_3 = (\Psi_{10}^{\boldsymbol{F}}, \boldsymbol{F}) \quad , \end{cases}$$

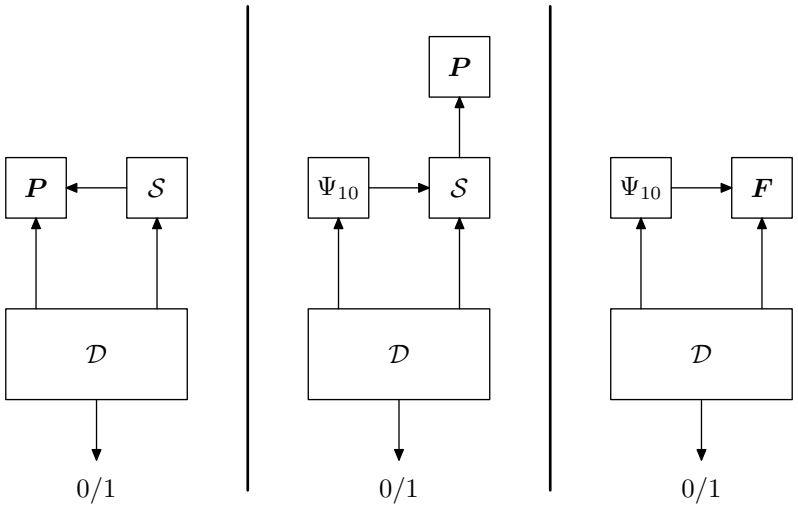

FIGURE 2.3 – Illustration de la preuve de l'indifférentiabilité de la construction de Luby-Rackoff à 10 tours.

et nous noterons, pour $i = 1$ à 3, $P_i = \Pr\left[\mathcal{D}^{\Sigma_i}(1^k) = 1\right]$.

Remarquons que cette décomposition correspond à l'idée intuitive des deux propriétés que doit posséder un « bon » simulateur : les réponses simulées doivent être cohérentes avec la construction Ψ_{10} (passage de Σ_1 à Σ_2), et les réponses doivent être indistinguables des réponses d'un oracle aléatoire $\boldsymbol{F}$ (passage de Σ_2 à Σ_3).

Nous cherchons donc à majorer la quantité :

$$\left|\Pr\left[\mathcal{D}^{\boldsymbol{P},\mathcal{S}^{\boldsymbol{P}}}(1^k) = 1\right] - \Pr\left[\mathcal{D}^{\Psi_{10}^{\boldsymbol{F}},\boldsymbol{F}}(1^k) = 1\right]\right| = |P_1 - P_3| \quad .$$

Commençons par montrer que $|P_1 - P_2|$ est négligeable :

Lemme 2.14.

$$|P_1 - P_2| \leq \frac{2^{20} \cdot q^4}{2^n} + \mathcal{O}\left(\frac{q^3}{2^n}\right) \quad . \qquad \triangledown$$

DÉMONSTRATION. Tout d'abord, remarquons que tant que le simulateur n'abandonne pas dans le système Σ_2, les réponses aux P-requêtes du distingueur sont identiques à celles qu'il obtient du système Σ_1 par construction du simulateur.

Il faut également montrer que les réponses aux F-requêtes du distingueur sont statistiquement proches dans les deux systèmes. Notons que cette distance statistique n'est pas nécessairement nulle car l'exécution de la simulation dans le système Σ_2 est modifiée par les requêtes de la construction Ψ_{10}. En supposant que le simulateur n'abandonne ni dans le système Σ_1 ni dans Σ_2, les réponses aux F-requêtes pour F_1, F_2, F_5, F_6, F_9 et F_{10} sont uniformément aléatoires et indépendantes des réponses précédentes dans les deux systèmes, le simulateur n'adaptant jamais les valeurs de ces fonctions.

Par contre, en fonction de la séquence de requêtes reçues par le simulateur, les réponses aux F-requêtes pour F_3, F_4, F_7 et F_8 sont soit tirées aléatoirement, soit adaptées par une exécution de `CompleteChain`. Considérons l'exemple suivant : le distingueur effectue une P-requête pour $L\|R$ et obtient la réponse $S\|T$, puis effectue successivement les F-requêtes pour $F_{10}(S)$, $F_9(D)$, avec $D = T \oplus F_{10}(S)$, $F_1(R)$ et enfin

$F_2(W)$, avec $W = L \oplus F_1(R)$. Lorsque $\mathcal{D}$ interagit avec Σ_1, le simulateur déclenche $\texttt{CompleteChain}_2(W, R, S, D)$ et l'adaptation a lieu en $F_3(X)$ et $F_4(Y)$. Par contre, lorsqu'il interagit avec Σ_2, la construction Ψ_{10} fait au simulateur les requêtes nécessaires pour calculer la réponse à la P-requête pour $L\|R$. On peut vérifier que cela déclenche un appel à $\texttt{CompleteChain}_6$ et que l'adaptation a lieu en $F_7(B)$ et $F_8(C)$. Cependant, sachant que le simulateur n'abandonne pas, cela n'entraine pas de distance statistique entre les sorties de Σ_1 et Σ_2.

En effet, considérons que la permutation $\boldsymbol{P}$ et les fonctions F_1, F_2, F_5, F_6, F_9 et F_{10} sont fixées. Alors, tant que le simulateur n'abandonne pas, les relations suivantes sont vérifiées, que l'adaptation ait lieu en $F_3(X)$ et $F_4(Y)$, ou en $F_7(B)$ et $F_8(C)$:

$$\begin{cases} F_3(X) \oplus F_7(B) = L \oplus F_1(R) \oplus F_5(Z) \oplus F_9(D) \oplus S \\ F_4(Y) \oplus F_8(C) = R \oplus F_2(W) \oplus F_6(A) \oplus F_{10}(S) \oplus T \\ \boldsymbol{P}(L|R) = S\|T \ . \end{cases}$$

Par conséquent, à $\boldsymbol{P}$ et F_1, F_2, F_5, F_6, F_9 et F_{10} fixés, la distribution jointe des fonctions F_3, F_4, F_7 et F_8 est identique quelle que soit la séquence d'adaptation.

Ainsi, en notant $\texttt{Abort}_i$ l'événement que le simulateur abandonne dans le système Σ_i, on a :

$$|P_1 - P_2| \leq \Pr[\texttt{Abort}_1] + \Pr[\texttt{Abort}_2] \ ,$$

soit, d'après le lemme 2.13, et en remarquant que le nombre total de F-requêtes au simulateur dans le système Σ_2 est inférieur à $10q$:

$$|P_1 - P_2| \leq \frac{96 \cdot q^4}{2^n} + \frac{96 \cdot (10q)^4}{2^n} + \mathcal{O}\left(\frac{q^3}{2^n}\right) \leq \frac{2^{20} \cdot q^4}{2^n} + \mathcal{O}\left(\frac{q^3}{2^n}\right) \ ,$$

ce qui conclut la démonstration [1]. ∎

Il nous reste à majorer $|P_2 - P_3|$:

Lemme 2.15.

$$|P_2 - P_3| \leq \frac{2^{20} \cdot q^4}{2^n} + \mathcal{O}\left(\frac{q^3}{2^n}\right) \ . \qquad \triangledown$$

Démonstration. Considérons la combinaison du distingueur $\mathcal{D}$ et de la construction Ψ_{10} comme un unique distingueur $\mathcal{D}'$. Ce distingueur interagit avec $\mathcal{S}^{\boldsymbol{P}}$ dans Σ_2 et avec $\boldsymbol{F}$ dans Σ_3. Il nous suffit donc de majorer la distance statistique entre les sorties des deux systèmes $\mathcal{S}^{\boldsymbol{P}}$ et $\boldsymbol{F}$.

En supposant que le simulateur n'abandonne pas dans le système Σ_2, les réponses au F-requêtes pour F_1, F_2, F_5, F_6, F_9 et F_{10} sont uniformément aléatoires et indépendantes des réponses précédentes, donc identiquement distribuées dans Σ_2 et Σ_3. Les réponses aux F-requêtes pour F_3, F_4, F_7 et F_8 sont uniformément aléatoires dans Σ_3, et uniformément aléatoires ou adaptées par une exécution de $\texttt{CompleteChain}$ dans Σ_2. Considérons que les fonctions F_i utilisées par le simulateur sont fixées (mais pas la permutation $\boldsymbol{P}$). Alors, comme dans la démonstration du lemme précédent, lors d'une adaptation en $F_3(X)$ et $F_4(Y)$, on a :

$$\begin{cases} F_3(X) = L \oplus F_1(R) \oplus F_5(Z) \oplus F_7(B) \oplus F_9(D) \oplus S \\ F_4(Y) = R \oplus F_2(W) \oplus F_6(A) \oplus F_8(C) \oplus F_{10}(S) \oplus T \\ \boldsymbol{P}(L|R) = S\|T \ . \end{cases}$$

1. Il est fort probable qu'une analyse dédiée du système intermédiaire permette d'obtenir une meilleure majoration de $\Pr[\texttt{Abort}_2]$.

Par conséquent, on voit que la distance statistique entre les valeurs adaptées et des valeurs uniformément aléatoires est égale à la distance statistique entre les réponses de la permutation aléatoire $\boldsymbol{P}$ et des réponses uniformément aléatoires. En utilisant le célèbre « PRP/PRF Switching Lemma » [BR06], qui établit que cette distance (pour q requêtes) est inférieure à $\frac{q^2}{2\cdot 2^{2n}}$, on obtient, en tenant compte du fait que le nombre d'adaptations du simulateur dans le système Σ_2 est inférieur à $(10q)+4\cdot(10q)^2$ car le nombre de F-requêtes au simulateur dans ce système est inférieur à $10q$:

$$|P_2 - P_3| \leq \Pr\left[\mathtt{Abort}_2\right] + \frac{\left((10q) + 4\cdot(10q)^2\right)^2}{2\cdot 2^{2n}} \ ,$$

soit, d'après le lemme 2.13 :

$$|P_2 - P_3| \leq \frac{96\cdot(10q)^4}{2^n} + \mathcal{O}\left(\frac{q^3}{2^n}\right) \leq \frac{2^{20}\cdot q^4}{2^n} + \mathcal{O}\left(\frac{q^3}{2^n}\right) \ ,$$

ce qui conclut la démonstration. ∎

En utilisant les deux lemmes précédents, et compte tenu de l'inégalité $|P_1 - P_3| \leq |P_1 - P_2| + |P_2 - P_3|$, on obtient :

Lemme 2.16. *Pour tout distingueur $\mathcal{D}$ faisant au plus q requêtes, on a :*

$$\left|\Pr\left[\mathcal{D}^{\Psi_{10}^{\boldsymbol{F}},\boldsymbol{F}}(1^k)=1\right] - \Pr\left[\mathcal{D}^{\boldsymbol{P},\mathcal{S}^{\boldsymbol{P}}}(1^k)=1\right]\right| \leq \frac{2^{21}\cdot q^4}{2^n} + \mathcal{O}\left(\frac{q^3}{2^n}\right) \ . \qquad \triangledown$$

Conclusion. La combinaison des lemmes 2.6 et 2.16 permet de conclure quant au théorème 2.4 énoncé au début de cette section.

2.4 Idée de la preuve pour 6 tours

Avec Jean-Sébastien Coron et Jacques Patarin, nous avons prouvé dans [CPS08] que la construction de Luby-Rackoff est indifférentiable d'une permutation aléatoire inversible dès 6 tours . Plus précisément, nous avons montré le résultat suivant :

Théorème 2.17. *Pour tout $q \in \mathbb{N}$, la construction de Luby-Rackoff à 6 tours $\Psi_6^{\boldsymbol{F}}$ est fortement (q,σ,ϵ)-indifférentiable d'une permutation aléatoire inversible $\boldsymbol{P}$, avec :*

$$\sigma(q) = \mathcal{O}(q^8) \quad \textit{et} \quad \epsilon(q) = \frac{2^{24}\cdot q^{16}}{2^n} \ . \qquad \diamondsuit$$

Comme on le voit, le nombre de requêtes effectuées par le simulateur est supérieur et la borne de sécurité est moins bonne que pour 10 tours. De plus le simulateur que nous avons utilisé est beaucoup plus complexe. La stratégie générale consistant à compléter les chaines qui se forment dans l'historique du simulateur reste la même, mais des mécanismes de garde additionnels, dont nous allons essayer de donner l'idée, doivent être ajoutés.

Considérons la figure 2.4 (partie droite) représentant la construction de Luby-Rackoff à 6 tours. Le simulateur fonctionne en complétant les chaines externes et les chaines centrales. Pour les chaines externes il réagit dès qu'une chaine de 3 éléments seulement (et non 4 comme pour 10 tours) se forme : on parlera de chaine (X,R,S) ou (A,S,R). Plus précisément, étant donnée une permutation $\boldsymbol{P}$, nous dirons que :

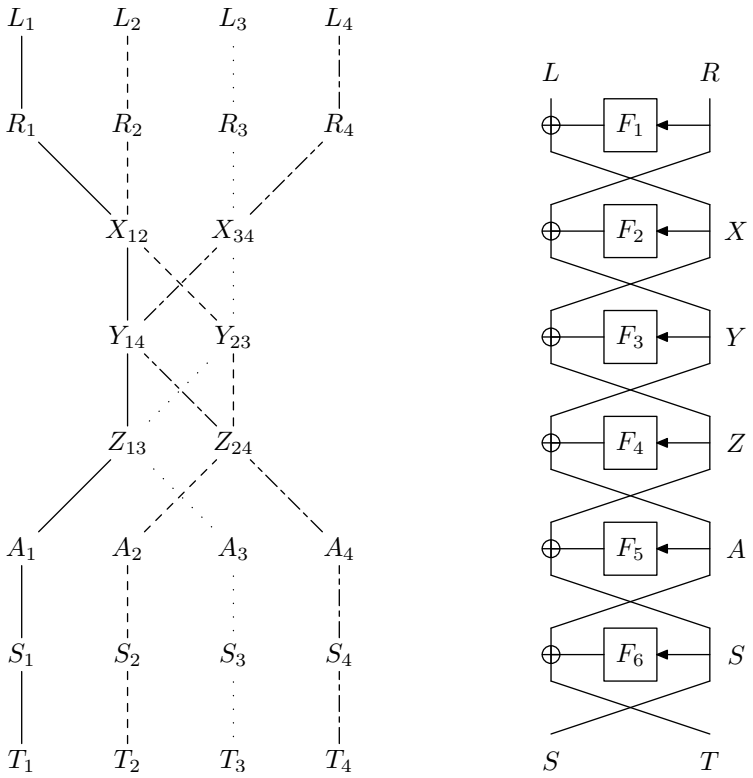

FIGURE 2.4 – Attaque possible sur la construction de Luby-Rackoff à 6 tours contre un simulateur trop simple. Les valeurs (L, R, X, Y, Z, A, S, T) partageant un même indice et reliées par des pointillés de motif identique correspondent à la même valeur d'entrée/sortie de Ψ_6^F. On peut facilement voir que $R_1 \oplus R_2 = R_3 \oplus R_4 = Y_{14} \oplus Y_{23} = A_1 \oplus A_3 = A_2 \oplus A_4$.

- $(X, R, S) \in F_2 \times F_1 \times F_6$ forme une 3-chaine si $\boldsymbol{P}((X \oplus F_1(R))\|R) = S\|T$ pour un T quelconque ;
- $(A, S, R) \in F_5 \times F_6 \times F_1$ forme une 3-chaine si $\boldsymbol{P}^{-1}(S\|(A \oplus F_6(S))) = L\|R$ pour un L quelconque.

Lors de la complétion d'une chaine (X, R, S) le simulateur s'adapte en $F_4(Z)$ et $F_5(A)$, et lors de la complétion d'une chaine (A, S, R) le simulateur s'adapte en $F_2(X)$ et $F_3(Y)$.

Le simulateur complète également les chaines centrales (X, Y, Z) et (Y, Z, A). Un triplet $(X, Y, Z) \in F_2 \times F_3 \times F_4$ forme une 3-chaine si $Z = X \oplus F_3(Y)$. De même un triplet $(Y, Z, A) \in F_3 \times F_4 \times F_5$ forme une 3-chaine si $A = Y \oplus F_4(Z)$. Dans le cas d'une chaine (X, Y, Z) le simulateur s'adapte en $F_5(A)$ et $F_6(S)$ et dans le cas d'une chaine (Y, Z, A) il s'adapte en $F_1(R)$ et $F_2(X)$. De façon équivalente on peut considérer un simulateur complétant tous les centres (Y, Z).

Cependant, si l'on se limite à cela, le distingueur peut employer des techniques proches de l'attaque sur 5 tours de la section 2.2 pour forcer le simulateur à s'adapter en une même valeur pour deux chaines distinctes, ce qui est impossible. Considérons l'attaque suivante (cf. figure 2.4) :

1. $\mathcal{D}$ choisit une valeur X_{12} arbitraire, ainsi que deux valeurs R_1 et R_2 arbitraires, et fait les F-requêtes $F_1(R_1)$ et $F_2(R_2)$.
2. $\mathcal{D}$ calcule $L_1 = X_{12} \oplus F_1(R_1)$ et $L_2 = X_{12} \oplus F_1(R_2)$ et fait les P-requêtes $\boldsymbol{P}(L_1\|R_1) = S_1\|T_1$ et $\boldsymbol{P}(L_2\|R_2) = S_2\|T_2$.
3. $\mathcal{D}$ fait alors les F-requêtes $F_6(S_1)$ et $F_6(S_2)$, et calcule les valeurs :

$$\begin{aligned} A_1 &= T_1 \oplus F_6(S_1) \\ A_2 &= T_2 \oplus F_6(S_2) \\ A_3 &= A_1 \oplus R_1 \oplus R_2 \\ A_4 &= A_2 \oplus R_1 \oplus R_2 \end{aligned}$$

4. enfin, $\mathcal{D}$ fait les F-requêtes $F_5(A_3)$ et $F_5(A_4)$, puis la F-requête $F_2(X_{12})$.

Remarquons que jusqu'à la F-requête $F_2(X_{12})$, aucune chaine de longueur 3 n'a été créée dans l'historique. Supposons donc que le simulateur a répondu à toutes les F-requêtes aléatoirement et analysons ce que déclenche la dernière F-requête $F_2(X_{12})$: on voit qu'elle crée deux chaines externes (X_{12}, R_1, S_1) et (X_{12}, R_2, S_2). Le simulateur va alors définir $F_2(X_{12})$ aléatoirement, puis il calcule $Y_{14} = R_1 \oplus F_2(X_{12})$ et $Y_{23} = R_2 \oplus F_2(X_{12})$. Comme pour l'attaque sur 5 tours, les notations sont choisies de telle sorte que les valeurs partageant un même indice appartiennent à une même chaine, *i.e.* correspondent à la même entrée/sortie de la construction Ψ_6^F. Le simulateur définit également $F_3(Y_{14})$ et $F_3(Y_{23})$ aléatoirement, et calcule $Z_{13} = X_{12} \oplus F_3(Y_{14})$ et $Z_{24} = X_{12} \oplus F_3(Y_{23})$. Il calcule également $A_1 = T_1 \oplus F_6(S_1)$ et $A_2 = T_2 \oplus F_6(S_2)$. Enfin il adapte les deux chaines en définissant :

$$\begin{cases} F_4(Z_{13}) = Y_{14} \oplus A_1 \\ F_5(A_1) = Z_{13} \oplus S_1 \end{cases} \quad \text{et} \quad \begin{cases} F_4(Z_{24}) = Y_{23} \oplus A_2 \\ F_5(A_2) = Z_{24} \oplus S_2 \end{cases}$$

On peut alors remarquer que par construction, les valeurs (Y_{23}, Z_{13}, A_3) et les valeurs (Y_{14}, Z_{24}, A_4) forment deux chaines distinctes. En effet :

$$\begin{aligned} Y_{23} \oplus F_4(Z_{13}) &= Y_{23} \oplus Y_{14} \oplus A_1 \\ &= R_2 \oplus F_2(X_{12}) \oplus R_1 \oplus F_2(X_{12}) \oplus A_1 \\ &= A_3 \end{aligned}$$

car A_3 a été défini par le distingueur par $A_3 = A_1 \oplus R_1 \oplus R_2$. Le raisonnement est similaire pour (Y_{14}, Z_{24}, A_4). Or il est maintenant impossible d'adapter ces deux chaines car elles passent toutes deux par la même valeur de X que nous noterons X_{34}. En effet, $Z_{13} \oplus F_3(Y_{23}) = Z_{24} \oplus F_3(Y_{14})$ car $Z_{13} \oplus F_3(Y_{14}) = Z_{24} \oplus F_3(Y_{23}) = X_{12}$. Comme les valeurs de S_3 et S_4 sont déjà fixées par $S_3 = Z_{13} \oplus F_5(A_3)$ et $S_4 = Z_{24} \oplus F_5(A_4)$ (rappelons que les F-requêtes pour $F_5(A_3)$ et $F_5(A_4)$ ont déjà été faites par le distingueur), le simulateur, s'il veut adapter ces deux chaines, doit trouver deux paires entrée/sortie $(L\|R, S\|T)$ et $(L'\|R', S'\|T')$ de la permutation $\boldsymbol{P}$ vérifiant $S = S_3$, $S' = S_4$, et $R \oplus R' = Y_{14} \oplus Y_{23}$ (cette dernière égalité provenant du fait que les chaines 3 et 4 passent par la même valeur X_{34}). Or ce n'est possible qu'avec une probabilité négligeable avec un nombre polynomial de requêtes à $\boldsymbol{P}$ [2].

Diverses variantes de ces attaques sont possibles et c'est pourquoi il est nécessaire d'empêcher la formation de structures similaires à celle de la figure 2.4. Pour l'attaque que nous venons de présenter, la parade consiste, à chaque F-requête $F_5(A)$, à vérifier pour tous les R_i et R_j dans l'historique de F_1 si la valeur $A' = A \oplus R_i \oplus R_j$ crée une chaine externe (A', S, R), et à compléter alors la chaine correspondante. Nous renvoyons le lecteur courageux à la version complète [3] de l'article de CRYPTO 2008 détaillant la preuve du théorème 2.17.

2.5 Modèles alternatifs d'indifférentiabilité

2.5.1 Modèle honnête mais curieux

Dodis et Punya ont introduit un modèle d'indifférentiabilité dénommé le modèle « honnête mais curieux » [DP06]. Ce modèle (qui n'est pas spécifique à l'indifférentiabilité) ne considère que des attaquants dits honnêtes mais curieux, c'est-à-dire ne déviant pas du comportement normal spécifié par le protocole cryptographique, mais pouvant acquérir de l'information supplémentaire à laquelle ne devrait pas avoir accès un participant légitime. Dans ce modèle, étant donnée une construction $\mathcal{C}^{\boldsymbol{F}}$ implémentant une fonctionnalité $\mathbb{G}$, un algorithme cherchant à distinguer $\mathcal{C}^{\boldsymbol{F}}$ de la primitive idéale $\boldsymbol{G}$ ne peut pas faire de requêtes directement à la fonctionnalité F. Il ne peut faire de requêtes qu'à la fonctionnalité G, mais ces requêtes peuvent être de deux types. Lors d'une requête du premier type, il reçoit simplement la réponse correspondant à $G(x)$. Lors d'une requête du second type, il reçoit également la transcription des échanges entre $\mathcal{C}$ et $\boldsymbol{F}$ nécessaires pour calculer la réponse à la requête qu'il a effectuée. Ainsi, dans le cas de la construction de Luby-Rackoff, il s'agit de toutes les valeurs d'entrée/sortie des fonctions internes nécessaires pour calculer la réponse à la requête. Nous noterons $\mathcal{T}_{\mathcal{C} \leftrightarrow \boldsymbol{F}}$ l'oracle retournant les communications entre $\mathcal{C}$ et $\boldsymbol{F}$.

Dans ce modèle, le rôle du simulateur est de simuler cette transcription pour la primitive idéale $\boldsymbol{G}$. Cependant, une différence supplémentaire avec le modèle d'indifférentiabilité général est que le simulateur ne peut pas faire de requêtes additionnelles à $\boldsymbol{G}$ autres que celles que lui demande le distingueur. Ceci est nécessaire pour pouvoir transformer un attaquant $\mathcal{A}$ honnête mais curieux contre un cryptosystème $\Gamma^{\mathcal{C}^{\boldsymbol{F}}}$ en un attaquant $\mathcal{A}'$ contre $\Gamma^{\boldsymbol{G}}$ à la manière de la démonstration du théorème 1.3 : l'attaquant $\mathcal{A}'$ étant défini comme la

2. En fait, sur cet exemple, il est encore possible d'adapter les 4 chaines au moment de la requête $F_2(X_{12})$, mais des exemples plus complexes montrent que le simulateur est obligé de réagir *avant* qu'une chaine de longueur 3 ne se forme.

3. disponible à `http://eprint.iacr.org/2008/246.pdf`

combinaison de $\mathcal{A}$ et du simulateur, ce dernier ne doit pas faire de requêtes non prescrites par le protocole pour que $\mathcal{A}'$ soit également un attaquant honnête mais curieux.

On obtient la définition formelle de l'indifférentiabilité (faible et forte) dans le modèle honnête mais curieux en remplaçant, dans les définitions 1.3 et 1.4, l'oracle $\boldsymbol{F}$ auquel a accès le distingueur par l'oracle $\mathcal{T}_{\mathcal{C}\leftrightarrow\boldsymbol{F}}$, et en restreignant l'accès du simulateur à la primitive idéale $\boldsymbol{G}$ comme expliqué ci-dessus.

Dodis et Puniya ont montré que pour une certaine classe de constructions, appelées constructions *transparentes*, l'indifférentiabilité dans le modèle honnête mais curieux implique l'indifférentiabilité dans le modèle général.

Définition 2.4 (Construction transparente [DP06])
Une construction $\mathcal{C}^F$ est dite transparente s'il existe un algorithme $\mathcal{M}_{\text{ex}}$, appelé extracteur, ayant accès à $\mathcal{T}_{\mathcal{C}\leftrightarrow F}$, et tel que pour tout x dans le domaine de F, $\mathcal{M}_{\text{ex}}^{\mathcal{T}_{\mathcal{C}\leftrightarrow F}}(x) = F(x)$, le nombre de requêtes de $\mathcal{M}_{\text{ex}}$ à $\mathcal{T}_{\mathcal{C}\leftrightarrow F}$ étant polynomial en k. ♦

Une construction $\mathcal{C}^{\boldsymbol{F}}$ est donc transparente si l'accès à la transcription des échanges entre $\mathcal{C}$ et $\boldsymbol{F}$ permet, avec un nombre polynomial de transcriptions, de calculer la valeur de $\boldsymbol{F}$ en n'importe quel point. Pour une construction transparente, tout se passe alors comme si un participant ayant accès à $\mathcal{T}_{\mathcal{C}\leftrightarrow\boldsymbol{F}}$ avait en fait directement accès à l'oracle $\boldsymbol{F}$, si bien qu'on a de façon naturelle le lemme suivant, dont on trouvera une démonstration dans [DP06] :

Lemme 2.18 ([DP06]). *Une construction transparente faiblement (resp. fortement) indifférentiable dans le modèle honnête mais curieux est faiblement (resp. fortement) indifférentiable dans le modèle général.* ▽

Dodis et Puniya ont montré que la construction de Luby-Rackoff $\Psi_r^{\boldsymbol{F}}$, pour un nombre de tours $r = \mathcal{O}(\log k)$ fonction logarithmique du paramètre de sécurité, est transparente (nous renvoyons à leur article pour une description de l'extracteur). Ils ont également montré que pour un nombre de tours $r = \omega(\log k)$ fonction super-logarithmique du paramètre de sécurité, la construction de Luby-Rackoff est indifférentiable d'une permutation aléatoire inversible dans le modèle honnête mais curieux. Malheureusement, ils ont également montré que pour ce nombre de tours $r = \omega(\log k)$, la construction n'est plus transparente, si bien qu'ils n'ont pas pu conclure quant à l'indifférentiabilité de la construction dans le modèle général.

On peut cependant faire la remarque suivante : la construction de Luby-Rackoff, avec un nombre de tours fonction logarithmique du paramètre de sécurité, *n'est pas* indifférentiable dans le modèle honnête mais curieux d'une permutation aléatoire inversible. En effet, lorsque $r = \mathcal{O}(\log k)$, il est possible, lorsqu'on a accès à la transcription des valeurs des fonctions de tour utilisées, de trouver une entrée $L\|R$ telle que $\Psi_r^{\boldsymbol{F}}(L\|R) = S_0\|T$, S_0 étant une valeur arbitraire et prédéterminée. Pour cela, on utilise simplement l'extracteur décrit par Dodis et Puniya pour obtenir la valeur de $F_r(S_0)$. La dernière étape de l'extracteur consiste justement en l'obtention d'une valeur d'entrée $L\|R$ telle que $\Psi_r^{\boldsymbol{F}}(L\|R) = S_0\|T$. De plus, l'extracteur ne fait que des requêtes directes à la permutation. Or ceci est clairement impossible pour une permutation aléatoire $\boldsymbol{P}$, si bien que pour tout simulateur $\mathcal{S}$, les systèmes $(\Psi_r^{\boldsymbol{F}}, \boldsymbol{F})$ et $(\boldsymbol{P}, \mathcal{S}^{\boldsymbol{P}})$ sont distinguables dans le modèle honnête mais curieux (pour $r = \mathcal{O}(\log k)$).

Comme nous avons montré que la construction de Luby-Rackoff à 6 tours est indifférentiable d'une permutation aléatoire inversible dans le modèle général, on en conclut que l'indifférentiabilité dans le modèle général n'implique pas nécessairement l'indifférentiabilité dans le modèle honnête mais curieux.

2.5.2 Résistance à la corrélation

Pour finir, nous allons nous intéresser à une notion introduite par Canetti, Goldreich et Halevi dans leur article sur les limites du modèle de l'oracle aléatoire [CGH98], et proche de celle d'indifférentiabilité : la *résistance à la corrélation*. Informellement, une construction implémentant une fonctionnalité $\mathbb{G}$ est dite résistante à la corrélation si tout ce qu'il est difficile de faire avec les entrées/sorties de la primitive idéale $\boldsymbol{G}$ est également difficile pour la construction.

Canetti *et al.* ont formulé leurs définitions dans le cadre de familles de fonctions (standard) de $\{0,1\}^*$ vers $\{0,1\}^n$. On peut cependant les étendre au cas de constructions utilisant un oracle, en particulier au cas de constructions implémentant une permutation de $\{0,1\}^{2n}$. Toutes les définitions qui suivent sont énoncées pour ce type particulier de constructions afin de faire écho à l'étude de la construction de Luby-Rackoff, mais peuvent être facilement généralisées. Dans ce cadre, une relation $\mathcal{R}$ d'ordre m désignera un sous-ensemble

$$\mathcal{R} \subset \left(\{0,1\}^{2n} \times \{0,1\}^{2n}\right)^m \ .$$

Nous nous limiterons aux relations efficacement reconnaissables, c'est-à-dire telles qu'il existe un algorithme décidant en temps polynomial si un m-uplet $((x_1,y_1),\ldots,(x_m,y_m))$ vérifie la relation.

Définition 2.5 (Relation évasive)
Une relation $\mathcal{R}$ d'ordre m est dite (q,ϵ)-*évasive* si pour tout algorithme $\mathcal{M}$ accédant à un oracle de permutation aléatoire inversible $\boldsymbol{P}$, auquel il fait au plus q requêtes, on a :

$$\Pr\left[(x_1,\ldots,x_m) \leftarrow \mathcal{M}^{\boldsymbol{P}}(1^k) \ : \ ((x_1,\boldsymbol{P}(x_1)),\ldots,(x_m,\boldsymbol{P}(x_m))) \in \mathcal{R}\right] \leq \epsilon \ .$$

$\mathcal{R}$ est simplement dite évasive si pour toute fonction $q \in \texttt{poly}(k)$, $\mathcal{R}$ est (q,ϵ)-évasive avec $\epsilon \in \texttt{negl}(k)$. ♦

Exemple 2.2. La relation d'ordre 1 $\{(L|R,0_n\|0_n) \ : \ L \in \{0,1\}^n, R \in \{0,1\}^n\}$ n'est pas évasive. En effet, il suffit de faire la requête $\boldsymbol{P}^{-1}(0_n\|0_n)$ pour trouver une paire entrée/sortie qui la satisfasse. Remarquons que cette même relation *est* évasive si la permutation n'est pas inversible, ou dans le cas d'un oracle de fonction aléatoire. ∗

Exemple 2.3. La relation d'ordre 1 $\{(L|0_n,S\|0_n) \ : \ L \in \{0,1\}^n, S \in \{0,1\}^n\}$ est évasive. Un algorithme faisant au plus q requêtes à un oracle de permutation aléatoire inversible a une probabilité de trouver une paire d'entrée/sortie la satisfaisant en $\mathcal{O}\left(\frac{q}{2^n}\right)$. ∗

Exemple 2.4. Considérons l'attaque sur la construction de Luby-Rackoff à 5 tours de la section 2.2. On peut remarquer que cette attaque revient à trouver des entrées/sorties de la construction vérifiant la relation évasive d'ordre 4 :

$$\{((L_1\|R_1,S_1\|T_1),(L_2\|R_2,S_2\|T_2),(L_3\|R_3,S_3\|T_3),(L_4\|R_4,S_4\|T_4)) \ : \\ R_1 \oplus R_2 \oplus R_3 \oplus R_4 = 0, \ S_1 \oplus S_2 \oplus S_3 \oplus S_4 = 0\} \ . \quad *$$

Définition 2.6 (Construction résistante à la corrélation)
Soit $\mathcal{C}^{\boldsymbol{F}}$ une construction utilisant une primitive idéale $\boldsymbol{F}$ et implémentant la fonctionnalité des permutations inversibles de $\{0,1\}^{2n}$. $\mathcal{C}^{\boldsymbol{F}}$ est dite *résistante à la corrélation* si pour tout $m \in \texttt{poly}(k)$, pour toute relation évasive $\mathcal{R}$ d'ordre m, et pour tout algorithme $\mathcal{M}$ accédant à l'oracle $\boldsymbol{F}$, auquel il fait au plus $q \in \texttt{poly}(k)$ requêtes, il existe une fonction $\epsilon \in \texttt{negl}(k)$ telle que :

$$\Pr\left[(x_1,\ldots,x_m) \leftarrow \mathcal{M}^{\boldsymbol{F}}(1^k) \ : \ ((x_1,\mathcal{C}^{\boldsymbol{F}}(x_1)),\ldots,(x_m,\mathcal{C}^{\boldsymbol{F}}(x_m))) \in \mathcal{R}\right] \leq \epsilon \ . \quad ♦$$

L'ensemble des relations évasives représente ce qu'il est difficile de faire pour une permutation aléatoire inversible. La résistance à la corrélation capture donc l'idée que toutes les relations entrées/sorties difficiles à satisfaire pour une permutation aléatoire inversible le sont aussi pour la construction $\mathcal{C}^{\boldsymbol{F}}$. Remarquons qu'il n'est pas nécessaire de donner accès à l'oracle $\mathcal{C}^{\boldsymbol{F}}$ à $\mathcal{M}$ qui peut calculer lui-même $\mathcal{C}^{\boldsymbol{F}}(x)$ via des requêtes à $\boldsymbol{F}$.

La résistance à la corrélation est une propriété plus faible que celle d'être indifférentiable d'une permutation aléatoire inversible. En effet, on a l'implication suivante :

Lemme 2.19. *Si une construction $\mathcal{C}^{\boldsymbol{F}}$ implémentant une permutation inversible est faiblement indifférentiable d'une permutation aléatoire inversible $\boldsymbol{P}$, alors $\mathcal{C}^{\boldsymbol{F}}$ est résistante à la corrélation.* ▽

Démonstration. Supposons $\mathcal{C}^{\boldsymbol{F}}$ non résistante à la corrélation. Alors il existe une relation évasive $\mathcal{R}$ d'ordre m et un algorithme $\mathcal{M}$, faisant $q \in \texttt{poly}(k)$ requêtes à $\boldsymbol{F}$, et capable de trouver, avec une probabilité non négligeable, m entrées/sorties de la construction vérifiant $\mathcal{R}$. On peut alors construire un distingueur $\mathcal{D}$ mettant en défaut l'indifférentiabilité de $\mathcal{C}^{\boldsymbol{F}}$ de la façon suivante : $\mathcal{D}$ exécute $\mathcal{M}$ en répondant à ses requêtes avec son propre oracle F. À l'issue de l'exécution, $\mathcal{M}$ retourne m valeurs $(x_1, \ldots, x_m)$. $\mathcal{D}$ effectue alors les requêtes $P(x_1)$, …, $P(x_m)$ à son oracle de permutation et vérifie si $((x_1, P(x_1)), \ldots, (x_m, P(x_m))) \in \mathcal{R}$. Si c'est le cas, il retourne 1, sinon il retourne 0.

Lorsque le distingueur interagit avec $(\mathcal{C}^{\boldsymbol{F}}, \boldsymbol{F})$, la probabilité qu'il retourne 1 est égale à la probabilité que $\mathcal{M}$ retourne $(x_1, \ldots, x_m)$ tel que $((x_1, \mathcal{C}^{\boldsymbol{F}}(x_1)), \ldots, (x_m, \mathcal{C}^{\boldsymbol{F}}(x_m))) \in \mathcal{R}$, qui est non négligeable par hypothèse. Par contre, lorsqu'il interagit avec $(\boldsymbol{P}, \mathcal{S}^{\boldsymbol{P}})$, alors la combinaison de $\mathcal{D}$ et $\mathcal{S}$ constitue un algorithme effectuant nombre polynomial de requêtes à $\boldsymbol{P}$, si bien que $((x_1, \boldsymbol{P}(x_1)), \ldots, (x_m, \boldsymbol{P}(x_m)))$ n'a qu'une probabilité négligeable de vérifier la relation évasive $\mathcal{R}$. L'avantage du distingueur est donc non négligeable (quel que soit le simulateur) et $\mathcal{C}^{\boldsymbol{F}}$ n'est pas faiblement indifférentiable d'une permutation aléatoire. ■

Nous conjecturons que la réciproque du lemme 2.19 est fausse, cependant nous ne connaissons pas de contre-exemple. Remarquons que la construction de Luby-Rackoff à 5 tours, qui n'est pas indifférentiable d'une permutation aléatoire inversible d'après l'attaque de la section 2.2, n'est pas non plus résistante à la corrélation d'après cette même attaque.

Intuitivement, la résistance à la corrélation est la propriété généralement utile dans un cryptosystème utilisant une permutation comme une boîte noire. On peut cependant construire des cryptosystèmes artificiels où l'existence d'un simulateur est nécessaire (c'est en fait une conséquence directe du théorème 1.3 qui établit que si une construction $\mathcal{C}^{\boldsymbol{F}}$ peut remplacer une primitive idéale $\boldsymbol{G}$ dans tout cryptosystème, alors il existe un simulateur pour $\mathcal{C}^{\boldsymbol{F}}$). Un problème intéressant est donc de formaliser le fait qu'un cryptosystème fasse abstraction de la structure interne d'une construction, afin de « débarrasser» les preuves de la nécessaire description d'un simulateur. On peut ainsi espérer obtenir des preuves plus simples et des bornes de sécurité plus fines.

Enfin, remarquons que la résistance à la corrélation est l'équivalent pour les constructions utilisant une primitive idéale de la notion de « distingueur à clé connue » introduite par Knudsen et Rijmen dans le cadre de constructions standard de chiffrements par blocs [KR07]. Notons que les auteurs présentent une attaque sur une construction de Feistel à 7 tours permettant de trouver des corrélations sur les entrées/sorties de la construction plus rapidement que pour une permutation aléatoire inversible. Cela ne remet cependant pas en cause notre résultat car la construction de Feistel considérée utilise des fonctions de tour ayant une structure particulière (elles sont toutes de la forme $x \mapsto f(x \oplus k_i)$ pour une fonction f fixée).

2.6 Discussion et questions ouvertes

Les cryptographes ont souvent été plus réticents à utiliser le modèle du chiffrement par blocs idéal que celui de l'oracle aléatoire, arguant qu'un chiffrement par blocs idéal est une primitive plus riche et contenant plus de structure qu'un oracle aléatoire.

Le théorème 2.1 indique que le chiffrement par blocs idéal et l'oracle aléatoire sont en fait deux modèles équivalents *d'un point de vue théorique et asymptotique* : si un cryptosystème est sûr dans le modèle du chiffrement par blocs idéal (ou dans le modèle de la permutation aléatoire inversible), alors le chiffrement par blocs idéal peut être remplacé par une construction de Luby-Rackoff à 6 tours utilisant un oracle aléatoire, asymptotiquement sans perte de sécurité. Réciproquement, si un cryptosystème est sûr dans le modèle de l'oracle aléatoire, alors il existe une construction de fonction de hachage utilisant un chiffrement par blocs idéal, pouvant remplacer l'oracle aléatoire, asymptotiquement sans perte de sécurité.

En revanche, le théorème ne dit rien sur la perte de sécurité potentielle lors de l'instanciation dans le modèle standard d'une fonction de hachage ou d'un chiffrement par blocs. En particulier, même si le théorème établit l'équivalence du chiffrement par blocs idéal et de l'oracle aléatoire, nous restons d'avis, pour toutes les raisons évoquées à la section 1.1.2, que l'utilisation du modèle du chiffrement par blocs idéal pour analyser un cryptosystème destiné à être utilisé avec AES reste plus risquée que l'utilisation du modèle de l'oracle aléatoire pour analyser un cryptosystème destiné à être utilisé avec SHA-2.

Sur le plan pratique, les bornes de sécurité des théorèmes 2.4 et 2.17 sont trop lâches pour permettre des applications concrètes pour la conception d'algorithmes de chiffrement par blocs. De plus, comme nous l'avons déjà souligné à la section 1.3.3, un résultat d'indifférentiabilité ne permet plus aucune conclusion lorsque la primitive idéale $\boldsymbol{F}$ utilisée par la construction $\mathcal{C}$ est remplacée par une primitive standard F possédant le moindre « défaut détectable ». Ainsi, il serait dangereux d'utiliser le théorème 2.17 pour construire un chiffrement par blocs à partir d'une fonction de hachage dont on serait convaincu qu'elle est résistante à la pré-image, à la seconde pré-image, et aux collisions (les trois propriétés de sécurité classiques d'une bonne fonction de hachage). En effet, si cette fonction se révélait vulnérable, par exemple, aux multicollisions [Jou04] (ce qui constitue un défaut par rapport à un oracle aléatoire), ceci pourrait être catastrophique pour la sécurité du chiffrement par blocs résultant.

Par ailleurs, si 6 tours sont nécessaires et suffisants pour remplacer une permutation aléatoire inversible par une construction de Luby-Rackoff dans *n'importe quel* cryptosystème sans perte de sécurité, il est possible, pour un cryptosystème particulier, que 6 tours se révèlent superflus. Ainsi, Gentry et Ramzan ont montré [GR04] que dans le chiffrement d'Even et Mansour [EM97], utilisant une permutation aléatoire inversible $\boldsymbol{P}$ pour chiffrer selon $x \mapsto k_2 \oplus \boldsymbol{P}(x \oplus k_1)$, où k_1 et k_2 sont deux clés secrètes, la permutation $\boldsymbol{P}$ peut être remplacée par une construction de Luby-Rackoff à 4 tours sans perte notable de sécurité.

De même, Phan et Pointcheval ont analysé le schéma de chiffrement à clé publique suivant [PP03] : $\phi_{\mathtt{pk}}$ étant une permutation à sens unique à trappe associée à la clé publique $\mathtt{pk}$, le chiffré d'un message $m \in \{0,1\}^{\ell}$ est défini par :

$$\mathcal{E}_{\mathtt{pk}}(m, r) = \phi_{\mathtt{pk}}(\boldsymbol{P}(m, r)) \ ,$$

où $r \xleftarrow{\$} \{0,1\}^{\ell'}$ et $\boldsymbol{P}$ est une permutation aléatoire de $\{0,1\}^{\ell+\ell'}$. Ils ont montré que ce schéma était sûr dans le modèle de la permutation aléatoire inversible. Ils ont ensuite

montré que l'on pouvait remplacer la permutation par une construction de Luby-Rackoff (dissymétrique si $\ell \neq \ell'$) à 3 tours utilisant un oracle aléatoire sans perte notable de sécurité. Ainsi, une analyse dédiée pour un cryptosystème donné pourra en général permettre d'utiliser une construction plus efficace et d'obtenir de meilleures bornes de sécurité.

Pour finir, les principaux problèmes ouverts autour de la thématique de cette partie sont les suivants :

1. Améliorer la borne de sécurité pour la construction de Luby-Rackoff à 6 tours du théorème 2.17, si possible sans trop augmenter la nombre de requêtes du simulateur, voire en le diminuant (notons qu'heuristiquement, si l'historique des deux premières fonctions F_1 et F_2 et des deux dernières fonctions F_5 et F_6 est en $\mathcal{O}(q)$, le nombre d'entrées/sorties sur lesquelles peut potentiellement raisonner le distingueur est en $\mathcal{O}(q^2)$, si bien qu'il semble difficile que le simulateur fasse moins de $\mathcal{O}(q^2)$ requêtes à $\boldsymbol{P}/\boldsymbol{P}^{-1}$). Remarquons que l'avantage ϵ du distingueur dépend du nombre de requêtes σ du simulateur, et qu'il n'est pas évident de définir une figure de mérite caractérisant une bonne simulation (le produit $\epsilon \cdot \sigma$?)
2. Il est connu que la construction de Luby-Rackoff à 3 tours (resp. 4 tours) reste pseudo-aléatoire (resp. pseudo-aléatoire inversible) lorsque les fonctions internes sont remplacées par des permutations [Pir06b]. Il serait donc intéressant de prouver que la construction de Luby-Rackoff à 6 tours reste indifférentiable d'une permutation aléatoire inversible avec des permutations inversibles internes (au lieu de fonctions) [4]. Ceci semble nettement plus dur car le simulateur doit alors également simuler les requêtes inverses aux permutations internes. On peut également considérer de légères variantes de la construction de Luby-Rackoff comme celle utilisée dans MISTY et KASUMI [Mat97, GM01, IYYK01].
3. Étudier d'autres constructions classiques, comme celles permettant d'obtenir une fonction pseudo-aléatoire à partir d'une permutation pseudo-aléatoire [HWKS98, Luc00], dans le cadre de l'indifférentiabilité.

4. Dodis *et al.* ont montré [DPP08] que si $\boldsymbol{P}$ est une permutation aléatoire de $\{0,1\}^n$, alors la construction $\boldsymbol{P} \oplus \boldsymbol{P}^{-1}$ est indifférentiable d'une fonction de $\{0,1\}^n$ vers $\{0,1\}^n$. On peut donc remplacer les fonctions internes de la construction de Luby-Rackoff à 6 tours par cette construction, et la nouvelle construction obtenue est indifférentiable d'une permutation aléatoire inversible de $\{0,1\}^{2n}$, mais effectue 12 appels aux permutations internes.

Deuxième partie

Le problème LPN en cryptographie

Introduction

Cette seconde partie délaisse les modèles idéalisés pour se concentrer sur ce que l'on appelle le modèle standard, dans lequel on ne s'appuie pas sur l'existence supposée d'objets idéaux tels qu'un oracle aléatoire ou un chiffrement par blocs idéal. Cela ne signifie pas pour autant que l'on n'y est pas amené à faire des hypothèses (la cryptographie n'utilisant aucune hypothèse non démontrée et reposant uniquement sur la théorie de l'information est particulièrement inefficace), mais celles-ci sont des conjectures « standard » de la théorie de la complexité, telles que la difficulté de la factorisation ou du logarithme discret. En fait dans toute cette partie nous n'aurons besoin que d'une seule hypothèse concernant la difficulté du problème LPN (*Learning Parity with Noise*). L'étude de ce problème, de sa difficulté et des meilleurs algorithmes permettant de le résoudre fait l'objet du chapitre 3.

Le problème LPN, et plus généralement les problèmes difficiles issus de la théorie de l'apprentissage, ont été utilisés sporadiquement par les cryptographes pour concevoir des cryptosystèmes sûrs. Le domaine a fait l'objet d'une soudaine attention lorsqu'en 2005 Juels et Weis ont proposé un protocole d'authentification dénommé HB^+ dont la sécurité peut être prouvée, dans le modèle des attaque dites actives, moyennant l'hypothèse de la difficulté du problème LPN. De surcroît, ce protocole est suffisamment simple pour être implanté dans des étiquettes radio-fréquence bas-coût. Cependant le protocole HB^+ n'est pas sûr dans le modèle de sécurité le plus général dit des attaques « man-in-the-middle », comme l'ont montré peu de temps après sa publication Gilbert, Robshaw et Sibert. Ce fut le début d'une série de tentatives d'amélioration du protocole HB^+ afin de le rendre résistant à ce type d'attaques tout en conservant son caractère extrêmement simple.

Notre travail sur le problème LPN, en collaboration avec Henri Gilbert et Matt Robshaw, a donné lieu à trois publications :

1. [GRS08a], portant sur la cryptanalyse de trois protocoles d'authentification dérivés de HB^+ ; ceci fait l'objet du chapitre 4 ;
2. [GRS08b], où nous avons proposé notre variante de HB^+ nommée (RANDOM-)$\mathrm{HB}^\#$; ce protocole est présenté au chapitre 5 ;
3. [GRS08c], où nous décrivons un schéma de chiffrement symétrique dont la sécurité peut être prouvée sous l'hypothèse de la difficulté du problème LPN, et que nous exposons au chapitre 6.

Chapitre 3

Introduction au problème LPN

Ce premier chapitre a pour but d'introduire le problème qui va nous occuper dans toute cette seconde partie : le problème LPN. Nous exposons les arguments étayant la conjecture qu'il est difficile à résoudre, étudions les différents algorithmes de résolution existants, puis donnons un aperçu des divers protocoles cryptographiques proposés dont la sécurité repose sur ce problème.

3.1 Définition du problème LPN

Considérons le problème suivant : on cherche à retrouver un vecteur binaire secret $\boldsymbol{x}$ de longueur k. Pour cela, on dispose d'un oracle $\Pi_{\boldsymbol{x}}$ qui, sur requête, renvoie des vecteurs de longueur $k+1$ distribués selon la loi de probabilité :

$$\{\boldsymbol{a} \leftarrow U_k \,:\, (\boldsymbol{a}, \boldsymbol{a}\cdot\boldsymbol{x})\} \ ,$$

où U_k désigne la distribution uniforme sur $\{0,1\}^k$. L'oracle $\Pi_{\boldsymbol{x}}$ renvoie donc des vecteurs aléatoires $\boldsymbol{a}$ et leur produit scalaire avec $\boldsymbol{x}$. Ce problème est très simple à résoudre : il suffit de faire des requêtes à l'oracle jusqu'à obtenir k vecteurs $\boldsymbol{a}_i$ indépendants puis de résoudre le système linéaire formé par ces vecteurs et les bits $\boldsymbol{a}_i \cdot \boldsymbol{x}$. Les vecteurs $\boldsymbol{a}_i$ étant uniformément aléatoires et indépendants, la probabilité, lorsque l'on fait k requêtes, d'obtenir k vecteurs $\boldsymbol{a}_i$ linéairement indépendants est égale à :

$$\prod_{i=1}^{k} \left(1 - \frac{1}{2^i}\right) \ .$$

Ceci résulte d'un dénombrement (aisé) des bases de $\mathrm{GF}(2)^k$. D'après le théorème des nombres pentagonaux d'Euler [Cam94, Chapitre 13], cette probabilité tend vers une quantité strictement positive $p(2) \simeq 0,2887$ lorsque k tend vers $+\infty$. Ainsi, il existe un algorithme qui, avec une probabilité supérieure à $p(2)$, retrouve le vecteur $\boldsymbol{x}$ avec k requêtes à l'oracle et $\mathcal{O}(k^3)$ calculs (il s'agit de la complexité du pivot de Gauss).

Supposons maintenant que l'oracle soit défectueux et qu'au lieu de renvoyer toujours la bonne valeur du produit scalaire, il introduise du bruit : au lieu de renvoyer $\boldsymbol{a}\cdot\boldsymbol{x}$, l'oracle renvoie $\boldsymbol{a}\cdot\boldsymbol{x} \oplus \nu$, où ν est un bit de bruit distribué selon une loi de Bernoulli de paramètre $\eta \in]0, \frac{1}{2}[$ (*i.e.* valant 1 avec probabilité η et 0 avec probabilité $1-\eta$) que nous noterons $\mathtt{Ber}_\eta$. Soit $\Pi_{\boldsymbol{x},\eta}$ l'oracle correspondant, renvoyant donc des vecteurs distribués selon la loi :

$$\{\boldsymbol{a} \leftarrow U_k \,;\, \nu \leftarrow \mathtt{Ber}_\eta \,:\, (\boldsymbol{a}, \boldsymbol{a}\cdot\boldsymbol{x} \oplus \nu)\} \ .$$

Le problème devient alors nettement plus dur dès que le niveau de bruit n'est pas trop faible (typiquement 1/8 ou 1/4). Il a fait l'objet de nombreux travaux en théorie de l'apprentissage, où on le désigne sous le nom de *problème LPN*, pour *Learning Parity with Noise*. En effet, en utilisant le vocabulaire de cette théorie, le problème consiste à « apprendre » une fonction parité $\boldsymbol{a} \mapsto \boldsymbol{a} \cdot \boldsymbol{x}$ inconnue à partir d'exemples bruités de cette fonction. Comme nous allons le voir, il s'agit d'un problème $\mathcal{NP}$-dur, et de nombreux résultats tendent à montrer que ce problème est dur *en moyenne*.

Il existe de nombreuses façons de formaliser le problème LPN. Tel que nous l'avons introduit, il apparaît comme un problème distributionnel, car l'instance à résoudre dépend des réponses aléatoires de l'oracle (cette approche sera développée plus en détail par la suite). Cependant, il est possible de définir un équivalent du problème LPN dans le cadre de la théorie de la complexité « classique », sans définir une distribution sur l'espace des instances. Dans ce cas on préférera l'appeler problème MDP (*Minimum Disagreement Problem*) [CJS94]. Il s'énonce ainsi :

Définition 3.1 (Problème MDP)
Soient q et k deux entiers et $q_0 \leq q$. Soit A une matrice binaire de taille $q \times k$, et $\boldsymbol{z}$ un vecteur de longueur q. Étant donnés A, $\boldsymbol{z}$, et q_0, trouver un vecteur $\boldsymbol{x}$ vérifiant au moins q_0 équations du système $A \cdot \boldsymbol{x} = \boldsymbol{z}$, c'est-à-dire tel que $\texttt{Hwt}(A \cdot \boldsymbol{x} \oplus \boldsymbol{z}) \leq q_0$. ♦

Le problème ainsi défini est dans la classe $\mathcal{NP}$ car le problème décisionnel associé (étant donnée une instance, existe-t-il un $\boldsymbol{x}$ tel que $\texttt{Hwt}(A \cdot \boldsymbol{x} \oplus \boldsymbol{z}) \leq q_0$?) est vérifiable en temps polynomial étant donné un certificat $\boldsymbol{x}$. Dans un article fondateur, Berlekamp, McEliece et van Tilborg ont montré que ce problème est $\mathcal{NP}$-complet [BMvT78]. Plus précisément, le résultat final de leur article est que le problème de décoder un code linéaire aléatoire est $\mathcal{NP}$-complet. Le problème du décodage d'un code linéaire aléatoire est en fait le problème d'optimisation associé au problème MDP : étant donnés la matrice génératrice A d'un code linéaire de dimension k et de longueur q, et un mot reçu $\boldsymbol{z}$, trouver le message $\boldsymbol{x}$ tel que le mot de code associé $A \cdot \boldsymbol{x}$ soit le plus proche de $\boldsymbol{z}$, *i.e.* minimisant $\texttt{Hwt}(A \cdot \boldsymbol{x} \oplus \boldsymbol{z})$. Berlekamp *et al.* n'ont pas travaillé directement sur le problème MDP mais sur un problème équivalent, le Décodage de Syndrome (dénommé Coset Weights dans l'article original), qui utilise non pas la matrice génératrice du code mais la matrice de parité H de taille $q \times (q - k)$. Étant donné un mot reçu $\boldsymbol{z}$, le syndrome associé est $\boldsymbol{s} = \boldsymbol{z} \cdot H$. Le décodage consiste alors à trouver l'erreur $\boldsymbol{e}$ de poids de Hamming minimal vérifiant l'équation $\boldsymbol{s} = \boldsymbol{e} \cdot H$, le message transmis le plus probable étant alors l'unique vecteur $\boldsymbol{x}$ tel que $A \cdot \boldsymbol{x} = \boldsymbol{z} \oplus \boldsymbol{e}$. Le problème Décodage de Syndrome est équivalent au problème MDP car le passage de la matrice génératrice à la matrice de parité d'un code peut se faire en temps polynomial. Berlekamp *et. al.* ont démontré leur résultat en réduisant le Décodage de Syndrome au problème « Three-Dimensional Matching », le 17-ième problème $\mathcal{NP}$-complet de la liste de Karp [Kar72].

Il existe un résultat plus fort que celui de Berlekamp *et al.* dû à Håstad [Hås01]. Considérons le problème d'optimisation correspondant au problème MDP, dénommé *Max-Ek-Lin-2* dans l'article de Håstad (comme nous venons de le voir il s'agit exactement du problème du décodage d'un code linéaire aléatoire) :

Définition 3.2 (Problème *Max-Ek-Lin-2*)
Soient q et k deux entiers. Soient A une matrice binaire de taille $q \times k$, et $\boldsymbol{z}$ un vecteur de longueur q. Étant donnés A et $\boldsymbol{z}$, trouver un vecteur $\boldsymbol{x}$ maximisant le nombre d'équations vérifiées (*i.e.* minimisant $\texttt{Hwt}(A \cdot \boldsymbol{x} \oplus \boldsymbol{z})$). ♦

On dira qu'un algorithme approxime le problème à un facteur c près s'il retourne un vecteur $\boldsymbol{x}$ satisfaisant au moins une fraction $\frac{1}{c}$ des équations. Remarquons que *Max-Ek-Lin-2* est facile à approximer à un facteur 2 près : en effet, un vecteur $\boldsymbol{x}$ aléatoire satisfait la moitié des équations avec une bonne probabilité. Le théorème de Håstad [Hås01, Théorème 5.5] énonce que pour tout $\epsilon > 0$, il est $\mathcal{NP}$-dur d'approximer *Max-Ek-Lin-2* à un facteur $2 - \epsilon$ près. En d'autres termes, il est $\mathcal{NP}$-dur de faire mieux que l'algorithme trivial qui teste des solutions aléatoires !

Tous les résultats précédents concernent la complexité du problème *dans le pire cas*. Or il est bien connu en cryptographie que cela n'est pas suffisant pour construire un cryptosystème sûr car le problème pourrait être facile à résoudre *en moyenne*. Mais nous allons voir que de nombreux arguments viennent étayer la conjecture que le problème LPN est effectivement dur en moyenne.

Commençons tout d'abord par formaliser ce que l'on entend par « résoudre le problème LPN ». La définition suivante est directement inspirée de la théorie de l'apprentissage, avec laquelle nous ferons le lien dans la section 3.2.

Définition 3.3 (Problème LPN)
Soit $\boldsymbol{x}$ un vecteur binaire de longueur k. Soit $\mathcal{A}$ un algorithme ayant accès à l'oracle $\Pi_{\boldsymbol{x},\eta}$. On dit que $\mathcal{A}$ (δ, T, q)-résout le problème LPN de paramètres (k, η) si $\mathcal{A}$ effectue au plus T opérations élémentaires, fait au plus q requêtes à l'oracle, et si :

$$\Pr\left[\boldsymbol{x} \xleftarrow{\$} \{0,1\}^k \,:\, \mathcal{A}^{\Pi_{\boldsymbol{x},\eta}}(1^k) = \boldsymbol{x}\right] = \delta \ ,$$

où la probabilité est prise sur $\boldsymbol{x}$, l'aléa de l'oracle $\Pi_{\boldsymbol{x},\eta}$ et l'aléa de $\mathcal{A}$. ♦

Nous sommes désormais en mesure d'énoncer la conjecture centrale de cette partie, concernant la difficulté du problème LPN, et sur laquelle vont reposer toutes les preuves de sécurité des chapitres qui vont suivre.

Conjecture 3.1. *Le problème LPN est dur,* i.e. *: pour tout $\eta \in]0, \frac{1}{2}[$ fixé, tout algorithme de résolution polynomial (en k) n'a qu'une probabilité négligeable (en k) de résoudre le problème LPN de paramètres (k, η). Plus précisément, si un algorithme $\mathcal{A}$ (δ, T, q)-résout le problème LPN de paramètres (k, η), alors :*

$$q, T \in \mathtt{poly}(k) \Rightarrow \delta \in \mathtt{negl}(k) \ . \qquad \triangledown$$

Les sections suivantes sont consacrées à la présentation des différents arguments en faveur de cette conjecture. Mais avant cela, remarquons que si l'on est autorisé à faire des requêtes à l'oracle LPN pour des vecteurs $\boldsymbol{a}$ *choisis*, alors on peut facilement retrouver le vecteur secret $\boldsymbol{x}$ en temps polynomial. Plus précisément, on a le résultat suivant :

Lemme 3.2. *Soit $\boldsymbol{x}$ un vecteur secret de taille k et soit $\widetilde{\Pi}_{\boldsymbol{x},\eta}$ l'oracle prenant en entrée un vecteur $\boldsymbol{a}$ de longueur k et retournant $\boldsymbol{a} \cdot \boldsymbol{x} \oplus \nu$, où $\nu \leftarrow \mathtt{Ber}_\eta$. Alors il existe un algorithme faisant $q = 8k^2 (1-2\eta)^{-2}$ requêtes à l'oracle, dont le temps de calcul est $T = \mathcal{O}\left(8k^2 (1-2\eta)^{-2}\right)$, et qui retourne le vecteur $\boldsymbol{x}$ avec une probabilité supérieure à $(1 - e^{-k})^k$.* $\triangledown$

Démonstration. La démonstration s'appuie sur les bornes de Chernoff et leur application à la technique du vote majoritaire rappelées dans l'appendice B. L'oracle $\widetilde{\Pi}_{\boldsymbol{x},\eta}$ retourne la bonne valeur de $\boldsymbol{a} \cdot \boldsymbol{x}$ avec probabilité $\frac{1}{2} + \epsilon$, où $\epsilon = \frac{1}{2}(1 - 2\eta)$. Considérons

l'algorithme qui, pour i variant de 1 à k, effectue L requêtes à $\widetilde{\Pi}_{\boldsymbol{x},\eta}$ pour le vecteur de base $\boldsymbol{e}_i$ dont tous les bits valent 0 sauf le i-ième qui vaut 1, puis décide de la valeur du i-ième bit de $\boldsymbol{x}$ en prenant le vote majoritaire sur les réponses. Comme nous le montrons dans l'appendice B, la probabilité d'erreur sur chaque bit est inférieure à $e^{-L\epsilon^2/2}$, donc en prenant $L = 8k\,(1-2\eta)^{-2}$ on obtient une probabilité d'erreur sur chaque bit inférieure à e^{-k}. L'algorithme fait donc $kL = 8k^2\,(1-2\eta)^{-2}$ requêtes à l'oracle et retourne la bonne valeur de $\boldsymbol{x}$ avec une probabilité supérieure à $(1-e^{-k})^k$. ∎

Nous attirons l'attention du lecteur sur la différence entre le lemme précédent et l'un des résultats les plus fondamentaux de la cryptographie, le théorème de Goldreich-Levin [GL89]. Celui-ci concerne le cas où l'oracle $\widetilde{\Pi}_{\boldsymbol{x},\eta}$ introduit un bruit non pas aléatoire mais complètement arbitraire, voire « malveillant », c'est-à-dire choisi par un adversaire : lorsqu'une même requête est faite plusieurs fois, l'oracle peut potentiellement renvoyer la même réponse et non une réponse indépendamment bruitée. Il ne sert alors à rien de faire plusieurs fois la même requête. Le théorème de Goldreich-Levin tel qu'il est énoncé dans [Lub96] établit que lorsque l'oracle renvoie la bonne valeur de $\boldsymbol{a}\cdot\boldsymbol{x}$ avec probabilité $\frac{1}{2}+\epsilon$ (en moyenne sur $\boldsymbol{a}$), on peut retrouver le vecteur secret $\boldsymbol{x}$ avec une probabilité $\frac{1}{2}$, un nombre de requêtes $\mathcal{O}(k^2\epsilon^{-2})$ et en temps $\mathcal{O}(k^3\epsilon^{-4})$. Un meilleur algorithme atteignant un rapport temps-succès $\mathtt{poly}(k)\epsilon^{-2}$ quasiment optimal peut être trouvé dans le livre de Goldreich [Gol01].

3.2 Liens avec la théorie de l'apprentissage

Le problème LPN est certainement l'un des problèmes les plus fondamentaux de la théorie de l'apprentissage. Cette théorie s'intéresse de façon très générale à l'apprentissage de *classes de concepts*. Un *concept* désigne simplement une fonction booléenne d'un ensemble X (généralement $\{0,1\}^k$, comme nous le supposerons par la suite) vers $\{0,1\}$, et une *classe de concepts* $\mathcal{C}$ est un ensemble de concepts. Le modèle d'apprentissage le plus célèbre est le modèle PAC (*Probably Approximately Correct*) dû à Valiant [Val84]. Dans ce modèle, l'ensemble X est muni d'une distribution $\mathcal{D}$ (très souvent la distribution uniforme), et pour un concept c, on définit un « oracle d'exemples » $EX(c,\mathcal{D})$, qui, sur requête, renvoie un exemple $(a, c(a))$, où $a \leftarrow \mathcal{D}$. Pour un concept c, on dit qu'une fonction f approxime c à la précision $\epsilon \geq 0$ pour la distribution $\mathcal{D}$ si :

$$\Pr_{a\leftarrow\mathcal{D}}\left[f(a) = c(a)\right] \geq 1-\epsilon \ .$$

On dit qu'une classe de concepts $\mathcal{C}$ sur $\{0,1\}^k$ peut être apprise efficacement dans le modèle PAC par rapport à la distribution $\mathcal{D}$ s'il existe un algorithme d'apprentissage $\mathcal{A}$ tel que pour tout $\epsilon > 0$, $\delta > 0$, et pour tout concept $c \in \mathcal{C}$, $\mathcal{A}$, ayant accès à l'oracle $EX(c,\mathcal{D})$, retourne avec une probabilité supérieure à $1-\delta$ une fonction hypothèse f qui approxime c avec une précision ϵ. L'algorithme $\mathcal{A}$ est considéré comme efficace s'il est polynomial en k, $\frac{1}{\epsilon}$ et $\frac{1}{\delta}$.

La *fonction parité* associée au vecteur $\boldsymbol{x} \in \{0,1\}^k$ est le concept $c_{\boldsymbol{x}} : \boldsymbol{a} \mapsto \boldsymbol{a}\cdot\boldsymbol{x}$. Il résulte de la discussion du début de la section 3.1 que le concept des fonctions parité peut être appris efficacement dans le modèle PAC par rapport à la distribution uniforme.

Une extension naturelle du modèle PAC introduite par Angluin et Laird [AL87] consiste à considérer un oracle d'exemples bruités. Lorsque le bruit est uniformément aléatoire on parle de *bruit aléatoire de classification* : l'oracle $EX(c,\mathcal{D})$ est remplacé par l'oracle $EX_\eta(c,\mathcal{D})$, pour $\eta \in]0, \frac{1}{2}[$, qui renvoie $(a, c(a)\oplus\nu)$ où $\nu \leftarrow \mathtt{Ber}_\eta$. Dans ce modèle plus

général on requiert, pour qu'un algorithme d'apprentissage soit considéré comme efficace, qu'il soit également polynomial en $\frac{1}{1-2\eta}$.

Reformulé dans ce contexte, le problème LPN consiste à apprendre la classe de concepts des fonctions parité dans le modèle PAC avec bruit aléatoire de classification, et supposer que le problème LPN est dur revient à supposer que ce n'est pas possible efficacement. En fait, ceci n'est pas tout à fait exact et demande les précisions suivantes :

- Pour résoudre le problème LPN, l'algorithme doit retourner exactement le vecteur $\boldsymbol{x}$, alors que dans le modèle PAC il suffit de retourner un vecteur $\boldsymbol{x}'$ fournissant une approximation à la précision ϵ du concept $c_{\boldsymbol{x}}$. Cependant ces deux problèmes sont asymptotiquement équivalents car tout autre vecteur que $\boldsymbol{x}$ ne fournit qu'une approximation à la précision $1/2$ de $c_{\boldsymbol{x}}$ (en effet, si $\boldsymbol{x}' \neq \boldsymbol{x}$, pour $\boldsymbol{a}$ uniformément distribué, $\Pr[\boldsymbol{a} \cdot \boldsymbol{x}' = \boldsymbol{a} \cdot \boldsymbol{x}] = \frac{1}{2}$).
- La définition du problème LPN demande qu'en moyenne sur $\boldsymbol{x}$, la probabilité de succès de l'algorithme de résolution soit non négligeable, alors que dans le modèle PAC l'algorithme doit avoir une probabilité de succès supérieure à $1 - \delta$ pour tout concept, ce qui est a priori plus fort. Nous verrons cependant à la section 3.4 que la propriété d'auto-réductibilité du problème LPN permet de montrer que ces deux définitions sont en fait équivalentes : un algorithme ayant une bonne probabilité de succès en moyenne sur $\boldsymbol{x}$ peut être converti en un algorithme ayant une bonne probabilité de succès *pour tout* $\boldsymbol{x}$.
- Enfin, dans le modèle PAC on demande que la complexité de l'algorithme de résolution ait une dépendance polynomiale en $\frac{1}{1-2\eta}$, alors que la conjecture 3.1 exprime que pour tout η fixé, il n'existe pas d'algorithme de résolution polynomial en k. Or clairement l'existence d'un algorithme de résolution du problème LPN dont le temps de calcul est $\texttt{poly}\left(k, \frac{1}{1-2\eta}\right)$ implique l'existence pour tout η fixé d'un algorithme de résolution dont le temps de calcul est $\texttt{poly}\,(k)$. Ainsi la conjecture 3.1 est à strictement parler plus forte que (*i.e.* implique) celle de l'impossibilité d'apprendre efficacement la classe de concepts des fonctions parité dans le modèle PAC avec bruit aléatoire de classification.

Remarque 3.1. À toute fin utile, signalons qu'il n'est pas nécessaire de supposer que le paramètre de bruit η est connu de l'algorithme de résolution. En effet une technique due à Laird [Lai88] permet, lorsque η est inconnu, de se ramener au cas où η est donné en entrée de l'algorithme de résolution. ∗

L'un des arguments les plus forts en faveur de la difficulté en moyenne du problème LPN est un résultat dû à Kearns [Kea98] relatif au modèle d'apprentissage dit SQ (*Statistical Queries*). Dans ce modèle, l'algorithme d'apprentissage ne peut faire de requêtes qu'à un oracle « global » $STAT(c, \mathcal{D})$ prenant en entrée une « requête statistique » constituée d'un prédicat $\chi : X \times \{0,1\} \to \{0,1\}$ et d'un paramètre de tolérance $\tau \in [0,1]$. $STAT(c, \mathcal{D})$ retourne une estimation $\hat{P}_\chi$ à τ près de la probabilité que $\chi(x, c(x)) = 1$ pour $x \leftarrow \mathcal{D}$. Pour être considéré comme efficace dans ce modèle un algorithme d'apprentissage est restreint à des requêtes telles que χ est évaluable en temps polynomial et telles que l'inverse du paramètre de tolérance est borné par une quantité polynomiale en k, $\frac{1}{\epsilon}$ et $\frac{1}{\delta}$.

Remarquons que la réponse à une requête statistique peut être aisément simulée lorsque l'on a accès à l'oracle $EX(c, \mathcal{D})$. En effet, les bornes de Chernoff impliquent que l'on peut estimer la quantité $\Pr[x \leftarrow \mathcal{D} : \chi(x, c(x)) = 1]$ à τ près avec une probabilité exponentiellement proche de 1 en faisant $\mathcal{O}\left(\frac{1}{\tau^2}\right)$ requêtes à $EX(c, \mathcal{D})$ et en calculant la moyenne

empirique de cette probabilité. Kearns a montré un résultat plus fort [Kea98] : les réponses aux requêtes statistiques peuvent également être simulées efficacement lorsque l'on a accès à un oracle bruité $EX_\eta(c, \mathcal{D})$. Ceci implique que toute classe de concepts pouvant être apprise à partir de requêtes statistiques peut également l'être dans le modèle PAC avec bruit aléatoire de classification. Dans le même article, Kearns a montré que le concept des fonctions parité ne peut pas être appris efficacement à partir de requêtes statistiques. Ceci a été raffiné par Blum *et al.* [BFJ+94] qui ont montré que toute classe de concepts contenant un nombre superpolynomial de fonctions parité ne peut pas être appris efficacement à partir de requêtes statistiques. Ainsi, ces résultats excluent une vaste classe d'algorithmes (en effet les algorithmes de type SQ forment la majorité des algorithmes d'apprentissage) comme solution efficace au problème de l'apprentissage des fonctions parité.

La question de savoir s'il existe des classes de concepts qu'il est possible d'apprendre efficacement en présence de bruit aléatoire de classification mais pas à partir de requêtes statistiques est un problème ouvert important en théorie de l'apprentissage. Une réponse positive a été partiellement donnée par Blum, Kalai et Wasserman [BKW03] comme nous le verrons dans la section 3.5.2 dédiée à leur algorithme de résolution du problème LPN.

3.3 Un lemme utile

Nous explicitons ici une réduction très intéressante pour simplifier les preuves de sécurité, et revenant à reformuler le problème LPN sous la forme d'un problème de distinction de deux distributions. Ce résultat prend la forme du lemme suivant, dû à Katz et Shin [KS06a], qui se sont inspirés du lemme 4.2 de [Reg05].

Lemme 3.3 ([KS06a]). *Supposons qu'il existe un algorithme à oracle $\mathcal{M}$ effectuant au plus T opérations élémentaires, faisant au plus q requêtes à l'oracle, et tel que :*

$$\left|\Pr\left[\boldsymbol{x} \xleftarrow{\$} \{0,1\}^k \,:\, \mathcal{M}^{\Pi_{\boldsymbol{x},\eta}}(1^k) = 1\right] - \Pr\left[\mathcal{M}^{U_{k+1}}(1^k) = 1\right]\right| \geq \delta \ .$$

Alors il existe un algorithme $\mathcal{A}$ de résolution du problème LPN effectuant au plus $T' = \mathcal{O}(T \cdot k\delta^{-2} \log k)$ opérations élémentaires, faisant au plus $q' = \mathcal{O}(q \cdot \delta^{-2} \log k)$ requêtes à l'oracle $\Pi_{\boldsymbol{x},\eta}$, et tel que :

$$\Pr\left[\boldsymbol{x} \xleftarrow{\$} \{0,1\}^k \,:\, \mathcal{A}^{\Pi_{\boldsymbol{x},\eta}}(1^k) = \boldsymbol{x}\right] \geq \frac{\delta}{4} \ . \qquad \triangledown$$

Ainsi, ce lemme exprime que si le problème LPN est dur, aucun algorithme efficace ne peut distinguer les réponses d'un oracle $\Pi_{\boldsymbol{x},\eta}$ de vecteurs tirés selon la distribution uniforme sur $\{0,1\}^{k+1}$ avec une probabilité non négligeable. Ce lemme peut aussi être vu comme une réduction polynomiale du problème décisionnel « ai-je accès à un oracle aléatoire ou à un oracle LPN ? » au problème de recherche « à quel oracle LPN ai-je accès ? » correspondant. Il nous sera très utile pour simplifier certaines preuves de sécurité, notamment au chapitre 6.

3.4 Auto-réductibilité

Le problème LPN possède une propriété remarquable reliant sa difficulté en moyenne à sa difficulté dans le pire cas : il est auto-réductible relativement à la distribution de $\boldsymbol{x}$. Intuitivement, cela signifie que le problème est aussi dur à résoudre pour un $\boldsymbol{x}$ aléatoire

que dans le cas le plus dur. La situation est analogue au cas du logarithme discret : pour un module premier p fixé, soit toutes les instances sont solubles en temps polynomial, soit seule une fraction négligeable des instances le sont. Au passage, nous allons également voir que l'on peut amplifier la probabilité de succès d'un algorithme de résolution d'une quantité notable à une quantité exponentiellement proche de 1. Plus précisément, on a le lemme suivant que nous adaptons du lemme 4.1 d'un article de Regev [Reg05].

Lemme 3.4 (Auto-réductibilité relativement à $\boldsymbol{x}$). *Supposons qu'il existe un algorithme à oracle $\mathcal{A}$ efficace qui, pour une fraction α notable des vecteurs $\boldsymbol{x}$, distingue, avec une probabilité δ notable, $\Pi_{\boldsymbol{x},\eta}$ de U_{k+1}. Alors il existe un algorithme à oracle $\mathcal{A}'$ efficace qui,* pour tout *vecteur $\boldsymbol{x}$, distingue $\Pi_{\boldsymbol{x},\eta}$ de U_{k+1} avec une probabilité exponentiellement proche de 1.* ▽

Démonstration. La démonstration de ce résultat repose sur l'observation suivante. Soit $\boldsymbol{t} \in \{0,1\}^k$ un vecteur quelconque. Si $(\boldsymbol{a}, z)$ est distribué selon $\Pi_{\boldsymbol{x},\eta}$, alors $(\boldsymbol{a}, z \oplus \boldsymbol{a}\cdot\boldsymbol{t})$ est distribué selon $\Pi_{\boldsymbol{x}\oplus\boldsymbol{t},\eta}$. Par contre si $(\boldsymbol{a}, z)$ est distribué selon U_{k+1}, alors $(\boldsymbol{a}, z \oplus \boldsymbol{a}\cdot\boldsymbol{t})$ est également distribué selon U_{k+1}.

Pour simplifier les notations nous noterons :

$$p = \Pr\left[\mathcal{A}^{U_{k+1}}(1^k) = 1\right] \quad \text{et} \quad p_{\boldsymbol{x}} = \Pr\left[\mathcal{A}^{\Pi_{\boldsymbol{x},\eta}}(1^k) = 1\right],$$

où les probabilités sont prises sur les réponses de l'oracle et l'aléa interne de $\mathcal{A}$. Par hypothèse on sait que pour une fraction α de tous les $\boldsymbol{x} \in \{0,1\}^k$,

$$|p_{\boldsymbol{x}} - p| \geq \delta \ .$$

Nous noterons S l'ensemble des vecteurs $\boldsymbol{x}$ vérifiant cette inégalité.

Soit $\mathcal{O}$ l'oracle auquel l'algorithme $\mathcal{A}'$ a accès. $\mathcal{A}'$ répète la procédure suivante un nombre de fois égal à $N_b = k\alpha^{-1}$:

1. appeler $\mathcal{A}^{U_{k+1}}(1^k)$ un nombre de fois égal à $N_a = k\delta^{-2}$; soit N'_a le nombre d'acceptations ; estimer la probabilité d'acceptation dans ce cas par $p(U_{k+1}) = \frac{N'_a}{N_a}$;
2. choisir un vecteur $\boldsymbol{t} \xleftarrow{\$} \{0,1\}^k$;
3. appeler $\mathcal{A}$ un nombre de fois égal à $N_a = k\delta^{-2}$ en répondant à ses requêtes avec des vecteurs de la forme $(\boldsymbol{a}, z \oplus \boldsymbol{a}\cdot\boldsymbol{t})$, où $(\boldsymbol{a}, z)$ est obtenu par une requête à l'oracle $\mathcal{O}$; nous noterons $\widetilde{\mathcal{O}}_{\boldsymbol{t}}$ l'oracle équivalent auquel accède $\mathcal{A}$; soit N''_a le nombre d'acceptations ; estimer la probabilité d'acceptation de $\mathcal{A}$ dans ce cas par $p(\widetilde{\mathcal{O}}_{\boldsymbol{t}}) = \frac{N''_a}{N_a}$;
4. si $\left|p(U_{k+1}) - p(\widetilde{\mathcal{O}}_{\boldsymbol{t}})\right| > \frac{\delta}{2}$, retourner 1 ;
5. sinon continuer ; si aucune des N_b itérations de la procédure n'a abouti à accepter, retourner 0.

Montrons que pour tout $\boldsymbol{x}$, $\mathcal{A}'$ distingue $\Pi_{\boldsymbol{x},\eta}$ de U_{k+1} avec une probabilité exponentiellement proche de 1. Tout d'abord, remarquons que les bornes de Chernoff impliquent (cf. appendice B) que les diverses estimations de probabilité faites par $\mathcal{A}'$ sont précises à $\delta/4$ près avec une probabilité exponentiellement proche de 1. Plus précisément :

$$\Pr\left[|p(U_{k+1}) - p| > \frac{\delta}{4}\right] \leq 2e^{-\frac{\delta^2 N_a}{12}}$$
$$\Pr\left[\left|p(\widetilde{\mathcal{O}}_{\boldsymbol{t}}) - \Pr\left[\mathcal{A}^{\widetilde{\mathcal{O}}_{\boldsymbol{t}}}(1^k) = 1\right]\right| > \frac{\delta}{4}\right] \leq 2e^{-\frac{\delta^2 N_a}{12}} \ ,$$

les probabilités étant prises sur une expérience d'estimation de $p(U_{k+1})$ ou $p(\widetilde{\mathcal{O}}_{\boldsymbol{t}})$. Ainsi le choix de $N_a = k\delta^{-2}$ implique que ces deux probabilités sont inférieures à $2e^{-k/12}$, donc négligeables.

Lorsque $\mathcal{O} = U_{k+1}$, on a $\widetilde{\mathcal{O}}_{\boldsymbol{t}} = U_{k+1}$ donc à chaque itération les deux estimations des phases 1 et 3 sont faites sur la même distribution de probabilité. D'après les inégalités ci-dessus, on a alors à chacune des N_b itérations $\left|p(U_{k+1}) - p(\widetilde{\mathcal{O}}_{\boldsymbol{t}})\right| \leq \frac{\delta}{2}$ avec une probabilité supérieure à $\left(1 - 2e^{-k/12}\right)^2$. Par conséquent $\mathcal{A}'$ ne retourne 1 qu'avec une probabilité inférieure à :

$$N_b \left(1 - \left(1 - 2e^{-k/12}\right)^2\right) = \mathtt{negl}(k) \ .$$

Considérons ensuite le cas où $\mathcal{O} = \Pi_{\boldsymbol{x},\eta}$. À chacune des itérations, $\mathcal{A}$ accède à $\widetilde{\mathcal{O}}_{\boldsymbol{t}} = \Pi_{\boldsymbol{x}\oplus\boldsymbol{t},\eta}$ pour $\boldsymbol{t}$ choisi uniformément dans $\{0,1\}^k$. La probabilité qu'au cours des N_b itérations, au moins un des vecteurs $\boldsymbol{x} \oplus \boldsymbol{t}$ appartienne à S, est supérieure à :

$$1 - (1-\alpha)^{N_b} \geq 1 - e^{-k} \ ,$$

la seconde inégalité découlant du choix de $N_b = k\alpha^{-1}$ et de $(1 - 1/x)^x < 1/e$ pour tout $x \geq 1$. Dans un tel cas, comme $|p_{\boldsymbol{x}\oplus\boldsymbol{t}} - p| \geq \delta$, les estimations $p(U_{k+1})$ et $p(\widetilde{\mathcal{O}}_{\boldsymbol{t}})$ sont distantes d'au moins $\delta/2$ avec une probabilité supérieure à $\left(1 - 2e^{-k/12}\right)^2$. Donc la probabilité que $\mathcal{A}'$ retourne 1 lorsque $\mathcal{O} = \Pi_{\boldsymbol{x},\eta}$ est supérieure à :

$$\left(1 - e^{-k}\right)\left(1 - 2e^{-k/12}\right)^2 = 1 - \mathtt{negl}(k) \ .$$

Ainsi, pour tout vecteur $\boldsymbol{x}$ on a :

$$\left|\Pr\left[\mathcal{A}'^{\Pi_{\boldsymbol{x},\eta}}(1^k) = 1\right] - \Pr\left[\mathcal{A}'^{U_{k+1}}(1^k) = 1\right]\right| \geq 1 - \mathtt{negl}(k) \ .$$

Enfin, notons que si l'algorithme $\mathcal{A}$ est efficace, l'algorithme $\mathcal{A}'$ l'est également puisqu'il appelle $\mathcal{A}$ au plus $N_a + N_b \cdot N_a = \mathcal{O}(k^2\delta^{-2}\alpha^{-1})$ fois, où α et δ sont des fonctions notables par hypothèse. ∎

3.5 Résolution du problème LPN

Dans cette section, nous analysons les différentes approches existantes pour résoudre le problème LPN, et comparons leur efficacité respective.

3.5.1 Méthodes élémentaires

Le méthode de résolution du problème LPN la plus simple consiste à obtenir k réponses $(\boldsymbol{a}_i, z_i)$ de l'oracle $\Pi_{\boldsymbol{x},\eta}$ telles que les $\boldsymbol{a}_i$ soient linéairement indépendants (ce qui arrive avec une probabilité supérieure à $p(2)$ lorsque l'on fait k requêtes, comme nous l'avons vu au début de ce chapitre), et pour lesquelles tous les bits d'erreurs soient nuls. Alternativement, on peut aussi tenter de deviner la position des erreurs, pour les corriger et obtenir k équations exactes. Dans les deux cas, on utilise ensuite le pivot de Gauss pour résoudre le système obtenu. On peut ensuite tester la validité de la solution ainsi trouvée à l'aide de quelques équations supplémentaires fournies par $\Pi_{\boldsymbol{x},\eta}$: la bonne solution vérifiera environ une fraction $(1 - \eta)$ des équations, tandis qu'un faux candidat n'en vérifiera qu'environ la moitié. Étudions plus en détails chacune de ces deux tactiques.

Approche 1 : recherche de k équations sans erreurs.

La probabilité que k équations retournées par l'oracle $\Pi_{\boldsymbol{x},\eta}$ soient toutes exactes est clairement $(1-\eta)^k$. Comme la probabilité que les $\boldsymbol{a}_i$ correspondants forment une base est supérieure à $p(2)$, la probabilité de succès de cette approche est supérieure à $p(2)(1-\eta)^k$. En répétant la procédure avec k nouvelles équations à chaque fois on obtient une probabilité de succès notable avec un nombre de requêtes et en temps repectivement :

$$q = \mathcal{O}\left(\left(\frac{1}{1-\eta}\right)^k\right) \quad \text{et} \quad T = \mathcal{O}\left(k^3\left(\frac{1}{1-\eta}\right)^k\right) \ .$$

On peut facilement réduire le nombre de requêtes nécessaires en partant d'un nombre d'équations suffisamment supérieur à k pour qu'il existe parmi elles k équations sans erreurs, puis les rechercher exhaustivement. Le nombre de requêtes nécessaires est réduit à $\mathcal{O}(k)$, mais la complexité reste équivalente à la méthode précédente. Cette technique a été étudiée en détail par Carrijo *et al.* [CTIN08].

Approche 2 : recherche exhaustive des erreurs.

La seconde approche consiste à obtenir k équations telles que les vecteurs $\boldsymbol{a}_i$ correspondants forment une base de $\{0,1\}^k$, puis à tenter de corriger exhaustivement les erreurs. Puisqu'il y a 2^k vecteurs d'erreur possibles, cette approche requiert une complexité de $\mathcal{O}\left(k^3 2^k\right)$ (en tenant compte du pivot de Gauss).

Cependant le nombre d'erreurs parmi k équations est environ ηk avec une forte probabilité, si bien qu'une mesure plus réaliste de la complexité de cette technique est donnée par :

$$T = \mathcal{O}\left(k^3\binom{k}{\eta k}\right) = \mathcal{O}\left(k^3\, 2^{H_2(\eta)k}\right) \ ,$$

où H_2 est la fonction entropie (cf. appendice C).

Cette méthode de résolution du problème LPN a été récemment étudiée par Golebiewski *et al.* [GMZZ08].

Bien qu'ayant une complexité exponentielle pour tout η fixé, ces techniques élémentaires peuvent s'avérer les plus efficaces pour des niveaux de bruit très faibles (cf. section 3.5.7).

3.5.2 L'algorithme BKW

L'algorithme BKW, du nom de ses inventeurs Blum, Kalai et Wasserman, a été publié en 2000 [BKW03], et s'est avéré fondamental à plus d'un titre. Il s'agit tout d'abord du premier algorithme (légèrement) sous-exponentiel pour résoudre le problème LPN. Il a par ailleurs permis d'établir une séparation entre les deux classes d'algorithmes d'apprentissage évoquées à la section 3.2, à savoir le modèle PAC avec bruit aléatoire et le modèle SQ. Enfin, BKW a eu des retombées dans le domaine de la réduction de réseau puisqu'il a permis d'obtenir les premiers algorithmes en temps $2^{\mathcal{O}(n)}$ pour résoudre le problème du plus court vecteur [AKS01, KS01].

Lorsque l'on essaie de généraliser au problème LPN la méthode de l'élimination gaussienne permettant de retrouver $\boldsymbol{x}$ à partir de produits scalaires $\boldsymbol{a}_i \cdot \boldsymbol{x}$ non bruités, on s'aperçoit que le principal problème vient de ce que le pivot de Gauss écrit chaque vecteur de base comme la somme de $\Omega(k)$ vecteurs $\boldsymbol{a}_i$. Or, lorsque l'on combine ainsi linéairement

des produits scalaires bruités, le bruit sur l'équation résultante tend exponentiellement vers $\frac{1}{2}$. Plus précisément on a le lemme suivant, également appelé *Pilling-Up Lemma* dans le contexte de la cryptanalyse linéaire, et dont la démonstration se fait facilement par récurrence :

Lemme 3.5. *Soient $(\boldsymbol{a}_i, z_i)_{1\leq i\leq s}$, où $z_i = \boldsymbol{a}_i \cdot \boldsymbol{x} \oplus \nu_i$, les réponses obtenues à s requêtes à un oracle LPN $\Pi_{\boldsymbol{x},\eta}$. Alors :*

$$\Pr\left[\left(\bigoplus_{i=1}^{s} \boldsymbol{a}_i\right) \cdot \boldsymbol{x} = \bigoplus_{i=1}^{s} z_i\right] = \frac{1}{2} + \frac{1}{2}(1-2\eta)^s \ . \qquad \triangledown$$

Ceci correspond à un paramètre de bruit $\frac{1}{2} - \frac{1}{2}(1-2\eta)^s$. Ainsi, lorsqu'on utilise le pivot de Gauss avec des équations bruitées, on écrit chaque vecteur de base comme la somme d'environ k vecteurs $\boldsymbol{a}_i$, si bien que l'équation obtenue n'est exacte qu'avec une probabilité exponentiellement proche de $\frac{1}{2}$. L'idée fondamentale de l'algorithme BKW est de parvenir à écrire les vecteurs de base comme la somme d'un *faible* nombre de vecteurs $\boldsymbol{a}_i$. Idéalement, il faudrait parvenir à n'utiliser que $\mathcal{O}(\log k)$ vecteurs afin d'obtenir une équation correcte avec probabilité $\frac{1}{2} + \frac{1}{\mathtt{poly}(k)}$. Nous verrons cependant que l'algorithme BKW, s'il fait mieux que le pivot de Gauss, n'atteint pas cet idéal.

Par la suite nous noterons $\boldsymbol{e}_i$ le vecteur dont toutes les composantes sont nulles sauf la i-ième qui vaut 1. L'algorithme BKW fonctionne en découpant les vecteurs $\boldsymbol{a}_i$ en α blocs de β bits ; nous supposerons donc que $k = \alpha\beta$ (l'algorithme peut facilement être adapté pour $k = \alpha\beta + \beta'$). Nous noterons V_i le sous-espace vectoriel de $\{0,1\}^k$ constitué des vecteurs dont les i derniers blocs de β bits valent 0 et nous appellerons *échantillon* d'ordre i, de taille q et de paramètre de bruit η un ensemble de q paires $(\boldsymbol{a}_i, z_i)$ où les vecteurs $\boldsymbol{a}_i$ sont uniformément et indépendamment distribués dans V_i et $z_i = \boldsymbol{a}_i \cdot \boldsymbol{x} \oplus \nu_i$, où $\nu_i \leftarrow \mathtt{Ber}_\eta$.

L'algorithme BKW utilise $\alpha 2^\beta$ réponses à des requêtes à l'oracle LPN et les combine récursivement pour obtenir des échantillons d'ordre croissant. Au départ ces paires $(\boldsymbol{a}_i, z_i)$ forment un échantillon d'ordre 0, de taille $q_0 = \alpha 2^\beta$ et de paramètre de bruit $\eta_0 = \eta$. L'algorithme les classe selon la valeur de leur dernier bloc de β bits. Puis pour chaque classe C_j, $1 \leq j \leq 2^\beta$, non vide, il choisit aléatoirement une paire $(\boldsymbol{a}, z)$ dans la classe et calcule, pour toutes les autres paires $(\boldsymbol{a}_i, z_i)$ de la classe, la paire $(\boldsymbol{a}'_i = \boldsymbol{a} \oplus \boldsymbol{a}_i, z'_i = z \oplus z_i)$. Puis il écarte la paire $(\boldsymbol{a}, z)$ utilisée. Les paires restantes forment alors un échantillon d'ordre 1 (car tous les bits du dernier bloc des vecteurs $\boldsymbol{a}'_i$ valent 0 et, la paire $(\boldsymbol{a}, z)$ ayant été écartée, ces vecteurs sont uniformément et indépendamment distribués dans V_1), de taille au moins $(\alpha - 1)2^\beta$ (car on a écarté au plus 2^β paires) et de paramètre de bruit $\eta_1 = \frac{1}{2} - \frac{1}{2}(1-2\eta)^2$.

On répète ensuite la procédure récursivement : à partir d'un échantillon d'ordre i, de taille q_i et de paramètre de bruit $\eta_i = \frac{1}{2} - \frac{1}{2}(1-2\eta)^{2^i}$, on crée de la façon décrite ci-dessus un échantillon d'ordre $i+1$, de taille supérieure à $q_i - 2^\beta$ et de paramètre de bruit $\eta_{i+1} = \frac{1}{2} - \frac{1}{2}(1-2\eta)^{2^{i+1}}$.

En répétant cela $\alpha - 1$ fois, on obtient un échantillon tel que les vecteurs $\boldsymbol{a}_i$ n'ont que leur premier bloc non nul, de taille supérieure à 2^β, et de paramètre de bruit $\eta_{\alpha-1} = \frac{1}{2} - \frac{1}{2}(1-2\eta)^{2^{\alpha-1}}$. Comme les vecteurs $\boldsymbol{a}_i$ sont uniformément distribués, le vecteur $\boldsymbol{e}_1$ se trouve parmi eux avec une probabilité supérieure à $1 - \frac{1}{e}$. Lorsque c'est la cas on obtient alors la valeur correcte du premier bit de $\boldsymbol{x}$ avec une probabilité $\frac{1}{2} + \epsilon$, où $\epsilon = \frac{1}{2}(1-2\eta)^{2^{\alpha-1}}$.

Il suffit alors de répéter l'expérience un nombre de fois égal à

$$2k\epsilon^{-2} = 8k\left(\frac{1}{1-2\eta}\right)^{2^\alpha}$$

comme dans l'algorithme du lemme 3.2 pour obtenir la bonne valeur du premier bit de $\boldsymbol{x}$ avec probabilité $1-e^{-k}$ (le fait que le vecteur $\boldsymbol{e}_1$ n'apparaisse qu'avec probabilité $1-\frac{1}{e}$ contribue seulement à multiplier le temps de calcul de l'algorithme par un facteur constant). On procède ensuite de même pour chaque vecteur de base $\boldsymbol{e}_i$ jusqu'à retrouver le vecteur $\boldsymbol{x}$ entier. Chaque vote pour un bit de $\boldsymbol{x}$ requiert un temps de calcul et un nombre de requêtes à l'oracle $\mathcal{O}(\alpha 2^\beta)$. Par conséquent le temps de calcul et le nombre total de requêtes faites à l'oracle par l'algorithme BKW est

$$\mathcal{O}\left(k^2\left(\frac{1}{1-2\eta}\right)^{2^\alpha}\alpha 2^\beta\right) \ ,$$

et l'algorithme retourne la valeur correcte de $\boldsymbol{x}$ avec probabilité $\left(1-e^{-k}\right)^k = 1-\mathtt{negl}(k)$. On a donc le théorème suivant :

Théorème 3.6. *Soient α, β deux entiers tels que $k \leq \alpha\beta$. Alors l'algorithme BKW (q, T, δ)-résout le problème LPN de paramètres (k, η), avec :*

$$q, T = \mathtt{poly}\left(\left(\frac{1}{1-2\eta}\right)^{2^\alpha}, 2^\beta\right) \quad \textit{et} \quad \delta = 1-\mathtt{negl}(k) \ . \qquad \diamondsuit$$

On en déduit le corollaire suivant en posant $\alpha = \gamma \log k$ et $\beta = \frac{k}{\gamma \log k}$, avec $\gamma \in]0,1[$.

Corollaire 3.7. *Soit $\gamma \in]0,1[$ une constante fixée. Pour tout paramètre de bruit η vérifiant $\eta \leq \frac{1}{2} - 2^{-k^{1-\gamma}}$ (et donc en particulier pour tout η fixé), le problème LPN de paramètres (k, η) peut être résolu en temps et avec un nombre de requêtes à l'oracle $2^{\mathcal{O}(k/\log k)}$.* $\triangledown$

L'algorithme BKW écrit chaque vecteur de base comme la somme de $2^{\alpha-1} = \mathcal{O}(k^\gamma)$ vecteurs $\boldsymbol{a}_i$, et pour cela il utilise un temps de calcul et un nombre de requêtes à l'oracle égaux à $\alpha 2^\beta = \gamma \log k \, 2^{k/\gamma \log k}$. C'est mieux que le pivot de Gauss, mais rien n'exclut qu'il soit possible de faire la même chose en utilisant moins de vecteurs et avec une complexité moindre. Ainsi, s'il existait un algorithme permettant de trouver, parmi seulement $2^{k^{1-\gamma}}$ vecteurs aléatoires, au plus $\mathcal{O}(k^\gamma)$ vecteurs dont la somme égale un certain vecteur cible en temps $2^{\mathcal{O}(k^{1-\gamma})}$ (ce qui est théoriquement possible puisque le nombre de sous-ensembles de k^γ vecteurs parmi $2^{k^{\gamma-1}}$ est environ $2^{k^\gamma \cdot k^{1-\gamma}} = 2^k$), cela permettrait d'obtenir des équations $\boldsymbol{e}_i \cdot \boldsymbol{x}$ correctes avec une probabilité $\frac{1}{2} + \frac{1}{2}(1-2\eta)^{\mathcal{O}(k^\gamma)}$. En utilisant $\gamma = \frac{1}{2}$, on pourrait par conséquent résoudre le problème LPN avec un nombre de requêtes q et en temps T tels que :

$$q, T = \mathtt{poly}\left(\left(\frac{1}{1-2\eta}\right)^{\mathcal{O}(\sqrt{k})}, 2^{\mathcal{O}(\sqrt{k})}\right) \ ,$$

soit $2^{\mathcal{O}(\sqrt{k})}$ pour η fixé. Il semble que l'on soit loin cependant d'obtenir un tel algorithme.

L'algorithme BKW est important en théorie de l'apprentissage car il permet d'établir une séparation entre le modèle PAC avec bruit aléatoire de classification et le modèle SQ. Si l'on se restreint aux fonctions parité ne dépendant que des $k' = \mathcal{O}(\log k \log\log k)$ premiers bits de $\boldsymbol{x}$, la complexité de l'algorithme BKW devient :

$$2^{\mathcal{O}\left(\frac{k'}{\log k'}\right)} = 2^{\mathcal{O}(\log k)} = \mathtt{poly}(k) \ .$$

La classe des fonctions parité ne dépendant que de leurs $\mathcal{O}(\log k \log\log k)$ premiers bits peut donc être apprise efficacement dans le modèle PAC avec bruit aléatoire de classification. Or cette classe contient $2^{\log k \log\log k} = k^{\log\log k}$ fonctions, soit un nombre superpolynomial de fonctions parité. Comme nous l'avons vu dans la section 3.2, une telle classe ne peut pas être apprise efficacement dans le modèle SQ. L'algorithme BKW fournit donc une séparation entre ces deux modèles *lorsque l'on autorise une dépendance arbitraire de la complexité de l'algorithme d'apprentissage en* $\frac{1}{1-2\eta}$. En effet, comme l'indique le théorème 3.6, la complexité de BKW n'est pas polynomiale en $\frac{1}{1-2\eta}$ (en particulier, pour apprendre efficacement les fonctions parité ne dépendant que de leurs $\mathcal{O}(\log k \log\log k)$ premiers bits, il faut $\eta \leq \frac{1}{2} - 2^{-(\log k)^{1-\gamma}}$ pour $\gamma \in]0,1[$ fixé). Améliorer la dépendance de l'algorithme BKW au paramètre de bruit est un problème ouvert important qui a notamment été étudié par Jackson [Jac03].

3.5.3 Les variantes de Levieil et Fouque

Levieil et Fouque ont proposé deux améliorations de l'algorithme BKW [LF06, Lev08]. La première, baptisée algorithme LF1, consiste à utiliser toutes les équations de l'échantillon d'ordre $\alpha - 1$ obtenues à la dernière phase (et non une seule comme c'est le cas pour l'algorithme BKW), et à chercher quelle valeur des β bits de $\boldsymbol{x}$ considérés maximise le nombre d'équations vérifiées. Rappelons que la taille $q_{\alpha-1}$ de l'échantillon d'ordre $\alpha - 1$ est supérieure à 2^β. Pour ce faire, plutôt que d'utiliser une recherche exhaustive de complexité $\mathcal{O}(2^{2\beta})$ (pour chacune des 2^β valeurs des bits de $\boldsymbol{x}$, il faut calculer 2^β équations), il est préférable d'utiliser la transformée de Walsh [CJM02, Doo03, BGM06] afin d'obtenir une complexité $\mathcal{O}(\beta\, 2^\beta)$.

La seconde variante, LF2, consiste, lors du calcul de l'échantillon d'ordre $i+1$ à partir de l'échantillon d'ordre i, à considérer tous les couples d'éléments possibles dans chaque classe C_j. D'un point de vue théorique on perd l'indépendance entre les vecteurs dans l'échantillon d'ordre supérieur nécessaire pour démontrer rigoureusement les performances de l'algorithme BKW. Cependant les expériences menées par Levieil et Fouque semblent montrer que cette perte d'indépendance théorique n'est pas un réel problème en pratique. De plus cet algorithme amplifie le nombre d'équations disponibles au cours de son exécution et peut donc être utilisé avec un nombre d'équations de départ bien plus restreint que l'algorithme BKW.

3.5.4 La variante de Lyubashevsky

Dans leur article [BKW03], Blum *et al.* remarquent que leur algorithme gaspille énormément les équations. En effet, du point de vue de la quantité d'information disponible, $\mathcal{O}(k)$ équations seulement suffisent pour retrouver le vecteur $\boldsymbol{x}$ avec une forte probabilité. Ils soulèvent donc le problème de trouver un algorithme dont le temps de calcul reste sous-exponentiel mais n'utilisant que $\texttt{poly}(k)$ requêtes à l'oracle LPN.

Ce problème a été résolu par Lyubashevsky qui a proposé un algorithme de résolution du problème LPN utilisant $k^{1+\epsilon}$ requêtes à l'oracle, de complexité $2^{\mathcal{O}(k/\log\log k)}$ (donc légèrement supérieure à celle de l'algorithme BKW) et dont la probabilité d'erreur est négligeable en k. Le paramètre de bruit doit cependant vérifier $\eta \leq \frac{1}{2} - 2^{-(\log k)^\gamma}$ pour toute constante $\gamma \in]0,1[$.

L'algorithme de Lyubashevsky utilise en fait l'algorithme BKW comme une boîte noire et lui fournit le grand nombre d'équations dont il a besoin en les construisant à partir des

$k^{1+\epsilon}$ paires $(\boldsymbol{a}_i, z_i)$ dont il dispose. Pour cela, il combine linéairement $N = \left\lceil \frac{2k}{\epsilon \log k} \right\rceil$ paires aléatoirement choisies pour construire des paires $(\boldsymbol{a}', z')$ telles que :

$$(\boldsymbol{a}' = \bigoplus_{i \in S} \boldsymbol{a}_i, z' = \bigoplus_{i \in S} z_i) \ ,$$

où S est un sous-ensemble aléatoire de $[\![1, k^{1+\epsilon}]\!]$ de N éléments. On peut alors montrer, en utilisant le *Leftover Hash Lemma* [IZ89], que les vecteurs $\boldsymbol{a}'$ ainsi construits sont presque uniformément distribués. Cependant le fait de combiner ainsi les équations augmente le bruit résultant, ce qui explique que la complexité de l'algorithme de Lyubashevsky soit légèrement supérieure à celle de l'algorithme BKW.

3.5.5 Liens avec les algorithmes de décodage

Comme nous l'avons déjà souligné, le problème LPN comporte de nombreuses similitudes avec le problème du décodage d'un code linéaire aléatoire. En effet, si l'on considère q réponses $(\boldsymbol{a}_i, z_i)$ de l'oracle $\Pi_{\boldsymbol{x},\eta}$, on peut former la matrice A dont la i-ième ligne est égale à $\boldsymbol{a}_i$ ainsi que le vecteur $\boldsymbol{z} = (z_1, \ldots, z_q)$. La matrice A est alors la matrice génératrice d'un code linéaire aléatoire de longueur q et de dimension k, et $\boldsymbol{z}$ est le mot reçu, c'est-à-dire le mot de code $A \cdot \boldsymbol{x}$ entaché de l'erreur $\boldsymbol{\nu} = (\nu_1, \ldots, \nu_q)$. Résoudre le problème LPN équivalent revient alors à retrouver le message initial $\boldsymbol{x}$, c'est-à-dire à décoder le code linéaire correspondant à la matrice A. La principale différence conceptuelle tient au fait qu'un algorithme de décodage doit utiliser une longueur de code q fixée et passée en paramètre, tandis qu'un algorithme de résolution du problème LPN peut effectuer un nombre arbitraire de requêtes à l'oracle LPN.

Les codes correcteurs employés dans la pratique tels que les codes de Reed-Solomon possèdent une structure particulière permettant un décodage efficace. Par contraste, les codes linéaires aléatoires ne sont jamais utilisés car leur décodage est un problème $\mathcal{NP}$-complet, donc peu susceptible d'avoir une solution rapide. Par conséquent le décodage d'un code linéaire aléatoire a reçu peu d'attention jusqu'à la proposition de cryptosystème asymétrique de McEliece [McE78], dont la sécurité repose justement sur ce problème. Les tentatives successives de cryptanalyse de ce cryptosystème ont alors permis d'améliorer les algorithmes de décodage d'un code linéaire aléatoire. Les premiers travaux afférents sont dus à Leon [Leo88], Lee et Brickell [LB88], et van Tilburg [vT88]. Puis Stern [Ste88] a proposé un algorithme permettant de trouver un mot de code de poids faible dans un code linéaire aléatoire, ce qui permet également de décoder : il suffit de considérer le code de matrice génératrice $A \| \boldsymbol{z}$, où $\boldsymbol{z}$ est le mot reçu ; un mot de code de poids minimal est alors le vecteur d'erreur $\boldsymbol{\nu}$. Cet algorithme a ensuite été amélioré par Chabaud [Cha94], Canteaut et Chabanne [CC94], Canteaut et Chabaud [CC98], et Canteaut et Sendrier [CS98a]. De nouvelles améliorations ont été très récemment apportées par Bernstein *et al.* [BLP08]. L'analyse asymptotique de la complexité de ces algorithmes n'est pas aisée car ils dépendent de nombreux paramètres, cependant les propositions 7 et 8 de [CC95] montrent qu'elle est exponentielle.

3.5.6 Liens avec les attaques par corrélation rapides

La cryptanalyse de chiffrements à flot mène très naturellement à des problèmes équivalents au problème LPN, notamment dans les *attaques par corrélation*. En effet, une vaste classe d'algorithmes de chiffrement à flot sont constitués d'un ou plusieurs registres à décalage à rétroaction linéaire (LFSR pour *Linear Feedback Shift Register*), filtrés par une

fonction non linéaire f. Il existe alors toujours des corrélations entre les bits de sortie z_i de l'algorithme de chiffrement et l'état initial du LFSR $\boldsymbol{x} = (x_1, \ldots, x_k)$, corrélations que l'on peut exprimer comme des relations linéaires bruitées $\Pr[z_i = \boldsymbol{a}_i \cdot \boldsymbol{x}] = \frac{1}{2} + \epsilon$.

Les attaques par corrélation ont été introduites par Siegenthaler [Sie84, Sie85], puis améliorées par Meier et Staffelbach en utilisant des techniques de décodage (et rebaptisées au passage attaques par corrélation rapides) [MS88, MS89]. Une longue liste de travaux s'en est suivie, notamment ceux de Johansson et Jönsson [JJ99b, JJ99a, JJ00], et Mihaljevic, Fossorier et Imai [MFI00, MFI01]. Une étude détaillée de ces attaques d'un point de vue algorithmique a été faite par Chose *et al.* [CJM02], et un exemple récent d'application contre l'algorithme de chiffrement à flot Grain est dû à Berbain *et al.* [BGM06].

Généralement, toutes ces attaques combinent recherche exhaustive sur une partie de l'état initial du LFSR avec des techniques de décodage ou des idées proches de celles de l'algorithme BKW et des astuces d'implémentation comme l'utilisation de la transformée de Walsh. Remarquons cependant que les paramètres de bruit issus des attaques par corrélation rapides sont souvent très proches de $\frac{1}{2}$, ce qui rend l'algorithme BKW, dont les performances se dégradent fortement lorsque η tend vers $\frac{1}{2}$, peu efficace en pratique. Fossorier *et al.* [FMI+06] ont étudié la transposition de ces techniques au problème LPN proprement dit, et leurs résultats confirment qu'elles s'avèrent plus efficaces que l'algorithme BKW pour certaines classes de paramètres.

3.5.7 Performances concrètes

Il serait intéressant d'étudier précisément les zones du plan (k, η) dans lesquelles chacune des méthodes de résolution du problème LPN se révèle la plus efficace. Voici une analyse rapide permettant de comparer les méthodes semblables à l'algorithme BKW (donc celle de Levieil et Fouque et celle de Lyubashevsky), et les méthodes plus élémentaires évoquées à la section 3.5.1. La complexité de l'algorithme BKW et de ses dérivés est dominée, à des termes polynomiaux en k près, par le terme $2^{k/\log k}$. Celui des méthodes élémentaires est plutôt de l'ordre de $\left(\frac{1}{1-\eta}\right)^k = 2^{-k\log(1-\eta)}$. Par conséquent l'algorithme BKW devient plus performant lorsque :

$$\frac{1}{\log k} < -\log(1-\eta) \quad \Leftrightarrow \quad \eta > 1 - 2^{-\frac{1}{\log k}} \ .$$

Ceci donne donc un ordre de grandeur, pour un k donné, du paramètre de bruit à partir duquel l'algorithme BKW devient plus performant que les méthodes élémentaires. Ainsi, pour $k = 256$, le seuil se situe à $8,3\%$ environ, et pour $k = 512$ à $7,4\%$ environ. Ceci ne constitue qu'une approximation grossière, mais une bonne règle semble être que pour $\eta < 10\%$, les méthodes élémentaires sont les plus efficaces (ce fait a été constaté expérimentalement par Levieil [Lev08]).

Pour conclure sur les méthodes de résolution du problème LPN, nous donnons dans la table 3.1 quelques exemples numériques de complexité pour des paramètres typiques. Pour cela, nous avons comparé les complexités de résolution annoncées dans la littérature (notamment [Lev08, CTIN08]) pour différentes valeurs des paramètres (k, η). Il en ressort que les variantes des méthodes élémentaires sont les plus rapides pour $\eta = 0,1$, et que les variantes de BKW sont meilleures à partir de $\eta = 0,125$. La table 3.1 est tirée de [Lev08]. Pour la référence [CTIN08], nous attirons l'attention du lecteur sur le fait que les complexités annoncées sont *par bit de clé*.

$\eta \backslash k$	128	256	512	768
$0,1$	2^{19}	2^{38}	2^{72}	2^{97}
$0,125$	2^{24}	2^{43}	2^{73}	2^{105}
$0,25$	2^{32}	2^{51}	2^{85}	2^{121}
$0,4$	2^{40}	2^{62}	2^{101}	2^{143}

TABLE 3.1 – Complexité des meilleurs algorithmes de résolution du problème LPN en fonction de (k, η). Valeurs tirées de [Lev08].

3.6 Protocoles cryptographiques reliés

Les liens entre théorie de l'apprentissage et cryptographie sont profonds. Comme l'ont montré Impagliazzo et Luby [IL90], la possibilité même de faire de la cryptographie exclut celle d'un apprentissage efficace « universel » et réciproquement. Plus précisément, Impagliazzo et Luby ont montré l'équivalence entre l'existence de fonctions à sens unique et l'existence de fonctions impossibles à « extrapoler » dans tout sens raisonnable du terme.

Les sections précédentes ont permis de présenter des résultats convergeant vers la même conclusion : le problème LPN est un problème difficile en moyenne. C'est donc tout naturellement que les cryptographes ont cherché à concevoir des protocoles cryptographiques reposant sur ce problème.

Des protocoles utilisant le problème du décodage d'un code correcteur linéaire aléatoire ont été proposés très tôt. Le plus célèbre d'entre eux est le système de chiffrement asymétrique de McEliece [McE78], et sa variante due à Niederreiter [Nie86] qui a été étendue à la signature par Courtois *et al.* [CFS01]. Stern a également proposé un protocole d'authentification sans transfert de connaissance reposant sur le problème du Décodage de Syndrome [Ste93].

Les premiers protocoles cryptographiques fondés sur des problèmes d'apprentissage difficiles et prenant en compte la difficulté en moyenne de ces problèmes ont été proposés par Blum *et al.* [BFKL93]. Dans cet article, Blum *et al.* ont proposé un générateur de bits pseudo-aléatoire, une fonction à sens unique et un système de chiffrement à clé secrète fondés sur la difficulté d'apprendre des classes de concepts très générales, ainsi qu'un générateur de bits pseudo-aléatoire fondé sur le problème LPN.

Enfin, Regev a proposé [Reg05] un système de chiffrement asymétrique basé sur le problème LWE (*Learning With Error*), qui est la généralisation du problème LPN à un corps $\mathrm{GF}(p)$, $p > 2$. Dans le même article, il a montré que le problème LWE peut être réduit à deux des problèmes de réseaux les plus importants, le problème du plus court vecteur (SVP) et le problème des plus courts vecteurs indépendants (SIVP). Cependant, cette réduction est quantique (mais Regev conjecture qu'on peut la rendre classique) et n'est possible que pour des corps suffisamment grands (plus précisément il faut $p > 2\sqrt{k}$). Elle ne s'applique donc pas au problème LPN proprement dit.

Chapitre 4

Premiers protocoles d'authentification

Dans ce chapitre, nous introduisons tout d'abord les notions générales relatives aux protocoles d'authentification, et la problématique de leur implantation dans des environnements d'exécution contraints. Puis nous présentons les deux premiers protocoles d'authentification fondés sur le problème LPN qui ont été proposés, HB et HB^+, et expliquons pourquoi ils ne sont pas sûrs dans certains modèles d'attaques. Enfin, nous exposons la cryptanalyse de trois protocoles qui ont été proposés pour rendre HB^+ résistant à ces attaques (HB^{++}, HB^* et HB-MP), résultats qui ont fait l'objet de la publication [GRS08a].

4.1 Introduction aux protocoles d'authentification

Un protocole d'*authentification d'entité* ou protocole d'*identification* (à distinguer des schémas d'authentification de message) implique deux entités : le « prouveur » et le « vérifieur ». Le but du prouveur est de convaincre le vérifieur de son identité. Pour cela, le prouveur et le vérifieur échangent un certain nombre de messages (à raison d'un message par *passe*) à l'issue desquels le vérifieur prend la décision binaire d'accepter ou de rejeter l'authentification. Contrairement aux schémas d'*authentification de message*, il n'y a pas d'autre information échangée entre le prouveur et le vérifieur que l'affirmation de l'identité du prouveur. De plus, les protocoles d'authentification d'entité impliquent le plus souvent une communication *en temps réel* entre le prouveur et le vérifieur, alors que l'authentification de message permet une vérification de l'origine du message par le destinataire arbitrairement postérieure dans le temps. Par la suite nous considérerons le prouveur et le vérifieur comme deux machines de Turing probabilistes interactives [Gol01, Chapitre 4] que nous noterons respectivement $\mathcal{P}$ et $\mathcal{V}$.

Un protocole d'authentification peut être de type asymétrique ou symétrique. Dans le premier cas, le prouveur possède une clé privée sk et le vérifieur possède la clé publique correspondante pk. Les méthodes d'authentification les plus courantes, comme celles décrites dans la norme ISO/IEC 9798-3, consistent à envoyer au prouveur un challenge chiffré avec sa clé publique que celui-ci doit alors déchiffrer, ou à envoyer un challenge que le prouveur doit signer à l'aide de sa clé privée. D'autres protocoles plus complexes possèdent la propriété « zero-knowledge » (sans transfert de connaissance), c'est-à-dire que le prouveur ne divulgue aucune autre information lors de l'exécution du protocole que le fait qu'il est bien en possession de sa clé privée. C'est le cas notamment des protocoles de Fiat-Shamir [FS86], Guillou-Quisquater [GQ88], et Schnorr [Sch89].

Tous les protocoles que nous allons considérer par la suite sont symétriques : le prouveur et le vérifieur partagent une clé secrète K. Là encore il existe des protocoles génériques qui emploient un chiffrement par blocs (par exemple la norme ISO/IEC 9798-2) ou un code d'authentification de message (norme ISO/IEC 9798-4), utilisés selon un mode challenge-réponse à 2 ou 3 passes suivant que l'on désire obtenir de l'authentification unilatérale ou mutuelle entre les deux participants. Pour une introduction complète aux protocoles d'authentification d'entité on se référera au chapitre 10 du *Handbook of Applied Cryptography* [MvOV96].

4.2 Environnements contraints

Les exemples de protocoles d'authentification précédemment cités remplissent leur fonction de façon sûre et efficace, mais ils peuvent s'avérer difficiles à implanter dans des environnements à capacités de calcul restreintes tels que les étiquettes radio-fréquence (en anglais *RFID*, pour *Radio-Frequency Identification*) à bas-coût. Ces puces sont destinées à des utilisations de masse telles que l'accès aux transports publics, le contrôle d'accès, les systèmes de paiement, les inventaires, le traçage et la localisation des livres dans les bibliothèques, des bagages dans les aéroports, etc. Elles sont souvent considérées comme le « code-barre » du futur. La possibilité de lire ces étiquettes à distance soulève pour la majorité de ces applications des problèmes de sécurité (protection contre le clonage et l'usurpation de droits) et de respect de la vie privée (confidentialité des informations stockées, intraçabilité, etc.). Dans les cas d'applications où la sécurité est très importante et le prix de manufacture de la puce n'est pas un facteur limitant (par exemple pour les passeports électroniques), on peut se permettre d'implanter des algorithmes lourds tels que RSA. Mais pour les utilisations où la pression économique due à la nécessité d'un déploiement à très grande échelle se conjugue à des enjeux de sécurité non négligeables (citons par exemple la protection contre la contrefaçon pour les médicaments ou les produits de luxe), il n'existe pas à l'heure actuelle de solution complètement satisfaisante au problème de l'authentification d'une puce bas-coût.

De nombreux paramètres entrent en jeu lorsque l'on s'intéresse à l'implantation hardware d'un protocole cryptographique : le plus important est le nombre de portes logiques nécessaires (c'est lui qui détermine principalement le coût de la puce), mais interviennent également la mémoire, le temps de calcul, la consommation de courant moyenne et maximale, ainsi que le nombre total de bits échangés entre le prouveur et le vérifieur. Ainsi, en ce qui concerne les algorithmes asymétriques, l'implantation typique de l'algorithme RSA requiert plus de 100 000 portes logiques, ce qui le réserve aux puces « haut de gamme ». Les algorithmes utilisant les courbes elliptiques sont beaucoup plus efficaces (Batina *et al.* [BGK+06] ont obtenu des implantations en 15 000 portes logiques) mais encore trop gourmands pour les puces bas-coût. En revanche, l'algorithme crypto-GPS [GPS06, MR07] peut être implanté avec moins de 1 000 portes logiques mais nécessite l'emploi de valeurs pré-calculées appelées « coupons ».

Les algorithmes d'authentification utilisant des algorithmes de chiffrement par blocs sont une alternative possible à la cryptographie asymétrique. La meilleure implantation de l'algorithme de chiffrement par blocs AES requiert environ 3 600 portes logiques [FDW04], ce qui le qualifie comme solution potentielle pour des puces de milieu de gamme. Nous avons récemment collaboré à la conception d'un algorithme de chiffrement par blocs, PRESENT [BKL+07, RPLP08], spécialement conçu pour permettre une implantation peu coûteuse (entre 1 000 et 1 500 portes logiques).

Enfin, Shamir a récemment proposé un schéma d'authentification symétrique reposant sur le carré modulaire dénommé SQUASH [Sha08] qui se prête bien à des implantations très économiques. Pour une étude complète de l'implantation bas-coût des principaux algorithmes cryptographiques, on pourra se référer à [EKP+07].

Comme nous l'avons indiqué en introduction, la simplicité des opérations en jeu dans la définition du problème LPN (addition et multiplication sur GF(2)) en fait un problème difficile de choix pour concevoir des protocoles à bas-coût. C'est la principale raison qui a incité Juels et Weis à proposer le protocole HB$^+$. Avant d'entrer dans les détails de ce protocole, nous allons définir précisément les modèles de sécurité que nous utiliserons pour évaluer les différents protocoles d'authentification auxquels nous allons nous intéresser.

4.3 Notations et modèles d'attaques

Considérons un protocole d'authentification à clé secrète générique dépendant de paramètres $\texttt{param}$ et impliquant un prouveur $\mathcal{P}_K^{\texttt{param}}$ et un vérifieur $\mathcal{V}_K^{\texttt{param}}$ partageant une clé secrète $K \in \mathcal{K}$. L'exécution complète du protocole entre un algorithme $\mathcal{M}$ (pouvant être un prouveur légitime ou un adversaire) et le vérifieur sera notée $\langle \mathcal{M}, \mathcal{V}_K^{\texttt{param}} \rangle$, et nous noterons formellement $\langle \mathcal{M}, \mathcal{V}_K^{\texttt{param}} \rangle = 1$ si le vérifieur accepte l'authentification et $\langle \mathcal{M}, \mathcal{V}_K^{\texttt{param}} \rangle = 0$ si le vérifieur rejette l'authentification.

Lorsque les algorithmes du prouveur ou du vérifieur sont probabilistes, il peut arriver que le vérifieur rejette l'authentification même lorsqu'il interagit avec un prouveur légitime. On parle alors de *faux rejet*. Nous définirons la probabilité de faux rejet par :

$$P_{\mathrm{FR}} = \Pr\left[K \xleftarrow{\$} \mathcal{K} \,:\, \langle \mathcal{P}_K^{\texttt{param}}, \mathcal{V}_K^{\texttt{param}} \rangle = 0\right],$$

la probabilité étant prise sur la clé K et l'aléa interne du prouveur et du vérifieur.

Par ailleurs, un adversaire peut tenter de s'authentifier en envoyant des messages aléatoires. Nous appellerons un tel adversaire *adversaire trivial*, noté $\mathcal{A}_{\mathrm{triv}}$. Il existe une certaine probabilité pour que ces messages aléatoires mènent à une authentification réussie. On parlera alors de *fausse acceptation*, et nous définirons la probabilité de fausse acceptation par :

$$P_{\mathrm{FA}} = \Pr\left[K \xleftarrow{\$} \mathcal{K} \,:\, \langle \mathcal{A}_{\mathrm{triv}}, \mathcal{V}_K^{\texttt{param}} \rangle = 1\right],$$

la probabilité étant prise sur sur la clé K et l'aléa interne de l'adversaire trivial et du vérifieur.

Pour qu'un protocole d'authentification soit utilisable dans la pratique ces deux quantités P_{FA} et P_{FR} doivent être les plus petites possibles. Sauf mention explicite du contraire, nous supposerons dans toute cette partie que ces deux quantités sont des fonctions négligeables du paramètre de sécurité (en effet, pour les protocoles auxquels nous allons nous intéresser, il est toujours possible de sélectionner les paramètres de façon à ce que cette hypothèse soit vérifiée). Nous donnons maintenant une définition des différents types d'attaquants que nous allons considérer par la suite.

Attaquant passif. Afin de modéliser un adversaire passif qui ne fait qu'espionner les échanges entre un prouveur et un vérifieur légitimes (par exemple en interceptant la canal radio pour un système RFID), nous définissons un oracle $\texttt{trans}_K^{\texttt{param}}$ qui retourne la transcription des communications lors d'une authentification entre un prouveur et un vérifieur

légitimes utilisant une clé K. Durant une première phase, un attaquant passif peut faire un nombre polynomial de requêtes à l'oracle `trans`, c'est-à-dire obtenir la transcription d'un nombre polynomial d'authentifications entre un prouveur et un vérifieur légitimes. Puis lors d'une deuxième phase il tente de s'authentifier auprès du vérifieur. Formellement, l'avantage de l'attaquant est défini comme la probabilité additionnelle par rapport à un attaquant trivial répondant de façon complètement aléatoire qu'il s'authentifie correctement :

$$\mathrm{Adv}^{\mathrm{passif}}_{\mathcal{A}}(\mathtt{param}) \stackrel{\mathrm{def}}{=} \Pr\left[K \xleftarrow{\$} \mathcal{K}, \mathcal{A}^{\mathtt{trans}^{\mathtt{param}}_K}(1^k) \,:\, \langle \mathcal{A}, \mathcal{V}^{\mathtt{param}}_K \rangle = 1\right] - P_{\mathrm{FA}} \ ,$$

la probabilité étant prise sur la clé K et l'aléa interne de l'attaquant, de l'oracle $\mathtt{trans}^{\mathtt{param}}_K$ et du vérifieur. Remarquons que le fait que le résultat de l'authentification entre le prouveur et le vérifieur légitimes soit disponible ou non pour l'attaquant lors de la première phase ne change quasiment pas son avantage : en effet puisque la probabilité de faux rejet est négligeable, le résultat de l'authentification n'apprend rien à l'adversaire.

Attaquant actif. Un modèle d'attaque plus large considère que l'adversaire est capable d'interroger activement un prouveur légitime pendant un certain laps de temps afin d'obtenir suffisamment d'information sur sa clé secrète K pour ensuite s'authentifier auprès du vérifieur en se faisant passer pour le prouveur légitime. On parlera alors d'attaquant actif. Ce modèle est parfaitement réaliste pour un système RFID où un attaquant peut furtivement interroger une étiquette à distance. Plus formellement, pendant une première phase, l'attaquant peut librement interagir avec le prouveur un nombre polynomial de fois. Puis, lors d'une seconde phase, lors de laquelle il ne peut plus interagir avec $\mathcal{P}^{\mathtt{param}}_K$ (auquel cas il pourrait simplement effectuer une attaque par relai), il tente de s'authentifier auprès du vérifieur. Son avantage est défini par :

$$\mathrm{Adv}^{\mathrm{actif}}_{\mathcal{A}}(\mathtt{param}) \stackrel{\mathrm{def}}{=} \Pr\left[K \xleftarrow{\$} \mathcal{K}, \mathcal{A}^{\mathcal{P}^{\mathtt{param}}_K}(1^k) \,:\, \langle \mathcal{A}, \mathcal{V}^{\mathtt{param}}_K \rangle = 1\right] - P_{\mathrm{FA}} \ ,$$

la probabilité étant prise sur la clé K et l'aléa interne de l'attaquant, du prouveur et du vérifieur.

Attaquant man-in-the-middle. Il est important dans le modèle de l'attaquant actif que l'adversaire ne puisse pas interagir concomitamment avec le prouveur *et* le vérifieur. En effet dans ce cas on obtient un modèle bien plus fort, l'attaquant *man-in-the-middle* (littéralement « l'homme au milieu ») : l'attaquant peut librement modifier les échanges entre le prouveur et le vérifieur *et observer si l'authentification réussit ou échoue*. Remarquons que lorsque les messages du vérifieur ne dépendent pas de la clé secrète K (ce qui est le cas de tous les protocoles que nous étudierons), un attaquant actif peut tout à fait simuler par lui-même les messages du vérifieur, si bien que l'observation de la décision du vérifieur est l'élément critique qui distingue le modèle d'attaquant actif du modèle d'attaquant man-in-the-middle. Formellement, l'avantage d'un attaquant man-in-the-middle est défini par :

$$\mathrm{Adv}^{\mathrm{mitm}}_{\mathcal{A}}(\mathtt{param}) \stackrel{\mathrm{def}}{=} \Pr\left[K \xleftarrow{\$} \mathcal{K}, \mathcal{A}^{\mathcal{P}^{\mathtt{param}}_K, \mathcal{V}^{\mathtt{param}}_K}(1^k) \,:\, \langle \mathcal{A}, \mathcal{V}^{\mathtt{param}}_K \rangle = 1\right] - P_{\mathrm{FA}} \ ,$$

la probabilité étant prise sur la clé K et l'aléa interne de l'attaquant, du prouveur et du vérifieur.

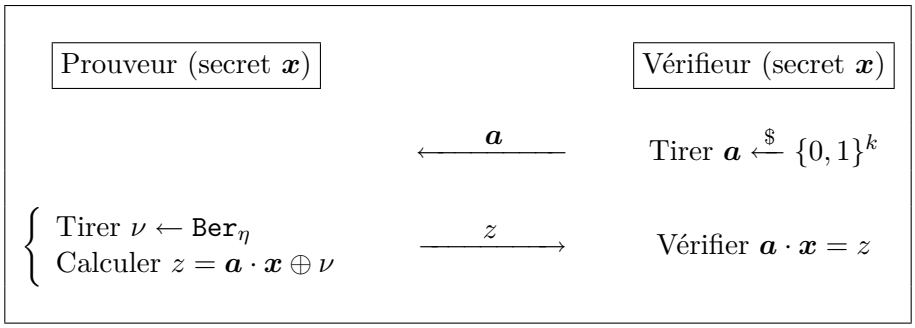

FIGURE 4.1 – Un tour du protocole HB. Il est répété r fois et l'authentification est réussie si le nombre de tours tels que $\boldsymbol{a} \cdot \boldsymbol{x} \neq z$ est inférieur ou égal à un seuil t.

Attaquant GRS. Nous aurons également besoin d'un modèle d'attaquant man-in-the-middle plus restreint que celui de la définition précédente, que nous appellerons modèle GRS, pour des raisons qui deviendront claires lorsque nous aborderons l'attaque de Gilbert, Robshaw et Sibert contre HB$^+$ à la section 4.5. Ce modèle est défini de façon très similaire au modèle man-in-the-middle général, à la différence près que l'attaquant n'est autorisé à modifier que les messages *du vérifieur vers le prouveur*. Autrement dit, l'attaquant peut librement interagir avec le prouveur, par contre les messages reçus par le vérifieur doivent être ceux d'un prouveur légitime. Contrairement au modèle actif, l'attaquant a connaissance du résultat de l'authentification. Formellement,

$$\mathrm{Adv}^{\mathrm{grs}}_{\mathcal{A}}(\mathtt{param}) \stackrel{\mathrm{def}}{=} \Pr\left[K \xleftarrow{\$} \mathcal{K}, \mathcal{A}^{\mathcal{P}_K^{\mathtt{param}}, \langle \cdot, \mathcal{V}_K^{\mathtt{param}} \rangle}(1^k) \,:\, \langle \mathcal{A}, \mathcal{V}_K^{\mathtt{param}} \rangle = 1\right] - P_{\mathrm{FA}} \ ,$$

la probabilité étant prise sur la clé K et l'aléa interne de l'attaquant, du prouveur et du vérifieur, et la notation $\langle \cdot, \mathcal{V}_K^{\mathtt{param}} \rangle$ signifiant que le résultat de l'authentification est connu de l'attaquant.

Dans tous les cas, on dira qu'un protocole est sûr dans un certain modèle d'attaque si tout adversaire efficace et respectant le modèle d'attaque n'a qu'une probabilité de succès négligeable contre le protocole. On peut facilement se convaincre que les modèles sont hiérarchisés selon :

$$\text{passif} \leq \text{actif} \leq \text{GRS} \leq \text{MITM} \ ,$$

la notation « A $\leq$ B » signifiant qu'un protocole sûr dans le modèle B est sûr dans le modèle A (ou de façon équivalente que l'existence d'un attaquant efficace et ayant une probabilité de succès non négligeable dans le modèle A implique l'existence d'un attaquant efficace et ayant une probabilité de succès non négligeable dans le modèle B).

4.4 L'ancêtre HB

Le protocole HB a été proposée par Hopper et Blum en 2001 [HB01]. Extrêmement simple, il était destiné à permettre à des humains de s'authentifier auprès d'ordinateurs de façon plus sûre que par un simple mot de passe.

Le prouveur et le vérifieur partagent un vecteur binaire secret $\boldsymbol{x}$ de longueur k. Pour authentifier le prouveur, le vérifieur répète r fois la procédure suivante (représentée sur la figure 4.1) : il tire un vecteur aléatoire $\boldsymbol{a}$ de longueur k et l'envoie au prouveur. Celui-ci

répond par un bit $z = \boldsymbol{a} \cdot \boldsymbol{x} \oplus \nu$, où $\nu \leftarrow \mathtt{Ber}_\eta$ est un bit de bruit valant 1 avec une probabilité $\eta \in]0, \frac{1}{2}[$ et 0 avec une probabilité $1 - \eta$. Au terme des r tours, le prouveur est authentifié par le vérifieur si le nombre de bits de réponse tels que $\boldsymbol{a} \cdot \boldsymbol{x} \neq z$ est inférieur ou égal à un seuil $t = ur$, où $u \in [\eta, \frac{1}{2}[$.

Un prouveur légitime est rejeté lorsqu'il introduit $(t + 1)$ erreurs ou plus, d'où une probabilité de faux rejet égale à :

$$P_{\mathrm{FR}} = \sum_{i=t+1}^{r} \binom{r}{i} \eta^i (1-\eta)^{r-i} \ .$$

Par ailleurs la probabilité de fausse acceptation d'un adversaire renvoyant des réponses parfaitement aléatoires est :

$$P_{\mathrm{FA}} = \frac{1}{2^r} \sum_{i=0}^{t} \binom{r}{i} \ .$$

Les bornes de Chernoff rappelées dans l'appendice B impliquent que ces deux quantités sont des fonctions négligeables de r (donc de k pour $r = \Theta(k)$), pour P_{FA} dès que η est constant et strictement inférieur à $\frac{1}{2}$, et pour P_{FR} dès que $u \geq (1+\gamma)\eta$, pour une constante $\gamma > 0$.

On peut montrer que HB est sûr contre les attaques passives. En effet, un attaquant qui tente de retrouver le secret $\boldsymbol{x}$ en espionnant simplement les échanges entre le prouveur et le vérifieur est exactement dans la position de résoudre une instance du problème LPN : il n'obtient que des produits scalaires bruités du vecteur secret $\boldsymbol{x}$ avec des vecteurs $\boldsymbol{a}$ aléatoires. En utilisant les notations introduites précédemment, on a $\mathtt{trans}_{\boldsymbol{x}}^{\eta} \simeq \Pi_{\boldsymbol{x},\eta}$. Au-delà de cette constatation évidente, on peut rendre l'argument de sécurité plus rigoureux par une réduction : un attaquant $\mathcal{A}$ polynomial possédant un avantage $\mathrm{Adv}_{\mathcal{A}}^{\mathrm{passif}}(k, r, \eta, u)$ notable peut être utilisé pour résoudre le problème LPN de paramètres (k, η) en temps polynomial. Une démonstration de ce résultat est donnée dans l'article de Juels et Weis dans le cas où l'attaquant ne tente de répondre qu'à un seul tour du protocole [JW05a]. Katz et Shin ont étendu la démonstration à un nombre de tours quelconque, ainsi qu'à la version parallèle de HB où les r tours sont effectués en une seule fois, le vérifieur envoyant un message $(\boldsymbol{a}_1, \ldots, \boldsymbol{a}_r)$ et le prouveur répondant par $(z_1, \ldots, z_r)$ où $z_i = \boldsymbol{a}_i \cdot \boldsymbol{x} \oplus \nu_i$, les bits de bruit ν_i étant indépendants [KS06a]. Leur démonstration n'était valable que pour $\eta < \frac{1}{4}$ et le résultat a été étendu à $\eta < \frac{1}{2}$ par Katz et Smith [KS06b].

Il existe cependant une attaque simple contre ce protocole lorsque l'adversaire est capable d'interagir activement avec le prouveur, c'est-à-dire dans le modèle actif. Elle consiste simplement à envoyer le même vecteur $\boldsymbol{a}$ au prouveur de façon répétée. En reprenant les notations du chapitre précédent, l'adversaire cherche à retrouver le secret $\boldsymbol{x}$ en questionnant l'oracle $\widetilde{\Pi}_{\boldsymbol{x},\eta}$. Le lemme 3.2 indique que ceci est possible avec une probabilité exponentiellement proche de 1 en temps et avec un nombre de requêtes $\mathcal{O}((1-2\eta)^{-2}k^2)$.

4.5 Le père HB$^+$

4.5.1 Description de HB$^+$

Partant du constat que les étiquettes radio-fréquence à bas-coût ont, tout comme les humains, des capacités de calcul limitées, Juels et Weis eurent l'idée d'adapter le protocole HB afin de le rendre utilisable pour l'authentification dans les systèmes RFID. Le problème majeur était de contrecarrer l'attaque active précédemment décrite, consistant à envoyer

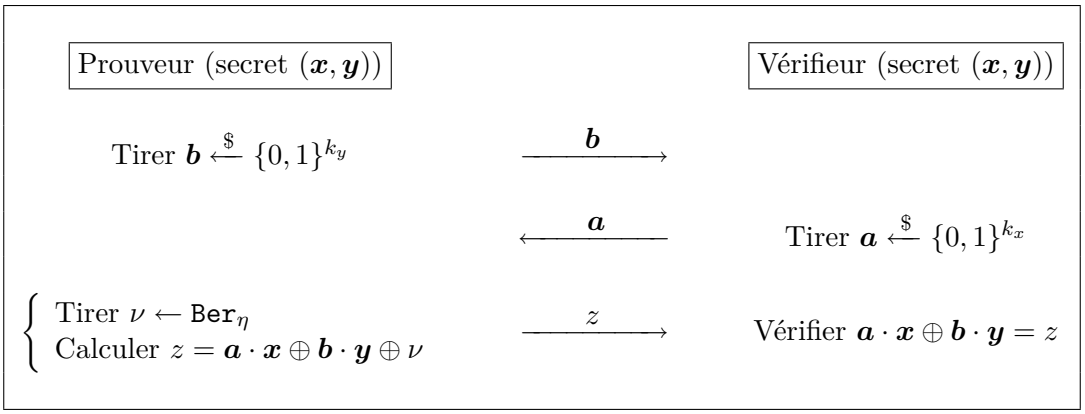

FIGURE 4.2 – Un tour du protocole HB$^+$. Il est répété r fois et l'authentification est réussie si le nombre de tours tels que $\boldsymbol{a} \cdot \boldsymbol{x} \oplus \boldsymbol{b} \cdot \boldsymbol{y} \neq z$ est inférieur ou égal à un seuil t.

plusieurs fois de suite le même vecteur $\boldsymbol{a}$ au prouveur afin d'obtenir le produit scalaire $\boldsymbol{a} \cdot \boldsymbol{x}$ avec certitude. Pour cela, Juels et Weis ont introduit une passe supplémentaire dite « de masquage » ou « d'engagement » destinée à masquer la valeur de $\boldsymbol{a} \cdot \boldsymbol{x} \oplus \nu$. Le prouveur et le vérifieur partagent désormais deux vecteurs secrets $\boldsymbol{x}$ et $\boldsymbol{y}$ de longueur respective k_x et k_y. L'authentification consiste en r tours se déroulant de la façon suivante (un tour est représenté sur la figure 4.2) : le prouveur tire un vecteur de masquage aléatoire $\boldsymbol{b}$ de longueur k_y et l'envoie au vérifieur ; celui-ci tire un vecteur challenge aléatoire $\boldsymbol{a}$ de longueur k_x et l'envoie au prouveur ; enfin le prouveur calcule le bit de réponse $z = \boldsymbol{a} \cdot \boldsymbol{x} \oplus \boldsymbol{b} \cdot \boldsymbol{y} \oplus \nu$, où $\nu \leftarrow \mathtt{Ber}_\eta$. Comme pour le protocole HB, au terme des r tours, le prouveur est authentifié par le vérifieur si le nombre de bits de réponse tels que $\boldsymbol{a} \cdot \boldsymbol{x} \oplus \boldsymbol{b} \cdot \boldsymbol{y} \neq z$ est inférieur ou égal à un seuil $t = ur$, où $u \in [\eta, \frac{1}{2}[$.

Remarquons que les probabilités de faux rejet et de fausse acceptation sont identiques à celles du protocole HB, soit :

$$P_{\mathrm{FR}} = \sum_{i=t+1}^{r} \binom{r}{i} \eta^i (1-\eta)^{r-i} \quad \text{et} \quad P_{\mathrm{FA}} = \frac{1}{2^r} \sum_{i=0}^{t} \binom{r}{i} \ .$$

HB$^+$ remplit parfaitement son rôle d'empêcher l'attaque active à laquelle était vulnérable HB. En effet, il est possible de prouver la sécurité de HB$^+$ contre les attaques actives. Comme pour HB, cela a été montré dans l'article original de Juels et Weis [JW05a] dans le cas où l'attaquant ne tente de répondre qu'à un seul tour du protocole, puis étendu au cas d'un nombre de tours quelconque ainsi qu'à la version parallèle (où la première passe de masquage se compose de r vecteurs $(\boldsymbol{b}_1, \ldots, \boldsymbol{b}_r)$, le challenge envoyé par le vérifieur de r vecteurs $(\boldsymbol{a}_1, \ldots, \boldsymbol{a}_r)$ et la réponse des r bits $(z_1, \ldots, z_r)$) par Katz et Shin [KS06a] pour $\eta < \frac{1}{4}$ puis par Katz et Smith [KS06b] pour le cas général $\eta < \frac{1}{2}$. Plus précisément, tout adversaire $\mathcal{A}$ polynomial possédant un avantage $\mathrm{Adv}_{\mathcal{A}}^{\mathrm{actif}}(k_x, k_y, r, \eta, u)$ notable peut être utilisé pour résoudre le problème LPN de paramètres (k_y, η) en temps polynomial. Remarquons que c'est uniquement la taille k_y du second secret qui intervient comme paramètre du problème LPN correspondant, k_x n'intervenant que dans la qualité de la réduction.

4.5.2 L'attaque de Gilbert-Robshaw-Sibert

Peu de temps après sa publication, Gilbert, Robshaw et Sibert publièrent une attaque contre le protocole HB$^+$ [GRS05]. Celle-ci n'entre évidemment pas dans la catégorie des

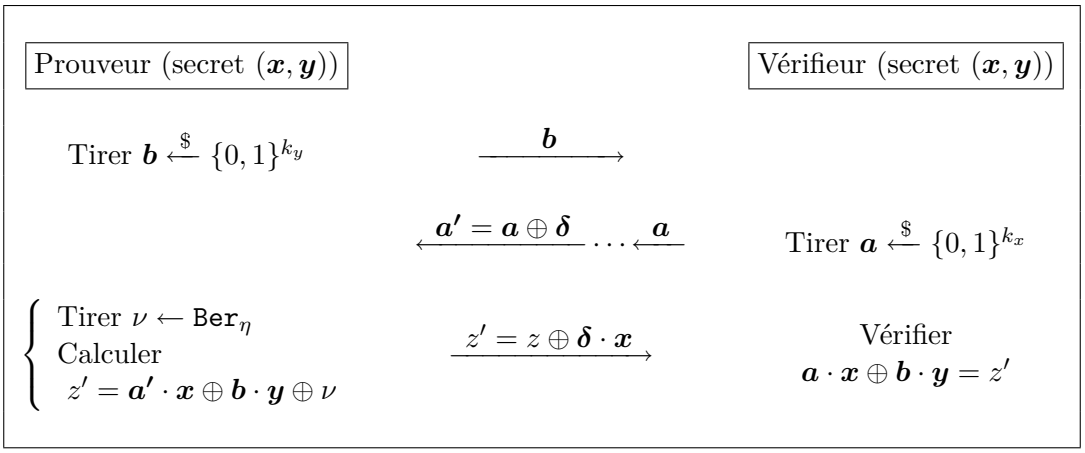

FIGURE 4.3 – L'attaque de Gilbert-Robshaw-Sibert sur un tour du protocole HB$^+$.

attaques actives contre lesquelles HB$^+$ est prouvé sûr. Il s'agit d'une attaque man-in-the-middle (plus précisément d'une attaque GRS, cf. section 4.3), lors de laquelle l'attaquant a la possibilité de modifier les échanges entre le prouveur et le vérifieur, et d'observer si l'authentification réussit ou non.

L'attaque consiste à modifier les vecteurs challenges $\boldsymbol{a}$ envoyés par le vérifieur. À chaque tour du protocole d'authentification, l'adversaire change le vecteur $\boldsymbol{a}$ en $\boldsymbol{a} \oplus \boldsymbol{\delta}$, pour un vecteur $\boldsymbol{\delta}$ arbitraire fixé. Le prouveur calcule donc le bit de réponse

$$z = \boldsymbol{a} \cdot \boldsymbol{x} \oplus \boldsymbol{b} \cdot \boldsymbol{y} \oplus \boldsymbol{\delta} \cdot \boldsymbol{x} \oplus \nu \ ,$$

alors que le vérifieur teste si $z = \boldsymbol{a} \cdot \boldsymbol{x} \oplus \boldsymbol{b} \cdot \boldsymbol{y}$. Le bit de bruit est donc systématiquement changé en $\nu \oplus \boldsymbol{\delta} \cdot \boldsymbol{x}$. Lorsque $\boldsymbol{\delta} \cdot \boldsymbol{x} = 0$, l'exécution du protocole n'est pas perturbée et l'authentification réussit avec probabilité $1 - P_{\mathrm{FR}}$ et échoue avec probabilité P_{FR} (rappelons que P_{FR} est une quantité négligeable en k). Au contraire, lorsque $\boldsymbol{\delta} \cdot \boldsymbol{x} = 1$, tous les bits de bruit sont inversés. L'authentification réussit donc avec une probabilité égale à :

$$\sum_{i=0}^{t} \binom{r}{i} (1-\eta)^i \eta^{r-i} = \sum_{j=r-t}^{r} \binom{r}{j} \eta^j (1-\eta)^{r-j} \leq P_{\mathrm{FR}} \ .$$

Chaque authentification révèle donc un bit du vecteur secret $\boldsymbol{x}$ avec une probabilité exponentiellement proche de 1. L'adversaire peut donc retrouver $\boldsymbol{x}$ en utilisant k vecteurs $\boldsymbol{\delta}$ indépendants. La probabilité de succès est supérieure à $(1 - P_{\mathrm{FR}})^k$, donc exponentiellement proche de 1.

Une fois le vecteur $\boldsymbol{x}$ correctement retrouvé, l'attaquant peut se faire passer pour le prouveur en envoyant des vecteurs de masquage $\boldsymbol{b} = \boldsymbol{0}$. Alternativement, il peut retrouver le vecteur $\boldsymbol{y}$ en se faisant passer pour un faux prouveur auprès du vérifieur : il envoie à chaque tour du protocole le même vecteur $\boldsymbol{b}$ et répond par le bit $\boldsymbol{a} \cdot \boldsymbol{x}$ à chaque challenge $\boldsymbol{a}$. Si l'authentification réussit alors $\boldsymbol{b} \cdot \boldsymbol{y} = 0$, sinon $\boldsymbol{b} \cdot \boldsymbol{y} = 1$. Remarquons que dans le premier scénario, l'attaque ne demande qu'à modifier les vecteurs challenges $\boldsymbol{a}$, donc uniquement les communications du vérifieur vers le prouveur. On voit donc qu'il s'agit d'une attaque dans le modèle GRS, ce qui constitue une justification *a posteriori* de la dénomination que nous avons introduite lors de la discussion des différents modèles d'attaque de la section 4.3.

Notons que l'attaque a une complexité *linéaire* en la taille de la clé. Elle est de ce fait extrêmement efficace, donc problématique. Comme le font remarquer Juels et Weis dans la version complète de leur article [JW05b], lors de l'attaque, l'authentification échoue environ une fois sur deux. Ceci peut éventuellement être détecté par le système qui en déduit qu'une attaque est en cours et prend alors les mesures appropriées (révocation du prouveur, etc.).

4.5.3 Vers des variantes résistantes aux attaques man-in-the-middle ?

Suite à la publication de l'attaque GRS, de nombreux cryptographes ont tenté d'améliorer HB$^+$ et d'obtenir un protocole d'une efficacité similaire à HB$^+$ mais résistant à cette attaque. En particulier, trois protocoles ont été proposés par différentes équipes : HB^{++} [BCD06], HB* [DK07], et HB-MP [MP07]. Mais nous allons voir dans les sections suivantes qu'aucun de ces protocoles ne résiste à des attaques de type GRS.

Avant de continuer plus avant dans cette partie, il convient de se demander si les attaques man-in-the-middle représentent une réelle menace, et s'il est utile d'essayer de rendre HB$^+$ résistant à ces attaques. Tout d'abord, bien que HB$^+$ ait été pensé par ses auteurs comme destiné aux systèmes RFID à bas-coût, il n'est pas interdit d'envisager son utilisation dans d'autres contextes, par exemple pour l'authentification à distance sur Internet. Personne ne discutera dans ce cas le fait que les attaques man-in-the-middle puissent réellement être mises en œuvre.

La question est nettement plus sujette à controverse en ce qui concerne le cas des systèmes RFID. Un scénario possible de mise en œuvre de l'attaque GRS consiste à créer un faux lecteur et une fausse étiquette, et à faire interagir le faux lecteur avec l'étiquette légitime et la fausse étiquette avec le lecteur légitime. Comme le font remarquer les auteurs de l'attaque [GRS05], il n'est pas nécessaire que le faux lecteur et la fausse étiquette se trouvent dans un même lieu physique, mais seulement que l'on puisse relayer leurs communications en temps réel. Même si ce scénario ne semble pas le plus facile à mettre en place, des attaques très similaires (mais plus rudimentaires, car il s'agit de simples attaques relai) ont été démontrées par Hancke [Han05], Kfir et Wool [KW05] et Kirschenbaum et Wool [KW06]. Mais l'attaquant peut aussi se contenter de causer des perturbations contrôlées aux communications radio-fréquences du lecteur vers l'étiquette. La plupart des experts de la physique des systèmes RFID considèrent qu'une telle manipulation des échanges est impossible à l'heure actuelle, mais n'est pas à exclure d'un point de vue *théorique*.

Nous ne tenterons pas d'apporter une réponse définitive à cette question, et nous nous contenterons de souligner que le travail des cryptographes consiste à être paranoïaques, et à anticiper les attaques futures, impossibles avec les technologies actuelles mais qui pourraient devenir réalisables avec les progrès de la technique.

4.6 Cryptanalyse de la variante HB^{++}

4.6.1 Description de HB^{++}

HB^{++} a été proposé par Bringer, Chabanne et Dottax [BCD06] dans le but de contrecarrer l'attaque GRS tout en conservant la simplicité de HB$^+$. Ce protocole est représenté sur la figure 4.4. Il est constitué de deux phases. Tout d'abord, quatre vecteurs secrets $\boldsymbol{x}$, $\boldsymbol{x}'$, $\boldsymbol{y}$ et $\boldsymbol{y}'$ de longueur k sont dérivés d'une clé maîtresse $\boldsymbol{Z}$ à l'aide d'une fonction de

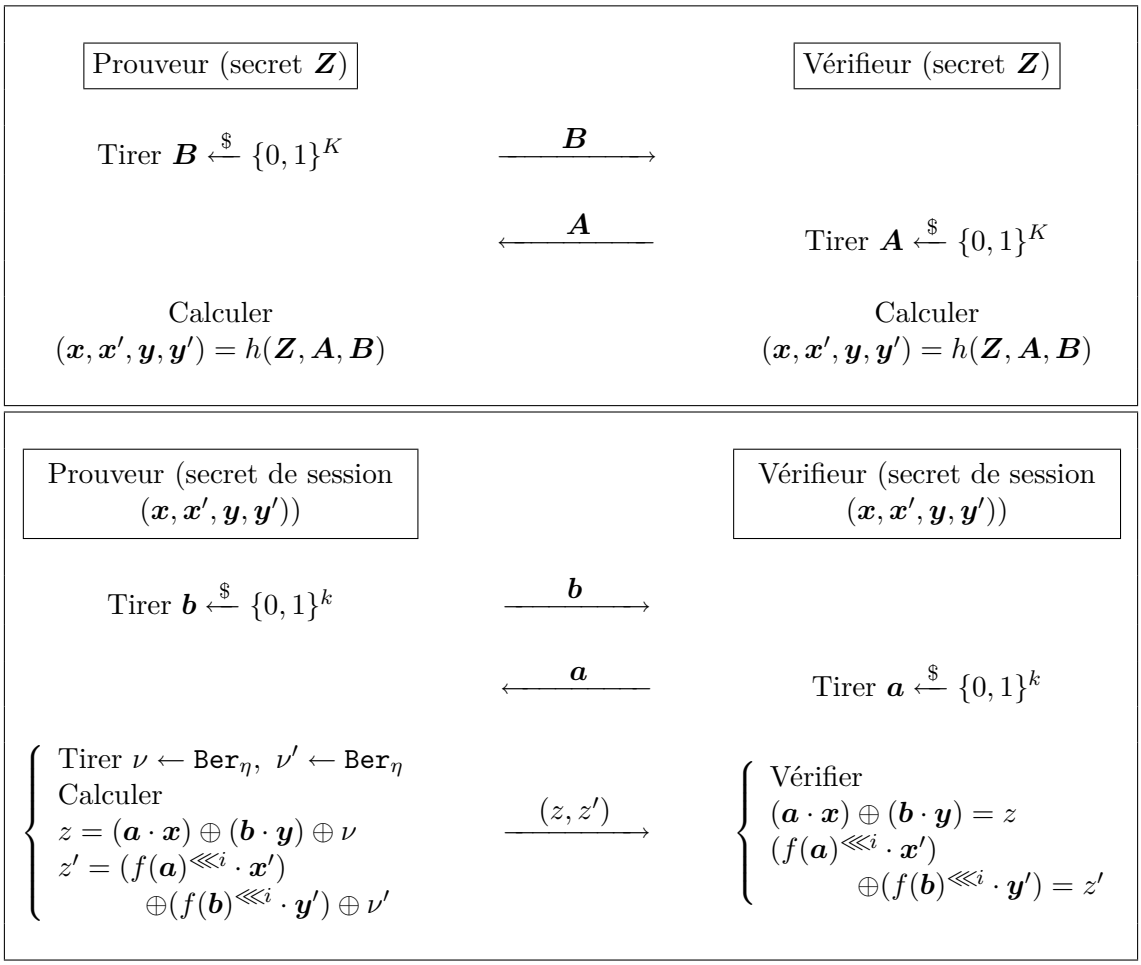

FIGURE 4.4 – Le protocole HB^{++}. La figure du haut représente la phase de dérivation des clés de session $(\boldsymbol{x}, \boldsymbol{x}', \boldsymbol{y}, \boldsymbol{y}')$ à partir de la clé maîtresse $\boldsymbol{Z}$. La figure du bas représente un tour de la phase d'authentification proprement dite. Il est répété r fois et l'authentification est réussie si les nombres d'erreurs sur les bits z et z' sont tous deux inférieurs ou égaux à un seuil t.

hachage universelle h et de deux vecteurs $\boldsymbol{A}$ et $\boldsymbol{B}$ échangés entre le prouveur et le vérifieur selon la formule :

$$(\boldsymbol{x}, \boldsymbol{x}', \boldsymbol{y}, \boldsymbol{y}') = h(\boldsymbol{Z}, \boldsymbol{A}, \boldsymbol{B}) \ .$$

Ces secrets dérivés peuvent être vus comme des clés de session. Nous reviendrons plus en détail sur cette partie du protocole par la suite.

Un tour du protocole d'authentification proprement dit, consiste, comme pour HB$^+$, en l'échange d'un vecteur de masquage $\boldsymbol{b}$ et d'un vecteur challenge $\boldsymbol{a}$, tous deux de longueur k, entre le prouveur et le vérifieur, mais à la différence de HB$^+$, deux bits de réponse sont calculés par le prouveur. Le premier vaut $z = \boldsymbol{a} \cdot \boldsymbol{x} \oplus \boldsymbol{b} \cdot \boldsymbol{y} \oplus \nu$ comme dans HB$^+$. Le second vaut

$$z' = (f(\boldsymbol{a})^{\lll i} \cdot \boldsymbol{x}') \oplus (f(\boldsymbol{b})^{\lll i} \cdot \boldsymbol{y}') \oplus \nu' \ ,$$

où f est une permutation sur $\{0,1\}^k$ (dont la description exacte sera discutée plus loin), et $\boldsymbol{c}^{\lll i}$ dénote le vecteur obtenu en appliquant aux bits de $\boldsymbol{c}$ une rotation circulaire de i bits vers la gauche, où i est le numéro du tour (les tours étant numérotés de 0 à $r-1$). Les deux bits de bruit ν et ν' sont tous deux tirés indépendamment selon le paramètre de bruit η. Le prouveur est authentifié si le nombre d'erreurs sur le bit de réponse z et le nombre d'erreurs sur z' sont tous deux inférieurs ou égaux à un seuil $t = ur$, avec $u \in [\eta, \frac{1}{2}[$. Comme on peut le constater, HB^{++} consiste en deux protocoles HB$^+$ parallèles avec des secrets indépendants mais des paires de challenges corrélés.

Les probabilités de faux rejet et de fausse acceptation deviennent :

$$P_{\mathrm{FR}} = 1 - \left(\sum_{i=0}^{t} \binom{r}{i} \eta^i (1-\eta)^{r-i} \right)^2 \quad \text{et} \quad P_{\mathrm{FA}} = \left(\frac{1}{2^r} \sum_{i=0}^{t} \binom{r}{i} \right)^2 .$$

La permutation f est choisie de façon à satisfaire de bonnes propriétés différentielles, c'est-à-dire à minimiser la quantité :

$$\Delta_f = \max_{\boldsymbol{v} \neq \boldsymbol{0}, \boldsymbol{w}} \left(\#\{\boldsymbol{a} \in \{0,1\}^k \mid f(\boldsymbol{a} \oplus \boldsymbol{v}) \oplus f(\boldsymbol{a}) = \boldsymbol{w}\} \right) \ .$$

Les auteurs de [BCD06] montrent alors de façon heuristique qu'en raison de cette propriété de la permutation f, une transposition directe de l'attaque GRS sur HB^{++} est inopérante. En effet, ignorons un instant les rotations effectuées pour le calcul du bit de réponse z' et considérons un attaquant essayant de déduire de l'information sur $\boldsymbol{x}$ et $\boldsymbol{x}'$ en ajoutant un vecteur fixé $\boldsymbol{\delta}$ aux challenges $\boldsymbol{a}$ du vérifieur. Le bit d'erreur sur z'_i est alors modifié en $\nu'_i \oplus (f(\boldsymbol{a}_i) \oplus f(\boldsymbol{a}_i \oplus \boldsymbol{\delta})) \cdot \boldsymbol{x}'$. Pour déduire une information significative de la décision du vérifieur, il est nécessaire que $(f(\boldsymbol{a}_i) \oplus f(\boldsymbol{a}_i \oplus \boldsymbol{\delta})) = \boldsymbol{\delta}'$, pour un $\boldsymbol{\delta}'$ fixé, pour un nombre de tours proches de r. Or par définition de Δ_f, la probabilité que cela arrive est environ :

$$p = \left(\frac{\Delta_f}{2^k} \right)^r \ ,$$

donc très faible. Ainsi, les auteurs de [BCD06] concluent que lorsque p est négligeable, HB^{++} est sûr contre les attaques de type GRS.

Remarquons que les rotations sont nécessaires pour déjouer une attaque due à Wagner sur une version préliminaire de HB^{++} où les rotations étaient omises, comme expliqué dans l'article original [BCD06]. Cette attaque consiste à utiliser la structure particulière de la permutation f suggérée par les auteurs de [BCD06] : afin d'en permettre une implantation économique, Bringer *et al.* proposent de construire f comme l'application parallèle d'une

permutation plus petite. Ainsi, si $k = 80$, et si g désigne une permutation de $\{0,1\}^5$, f est définie comme la juxtaposition de 16 fonctions g : pour $\boldsymbol{a} = (a_1, \ldots, a_{16})$, où les a_i sont des blocs de 5 bits, on a :

$$f(\boldsymbol{a}) = (g(a_1), \ldots, g(a_{16})) \ .$$

Dans ce cas, il est possible de retrouver les secrets $\boldsymbol{x}$ et $\boldsymbol{x'}$ par blocs de 5 bits successifs grâce à une recherche exhaustive sur la valeur de ces blocs. Pour cela, on fait une hypothèse sur la valeur d'un blocs de 5 bits de $\boldsymbol{x}$ ainsi que de $\boldsymbol{x'}$. Puis on choisit un vecteur $\boldsymbol{\delta}$ dont le support est réduit au bloc de 5 bits considéré que l'on ajoute aux challenges $\boldsymbol{a_i}$ du vérifieur. Il résulte de la forme de f que le vecteur $\boldsymbol{\delta}'_i = f(\boldsymbol{a}_i \oplus \boldsymbol{\delta}) \oplus f(\boldsymbol{a}_i)$ a lui aussi un support réduit au bloc de 5 bits considéré. On corrige alors les réponses z et z' du prouveur en $z \oplus \boldsymbol{\delta} \cdot \boldsymbol{x}$ et $z' \oplus \boldsymbol{\delta}'_i \cdot \boldsymbol{x'}$ en fonction des hypothèses faites sur la valeur du bloc de $\boldsymbol{x}$ et $\boldsymbol{x'}$. Si le vérifieur accepte l'authentification, avec une forte probabilité l'hypothèse était la bonne, sinon elle était fausse. En répétant la procédure pour chaque bloc de 5 bits, on retrouve la totalité des vecteurs $\boldsymbol{x}$ et $\boldsymbol{x'}$. Remarquons que ce raisonnement ignore la phase de renouvellement des secrets de session.

Note sur la variante de Piramuthu. Piramuthu donne dans [Pir06a] une cryptanalyse de HB^{++}, cependant celle-ci repose sur l'hypothèse qu'il existe un moyen de forcer le prouveur à utiliser plusieurs fois le même vecteur $\boldsymbol{b}$, ce qui est une supposition extrêmement forte, injustifiable dans la plupart des cas. Piramuthu présente également une variante de HB^{++} mais dont les spécifications et les propriétés sont peu claires. Nous n'étudierons donc pas cette variante dans ce mémoire.

4.6.2 Attaque de HB^{++} sans le renouvellement des secrets

Nous allons voir dans un premier temps comment attaquer le protocole HB^{++} sans la phase de renouvellement de secret : les mêmes clés de session $(\boldsymbol{x}, \boldsymbol{x'}, \boldsymbol{y}, \boldsymbol{y'})$ sont utilisées à chaque authentification. Nous verrons dans la section suivante comment étendre l'attaque au cas général.

Les arguments de sécurité en faveur de HB^{++} partent du principe que *tous* les tours de l'authentification sont perturbés par l'attaquant. Dans ce cas il est vrai que le nombre d'erreurs sur le bit de réponse z' sera toujours supérieur au seuil avec une forte probabilité, si bien que le vérifieur rejettera toujours l'authentification et sa réponse n'apprendra rien à l'attaquant. Nous allons voir cependant qu'il est possible d'obtenir de l'information en ne perturbant qu'une fraction des tours de façon à garder un nombre d'erreurs sur le bit z' inférieur au seuil t.

Soit donc $s < r$ un entier représentant le nombre de tours d'authentification perturbés : l'attaquant change le challenge $\boldsymbol{a}_i$ du tour i en $\boldsymbol{a}'_i = \boldsymbol{a}_i \oplus \boldsymbol{\delta}$ pour les tours $0 \leq i \leq s-1$, et laisse les tours $s \leq i \leq r-1$ se dérouler inchangés. Notons respectivement σ_i et σ'_i les bits d'erreur sur les bits de réponse z et z' au tour i. On a alors, pour $0 \leq i \leq s-1$,

$$\begin{cases} \sigma_i = \nu_i \oplus \boldsymbol{\delta} \cdot \boldsymbol{x} \\ \sigma'_i = \nu'_i \oplus \boldsymbol{\delta}'_i \cdot \boldsymbol{x'} \quad \text{avec} \quad \boldsymbol{\delta}'_i = (f(\boldsymbol{a}_i \oplus \boldsymbol{\delta}) \oplus f(\boldsymbol{a}_i))^{\lll i} \end{cases} ,$$

tandis que pour $s \leq i \leq r-1$ on a simplement $\sigma_i = \nu_i$ et $\sigma'_i = \nu'_i$.

Soit N (respectivement N') le nombre total d'erreurs sur le bit de réponse z (respectivement z') à la fin de l'attaque. Comme nous l'avons vu dans la section 4.6.1, la permutation

f est choisie pour avoir de bonnes propriétés différentielles : pour deux vecteurs $\boldsymbol{\delta}$, $\boldsymbol{\delta}'$ fixés, on a :

$$\Pr_{\boldsymbol{a}}[f(\boldsymbol{a} \oplus \boldsymbol{\delta}) \oplus f(\boldsymbol{a}) = \boldsymbol{\delta}'] \leq \frac{\Delta_f}{2^k} \ ,$$

quantité très faible et proche de 2^{-k}. Ainsi, pour $0 \leq i \leq s-1$, les vecteurs $\boldsymbol{a}_i$ étant uniformément distribués, les vecteurs $\boldsymbol{\delta}'_i$ sont presque uniformément distribués. En supposant $\boldsymbol{x}' \neq \mathbf{0}$, on en déduit que les bits $\boldsymbol{\delta}'_i \cdot \boldsymbol{x}'$ sont presque uniformément distribués, si bien que les bits de bruit σ'_i le sont également. Nous pouvons donc faire l'approximation que N' est la somme de s variables indépendantes prenant la valeur 0 ou 1 avec probabilité $\frac{1}{2}$ et de $r-s$ variables indépendantes prenant la valeur 0 avec probabilité $1-\eta$ et 1 avec probabilité η, si bien que :

$$\Pr[N' = l] \simeq \sum_{i=0}^{l} \binom{s}{i} \frac{1}{2^s} \binom{r-s}{l-i} \eta^{l-i} (1-\eta)^{r-s-l+i} \ .$$

La valeur moyenne de N' vaut $\mu' = \frac{s}{2} + \eta(r-s) = \frac{1}{2}(1-2\eta)s + \eta r$.

Contrairement à celle de N', la distribution de N dépend de $\boldsymbol{\delta}$. Lorsque $\boldsymbol{\delta} \cdot \boldsymbol{x} = 0$, les bits de réponse z_i sont inchangés et N suit une loi binomiale de paramètres r et η :

$$\Pr[N = l \,|\, \boldsymbol{\delta} \cdot \boldsymbol{x} = 0] = \binom{r}{l} \eta^l (1-\eta)^{r-l} \ .$$

La valeur moyenne de N dans ce cas vaut $\mu_0 = \eta r < t$. Lorsque $\boldsymbol{\delta} \cdot \boldsymbol{x} = 1$, les s premiers bits de réponse z_i sont corrects avec probabilité η et incorrects avec probabilité $1-\eta$, alors que les $r-s$ derniers bits de réponse sont inchangés. Dans ce cas N est la somme de s variables indépendantes prenant la valeur 0 avec probabilité η et 1 avec probabilité $1-\eta$ et de $r-s$ variables indépendantes prenant la valeur 0 avec probabilité $1-\eta$ et 1 avec probabilité η, si bien que :

$$\Pr[N = l \,|\, \boldsymbol{\delta} \cdot \boldsymbol{x} = 1] = \sum_{i=0}^{l} \binom{s}{i} (1-\eta)^i \eta^{s-i} \binom{r-s}{l-i} \eta^{l-i} (1-\eta)^{r-s-l+i} \ .$$

La valeur moyenne de N dans ce cas vaut $\mu_1 = (1-\eta)s + \eta(r-s) = (1-2\eta)s + \eta r$.

Par conséquent, si l'on choisit s tel que $\mu' < t$ et $\mu_1 > t$, le nombre d'erreurs sur z' sera sous le seuil de rejet avec une forte probabilité, si bien que la décision du vérifieur indiquera si le nombre d'erreurs sur z est inférieur ou supérieur à t, ce qui indiquera alors si $\boldsymbol{\delta} \cdot \boldsymbol{x} = 0$ ou $\boldsymbol{\delta} \cdot \boldsymbol{x} = 1$. D'après les expressions de μ' et μ_1, pour que l'attaque soit valide, s doit vérifier :

$$\frac{t - \eta r}{1 - 2\eta} < s < 2 \cdot \frac{t - \eta r}{1 - 2\eta} \ .$$

Pour éviter que la probabilité de faux rejet soit trop élevée, l'écart entre le seuil t et le nombre moyen d'erreurs ηr doit être suffisamment grand, si bien qu'en général l'existence de telles valeurs de s est assurée.

La stratégie d'attaque est donc la suivante : à l'issue de l'authentification perturbée, si le vérifieur accepte l'attaquant en déduit que $\boldsymbol{\delta} \cdot \boldsymbol{x} = 0$, si elle échoue il en déduit que $\boldsymbol{\delta} \cdot \boldsymbol{x} = 1$. Calculons les probabilités d'erreur (sur $\boldsymbol{\delta}$ et l'aléa du protocole lui-même) de cette stratégie. Nous noterons :

- P_{acc} la probabilité que le vérifieur accepte l'authentification perturbée
- $P_{\text{rej}} = 1 - P_{\text{acc}}$ la probabilité que le vérifieur rejette l'authentification perturbée

- $P_{\text{er|acc}}$ la probabilité d'erreur de l'attaquant lors de sa décision quant à la valeur de $\boldsymbol{\delta} \cdot \boldsymbol{x}$ sachant que le vérifieur accepte l'authentification
- $P_{\text{er|rej}}$ la probabilité d'erreur de l'attaquant lors de sa décision quant à la valeur de $\boldsymbol{\delta} \cdot \boldsymbol{x}$ sachant que le vérifieur rejette l'authentification
- P_{er} la probabilité d'erreur totale de l'attaquant lors de sa décision quant à la valeur de $\boldsymbol{\delta} \cdot \boldsymbol{x}$.

Afin d'obtenir les expressions de ces probabilités, nous avons besoin des trois probabilités suivantes :

$$\begin{aligned}
\Pr\left[N' \leq t\right] &= \sum_{i=0}^{t} \sum_{j=0}^{t-i} \binom{s}{i} \frac{1}{2^s} \binom{r-s}{j} \eta^j (1-\eta)^{r-s-j} \\
\Pr\left[N \leq t \,|\, \boldsymbol{\delta} \cdot \boldsymbol{x} = 0\right] &= \sum_{i=0}^{t} \binom{r}{i} \eta^i (1-\eta)^{r-i} \\
\Pr\left[N \leq t \,|\, \boldsymbol{\delta} \cdot \boldsymbol{x} = 1\right] &= \sum_{i=0}^{t} \sum_{j=0}^{t-i} \binom{s}{i} (1-\eta)^i \eta^{s-i} \binom{r-s}{j} \eta^j (1-\eta)^{r-s-j} \quad .
\end{aligned}$$

La probabilité d'acceptation est alors donnée par :

$$\begin{aligned}
P_{\text{acc}} &= \frac{1}{2} \left(\Pr\left[\text{accepte} \,|\, \boldsymbol{\delta} \cdot \boldsymbol{x} = 0\right] + \Pr\left[\text{accepte} \,|\, \boldsymbol{\delta} \cdot \boldsymbol{x} = 1\right]\right) \\
&= \frac{1}{2} \left(\Pr\left[(N \leq t) \wedge (N' \leq t) \,|\, \boldsymbol{\delta} \cdot \boldsymbol{x} = 0\right] + \Pr\left[(N \leq t) \wedge (N' \leq t) \,|\, \boldsymbol{\delta} \cdot \boldsymbol{x} = 1\right]\right) \\
&= \frac{1}{2} \Pr\left[N' \leq t\right] \left(\Pr\left[(N \leq t) \,|\, \boldsymbol{\delta} \cdot \boldsymbol{x} = 0\right] + \Pr\left[(N \leq t) \,|\, \boldsymbol{\delta} \cdot \boldsymbol{x} = 1\right]\right) \quad .
\end{aligned}$$

La probabilité d'erreur lors d'une acceptation est donnée par :

$$\begin{aligned}
P_{\text{er|acc}} &= \Pr\left[\boldsymbol{\delta} \cdot \boldsymbol{x} = 1 \,|\, \text{accepte}\right] \\
&= \frac{\Pr\left[\text{accepte} \,|\, \boldsymbol{\delta} \cdot \boldsymbol{x} = 1\right]}{2P_{\text{acc}}} \\
&= \frac{\Pr\left[N' \leq t\right] \cdot \Pr\left[(N \leq t) \,|\, \boldsymbol{\delta} \cdot \boldsymbol{x} = 1\right]}{2P_{\text{acc}}} \quad .
\end{aligned}$$

La probabilité d'erreur lors d'un rejet est donnée par :

$$\begin{aligned}
P_{\text{er|rej}} &= \Pr\left[\boldsymbol{\delta} \cdot \boldsymbol{x} = 0 \,|\, \text{rejette}\right] \\
&= \frac{\Pr\left[\text{rejette} \,|\, \boldsymbol{\delta} \cdot \boldsymbol{x} = 0\right]}{2(1 - P_{\text{acc}})} \\
&= \frac{1 - \Pr\left[N' \leq t\right] \cdot \Pr\left[(N \leq t) \,|\, \boldsymbol{\delta} \cdot \boldsymbol{x} = 0\right]}{2(1 - P_{\text{acc}})} \quad .
\end{aligned}$$

Enfin, la probabilité d'erreur totale est donnée par :

$$\begin{aligned}
P_{\text{er}} &= P_{\text{er|acc}} \cdot P_{\text{acc}} + P_{\text{er|rej}} \cdot P_{\text{rej}} \\
&= \frac{1}{2} \left(1 - \Pr\left[N' \leq t\right] \left(\Pr\left[(N \leq t) \,|\, \boldsymbol{\delta} \cdot \boldsymbol{x} = 1\right] - \Pr\left[(N \leq t) \,|\, \boldsymbol{\delta} \cdot \boldsymbol{x} = 0\right]\right)\right) \quad .
\end{aligned}$$

Munis de ces expressions, nous pouvons calculer ces différentes probabilités pour diverses valeurs des paramètres en fonction du nombre s de tours perturbés, pour s variant de $s_{\min} = \left\lceil \frac{t-\eta r}{1-2\eta} \right\rceil$ à $s_{\max} = \left\lfloor 2 \cdot \frac{t-\eta r}{1-2\eta} \right\rfloor$. La probabilité d'erreur P_{er} passe par un minimum

r	η	t	$s_{\min}$	$s_{\max}$	s_{opt}	P_{acc}	$P_{\text{er}\|\text{acc}}$	$P_{\text{er}\|\text{rej}}$	P_{er}
80	0,25	30	21	40	31	0,41	0,09	0,21	0,16
160	0,25	60	41	80	58	0,45	0,06	0,14	0,10
256	0,25	96	65	128	90	0,47	0,03	0,08	0,06

TABLE 4.1 – Valeur du nombre optimal de tours perturbés et des probabilités d'erreur correspondantes lors de l'attaque contre HB^{++} consistant à faire une hypothèse quelle que soit la décision du vérifieur.

r	η	t	$s_{\max}$	P_{acc}^{-1}	$P_{\text{er}\|\text{acc}}$
80	0,25	30	40	3,61	0,07
160	0,25	60	80	3,73	2.10^{-4}
256	0,25	96	128	3,78	$2,6.10^{-6}$

TABLE 4.2 – Valeur de la probabilité d'erreur lors de l'attaque contre HB^{++} consistant à ne considérer que les authentifications acceptées par le vérifieur.

pour une certaine valeur de $s = s_{\text{opt}}$. Divers exemples de paramètres sont donnés dans la table 4.1. Il apparait alors qu'il peut être relativement incertain de faire l'hypothèse que $\boldsymbol{\delta} \cdot \boldsymbol{x} = 1$ lorsque le vérifieur rejette l'authentification. En effet, cela peut souvent arriver en raison d'un nombre d'erreurs supérieur à t sur z', alors que $\boldsymbol{\delta} \cdot \boldsymbol{x} = 0$. Une meilleure stratégie consiste à ne considérer que les authentifications acceptées. La probabilité d'erreur totale se réduit alors à $P_{\text{er}|\text{acc}}$, qui est une fonction décroissante de s, ce qui implique qu'il vaut mieux dans ce cas utiliser $s = s_{\max}$. La contrepartie est qu'un plus grand nombre d'authentifications sont requises (précisément P_{acc}^{-1} fois plus). Cependant ceci ne représente pas un facteur significatif, puisque lorsque $s = s_{\max}$, on a $\mu' \simeq t$, si bien que $\Pr[N' \leq t] \simeq \frac{1}{2}$, et par conséquent $P_{\text{acc}} \simeq \Pr[N' \leq t] \Pr[\boldsymbol{\delta} \cdot \boldsymbol{x} = 0] \simeq \frac{1}{4}$. Des exemples de valeur concrètes pour cette stratégie sont données dans la table 4.2. On voit que l'on obtient une probabilité d'erreur très faible, inférieure en général à quelques pour-cent. Par la suite nous noterons η_{eff} cette probabilité d'erreur.

Ainsi, l'attaquant a la possibilité d'obtenir la valeur du produit scalaire $\boldsymbol{\delta} \cdot \boldsymbol{x}$ pour des vecteurs $\boldsymbol{\delta}$ choisis, avec une probabilité d'erreur très faible, et peut ainsi retrouver le vecteur secret $\boldsymbol{x}$ avec une faible nombre d'authentifications. Ainsi, pour les paramètres $r = 160$, $\eta = 0,25$, $t = 60$, le niveau de bruit est $\eta_{\text{eff}} = 2.10^{-4}$, si bien que pour une longueur k du vecteur secret $\boldsymbol{x}$ typique de quelques centaines de bits, on pourra obtenir k équations sans erreurs avec une forte probabilité. Une fois le vecteur $\boldsymbol{x}$ connu, $\boldsymbol{x}'$ peut être retrouvé en ajoutant au i-ième challenge $\boldsymbol{a}_i$ un vecteur $\boldsymbol{\delta}_i$ tel que $\boldsymbol{\delta}_i \cdot \boldsymbol{x} = 0$ et $\boldsymbol{\delta}'_i = (f(\boldsymbol{a}_i \oplus \boldsymbol{\delta}_i) \oplus f(\boldsymbol{a}_i))^{\lll i}$ est constant, ce qui donnera des équations linéaires approchées sur $\boldsymbol{x}$'.

Cette attaque suppose que les authentifications successives soient réalisées avec les mêmes secrets de session $\boldsymbol{x}$, $\boldsymbol{x}'$. Mais nous allons voir que l'on peut l'étendre au cas où la phase de renouvellement de secret est réalisée avant chaque authentification.

4.6.3 Attaque de HB^{++} avec le renouvellement des secrets

Considérons maintenant la situation où les secrets $\boldsymbol{x}$, $\boldsymbol{x}'$, $\boldsymbol{y}$ et $\boldsymbol{y}'$ sont renouvelés avant chaque authentification. Nous nous focaliserons dans cette section sur l'instanciation

concrète proposée par les auteurs [BCD06]. Pour cela, il nous faut entrer dans les détails de la fonction de hachage universelle h utilisée pour la dérivation des $4 \times 80 = 320$ bits de clés de session $(\boldsymbol{x}, \boldsymbol{x}', \boldsymbol{y}, \boldsymbol{y}')$ à partir d'une clé maitresse $\boldsymbol{Z}$ de $K = 768$ bits. Cette fonction est dérivée de la famille de fonctions de hachage WH, une variante de la famille de fonctions de hachage NH sur laquelle repose la code d'authentification de message UMAC [BHK+99]. WH a été proposé par Kaps *et al.* [KYS05].

La fonction h utilise une instance de WH définie comme suit : étant donnés deux mots de 160 bits $\boldsymbol{K} = (K_1, \ldots, K_n) \in (\mathbb{F}_{2^{16}})^n$, et $\boldsymbol{M} = (M_1, \ldots, M_n) \in (\mathbb{F}_{2^{16}})^n$, où $n = 10$, le mot de 16 bits $\mathrm{WH}_{\boldsymbol{K}}(\boldsymbol{M})$ est défini par :

$$\mathrm{WH}_{\boldsymbol{K}}(\boldsymbol{M}) = \sum_{i=1}^{n/2} (M_{2i-1} + K_{2i-1}) \cdot (M_{2i} + K_{2i}) \cdot c_i \ ,$$

où les c_i sont des constantes de $\mathbb{F}_{2^{16}}$ définies dans [KYS05].

La fonction h résulte de $l = 20$ invocations de cette instance de WH, selon une construction d'une famille de fonctions de hachage de plus grande taille de clé et de sortie, dénommée WH^T, proposée dans [KYS05]. Étant donné $\boldsymbol{Z} = (Z_1, \ldots, Z_{n+2(l-1)}) \in (\mathbb{F}_{2^{16}})^{n+2(l-1)} = (\mathbb{F}_{2^{16}})^{48}$, et $\boldsymbol{M} = (M_1, \ldots, M_n) \in (\mathbb{F}_{2^{16}})^n = (\mathbb{F}_{2^{16}})^{10}$, le l-uplet de mots de $\mathbb{F}_{2^{16}}$ $\mathrm{WH}^T_{\boldsymbol{Z}}(\boldsymbol{M})$ est défini par

$$\mathrm{WH}^T_{\boldsymbol{Z}}(\boldsymbol{M}) = (\mathrm{WH}_{Z_1,\ldots,Z_n}(\boldsymbol{M}), \mathrm{WH}_{Z_3,\ldots,Z_{n+2}}(\boldsymbol{M}), \ldots, \mathrm{WH}_{Z_{2l-1},\ldots,Z_{n+2l-2}}(\boldsymbol{M})) \ .$$

Avec les notations de la figure 4.4 la fonction h est définie par

$$h(\boldsymbol{Z}, \boldsymbol{A}, \boldsymbol{B}) = \mathrm{WH}^T_{\boldsymbol{Z}}(\boldsymbol{A} \| \boldsymbol{B}) \ .$$

On peut voir à partir des équations précédentes qu'étant donnée une paire $(\boldsymbol{A}, \boldsymbol{B})$ fixée, $h(\boldsymbol{Z}, \boldsymbol{A}, \boldsymbol{B})$ est une fonction quadratique connue de $(Z_1, \ldots, Z_{48})$. Cependant le caractère quadratique n'apporte pas ici une sécurité suffisante. En effet, pour une paire $(\boldsymbol{A}, \boldsymbol{B})$ fixée, chacun des $l = 20$ mots de 16 bits de $h(\boldsymbol{Z}, \boldsymbol{A}, \boldsymbol{B})$ peut être exprimé comme une fonction *affine* connue à coefficients dans $\mathbb{F}_{2^{16}}$ de seulement 15 mots de 16 bits inconnus, à savoir 10 valeurs consécutives de $(Z_1, \ldots, Z_{48})$ et 5 des 24 produits $Z_1 \cdot Z_2$, $Z_3 \cdot Z_4$, $\ldots$, $Z_{47} \cdot Z_{48}$. De façon équivalente, si l'on considère les équation sur $\mathbb{F}_2$ au lieu de $\mathbb{F}_{2^{16}}$, chaque bit de $h(\boldsymbol{Z}, \boldsymbol{A}, \boldsymbol{B})$ peut être exprimé, pour toute paire $(\boldsymbol{A}, \boldsymbol{B})$ fixée, comme une fonction affine connue de seulement 240 bits inconnus, à savoir 160 bits de $\boldsymbol{Z}$ et 80 fonctions quadratiques de bits de $\boldsymbol{Z}$. Remarquons de plus que les 16 premiers bits de $\boldsymbol{x}$, qui forment le mot $\mathrm{WH}_{Z_1,\ldots,Z_{10}}(\boldsymbol{A} \| \boldsymbol{B})$, dépendent des mêmes 240 bits inconnus, de même que les 16 bits suivants, etc. Nous appellerons par la suite ces bits inconnus bits de *clé étendue*. Ainsi, la fonction h implique 1 152 bits de clé étendue au total : 768 bits de $\boldsymbol{Z}$ et $24 \times 16 = 384$ fonctions quadratiques de bits de $\boldsymbol{Z}$. On voit donc que les clés de session sont des fonctions affines d'une clé secrète dont la longueur effective est peu supérieure à celle de la clé maitresse $\boldsymbol{Z}$.

Ces remarques faites, nous présentons maintenant la cryptanalyse effective de HB^{++}. Nous avons vu précédemment comment l'attaquant peut, en perturbant $s < r$ tours du protocole d'authentification, obtenir une approximation linéaire, avec une probabilité d'erreur η_{eff}, sur les bits du secret de session $\boldsymbol{x}$. Lorsque la phase de renouvellement des secrets de session est appliquée, cette équation linéaire approchée sur $\boldsymbol{x}$ se traduit par une équation linéaire approchée (avec la même probabilité d'erreur η_{eff}) sur les bits de la clé étendue. De plus, en choisissant un vecteur de perturbation $\boldsymbol{\delta}$ n'ayant pour support qu'un bloc de 16

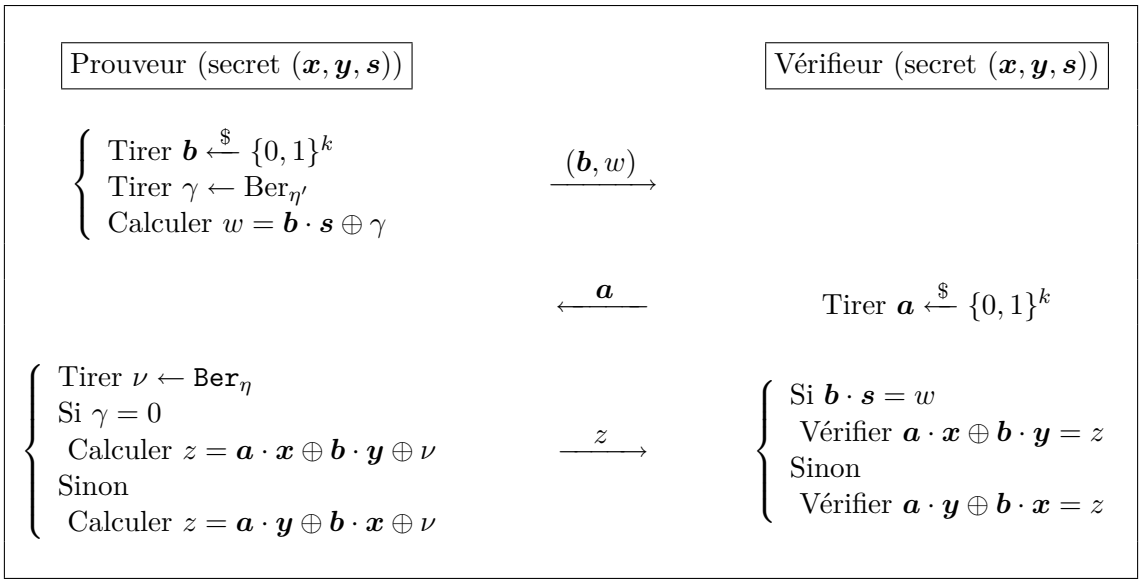

FIGURE 4.5 – Un tour du protocole HB*. Il est répété r fois et l'authentification est réussie si le nombre de tours tels que le test échoue est inférieur ou égal à un seuil t.

bits correspondant aux bits de $\boldsymbol{x}$ ne dépendant que des mêmes 240 bits de clé étendue, il est possible d'obtenir une équation linéaire approchée, toujours avec une probabilité d'erreur η_{eff}, sur seulement 240 bits de la clé étendue. En collectant suffisamment d'équations, on peut alors retrouver les 240 bits de clé étendue, en utilisant l'algorithme de résolution du problème LPN le plus approprié. Le paramètre η_{eff} étant généralement faible, les méthodes de résolution élémentaires évoquées à la section 3.5.1 sont généralement les plus efficaces. Une fois tous les bits de clé étendue intervenant dans le calcul de $\boldsymbol{x}$ retrouvés, l'attaquant peut calculer le vecteur $\boldsymbol{x}$ dans toutes les authentifications subséquentes. Il peut ainsi dériver des équations linéaires approchées sur les bits de clé étendue intervenant dans le calcul de $\boldsymbol{x}'$ et retrouver de la même façon ces bits de clé étendue.

À ce stade, l'attaquant possède suffisamment d'information pour se faire passer pour le prouveur légitime sans avoir à reconstituer l'intégralité de la clé étendue nécessaire pour dériver $\boldsymbol{y}$ et $\boldsymbol{y}'$. Il peut par exemple réutiliser des vecteurs de masquage $\boldsymbol{b}$ utilisés lors d'une authentification réussie et, à l'aide des vecteurs $\boldsymbol{x}$ et $\boldsymbol{x}'$ qu'il est maintenant en mesure de calculer, corriger les bits de réponse z et z' de façon appropriée. Pour les paramètres typiques proposés dans [BCD06], la table 4.2 indique que le paramètre η_{eff} sera suffisamment faible pour obtenir, avec une forte probabilité, des équations sans erreur sur les bits de clé étendue. La complexité résultante est donc quasiment linéaire en la taille de la clé étendue.

4.7 Cryptanalyse de la variante HB*

4.7.1 Description de HB*

La variante HB* a été proposée par Duc et Kim [DK07]. À nouveau, ce protocole consiste en r tours constitués chacuns de trois passes représentées sur la figure 4.5. En plus des deux secrets $\boldsymbol{x}$ et $\boldsymbol{y}$ de longueur k, un vecteur secret supplémentaire $\boldsymbol{s}$, également de longueur k, est utilisé pour transmettre secrètement un bit γ du prouveur vers le vérifieur,

valant 1 avec une probabilité $\eta' \in]0, \frac{1}{2}]$, et servant à indiquer si la réponse correcte doit être calculée selon la formule $\boldsymbol{a} \cdot \boldsymbol{x} \oplus \boldsymbol{b} \cdot \boldsymbol{y}$ ou $\boldsymbol{a} \cdot \boldsymbol{y} \oplus \boldsymbol{b} \cdot \boldsymbol{x}$. Comme dans HB$^+$, le prouveur est authentifié si le nombre d'erreurs est inférieur ou égal à un seuil $t = ur$, $u \in [\eta, \frac{1}{2}[$. Les formules des probabilités de faux rejet et de fausse acceptation sont identiques à celles de HB$^+$ (en particulier elles sont indépendantes de η'). Les calculs effectués par le prouveur sont environ deux fois plus importants que dans le cas de HB$^+$, mais restent inférieurs à ceux requis dans le cas de HB^{++}.

Les auteurs de HB* donnent dans leur article des arguments de sécurité heuristiques concernant les attaques de type GRS. En particulier, ils expliquent que l'observation des paires $(\boldsymbol{b}, \gamma)$ ne permet pas de retrouver le secret $\boldsymbol{s}$ (ce qui est vrai puisqu'il s'agit simplement d'une instance du problème LPN), si bien qu'un attaquant perturbant les vecteurs $\boldsymbol{a}$ en $\boldsymbol{a} \oplus \boldsymbol{\delta}$ sera dans l'impossibilité de savoir si l'erreur rajoutée est $\boldsymbol{\delta} \cdot \boldsymbol{x}$ ou $\boldsymbol{\delta} \cdot \boldsymbol{y}$. Cependant, nous allons voir que HB* n'offre en fait que peu de sécurité additionnelle contre les attaques GRS.

4.7.2 Cryptanalyse de HB*

HB* reste en fait vulnérable à une attaque extrêmement proche de l'attaque GRS. L'attaquant peut en effet obtenir de l'information sur $\boldsymbol{\delta} \cdot \boldsymbol{x}$ et $\boldsymbol{\delta} \cdot \boldsymbol{y}$ pour $\boldsymbol{\delta}$ arbitraire d'une manière très similaire. Pour cela, l'attaque consiste, exactement comme dans l'attaque GRS, à modifier les challenges $\boldsymbol{a}$ en leur ajoutant un vecteur $\boldsymbol{\delta}$ fixé. Lorsque $\boldsymbol{\delta} \cdot \boldsymbol{x} = 0$ et $\boldsymbol{\delta} \cdot \boldsymbol{y} = 0$, le protocole n'est pas perturbé et l'authentification réussit avec probabilité $1 - P_{\mathrm{FR}}$. Dans tous les autres cas, la probabilité d'authentification est moindre, si bien que la décision du vérifieur donne effectivement de l'information à l'attaquant.

Plus précisément, selon les valeurs de $\boldsymbol{\delta} \cdot \boldsymbol{x}$ et $\boldsymbol{\delta} \cdot \boldsymbol{y}$, la réponse du prouveur à chaque tour du protocole est incorrecte avec les probabilités suivantes :

1. si $\boldsymbol{\delta} \cdot \boldsymbol{x} = 0$ et $\boldsymbol{\delta} \cdot \boldsymbol{y} = 0$, comme nous venons de le voir, aucun des tours n'est perturbé. La réponse du prouveur est incorrecte lorsque $\nu = 1$, donc avec probabilité $\eta_1 = \eta$. Le vérifieur accepte avec probabilité $1 - P_{\mathrm{FR}}$ et rejette avec probabilité P_{FR}.
2. si $\boldsymbol{\delta} \cdot \boldsymbol{x} = 0$ et $\boldsymbol{\delta} \cdot \boldsymbol{y} = 1$, la réponse du prouveur est incorrecte lorsque $(\gamma = 0, \nu = 1)$ ou $(\gamma = 1, \nu = 0)$, donc avec probabilité
$$\eta_2 = (1 - \eta)\eta' + (1 - \eta')\eta = \eta + (1 - 2\eta)\eta' > \eta \ .$$
3. si $\boldsymbol{\delta} \cdot \boldsymbol{x} = 1$ et $\boldsymbol{\delta} \cdot \boldsymbol{y} = 0$, la réponse du prouveur est incorrecte lorsque $(\gamma = 0, \nu = 0)$ ou $(\gamma = 1, \nu = 1)$, donc avec probabilité
$$\eta_3 = (1 - \eta)(1 - \eta') + \eta\eta' = \eta + (1 - 2\eta)(1 - \eta') > \eta \ .$$
4. si $\boldsymbol{\delta} \cdot \boldsymbol{x} = 1$ et $\boldsymbol{\delta} \cdot \boldsymbol{y} = 1$, quelle que soit la valeur de γ, la réponse du prouveur est incorrecte lorsque $\nu = 0$, donc avec probabilité
$$\eta_4 = (1 - \eta) = \eta + (1 - 2\eta) > \eta \ .$$

Remarquons que $\eta_1 < \eta_2 \leq \frac{1}{2} \leq \eta_3 < \eta_4$. Remarquons également que lorsque $\eta' \to 0$, $\eta_2 \to \eta_1$ et $\eta_3 \to \eta_4$; le cas $\eta' = 0$ correspond exactement au protocole HB$^+$. Lorsque $\eta' \to \frac{1}{2}$, on a $\eta_2 \to \frac{1}{2}$ et $\eta_3 \to \frac{1}{2}$. Dans chacun des cas 2, 3 et 4, le vérifieur rejette l'authentification avec une probabilité supérieure à P_{FR}. Les probabilités d'acceptation et de rejet dans chacun des cas 1, 2, 3 et 4 sont données respectivement par

$$P_i^{\mathrm{acc}} = \sum_{j=0}^{t} \binom{r}{j} \eta_i^j (1 - \eta_i)^{r-j} \quad \text{et} \quad P_i^{\mathrm{rej}} = \sum_{j=t+1}^{r} \binom{r}{j} \eta_i^j (1 - \eta_i)^{r-j} \ .$$

Selon la valeur de η', il est possible de déduire de la décision du vérifieur le cas dans lequel on se trouve. Remarquons tout d'abord que puisque η_3 et η_4 sont tous deux supérieurs à $\frac{1}{2}$, les probabilités P_3^{acc} et P_4^{acc} sont toutes deux inférieures à la probabilité de fausse acceptation du protocole P_{FA}, donc négligeables. Le seul cas dépendant de la valeur de η' est donc le cas 2. Trois situations sont à distinguer selon la position de η_2 par rapport au seuil de rejet $u = t/r$.

Situation 1 : $\eta_2 < u - \epsilon$ pour une constante $\epsilon > 0$. Ceci correspond au cas où $\eta' < \frac{u-\eta-\epsilon}{1-2\eta}$. Dans ce cas les bornes de Chernoff indiquent que le vérifieur accepte avec une probabilité exponentiellement proche de un dans le cas 2 où $\boldsymbol{\delta} \cdot \boldsymbol{x} = 0$ et $\boldsymbol{\delta} \cdot \boldsymbol{y} = 1$. On a donc asymptotiquement :

$$P_1^{\text{acc}} \simeq 1, \quad P_2^{\text{acc}} \simeq 1, \quad P_3^{\text{acc}} \simeq 0, \quad P_4^{\text{acc}} \simeq 0 \ .$$

On est donc quasiment dans le cas de HB$^+$: en effet le vérifieur accepte avec forte probabilité lorsque $\boldsymbol{\delta} \cdot \boldsymbol{x} = 0$ et rejette avec forte probabilité lorsque $\boldsymbol{\delta} \cdot \boldsymbol{x} = 1$. L'attaque GRS s'applique donc exactement comme dans le cas de HB$^+$. L'attaquant peut retrouver $\boldsymbol{x}$ en temps linéaire avec $\mathcal{O}(k)$ vecteurs $\boldsymbol{\delta}$ indépendants. Une fois le vecteur $\boldsymbol{x}$ connu, l'attaquant peut se faire passer pour le prouveur en envoyant $(\boldsymbol{b}, w) = (\boldsymbol{0}, 0)$ comme premier message.

Situation 2 : $\eta_2 > u + \epsilon$ pour une constante $\epsilon > 0$. Ceci correspond au cas où $\eta' > \frac{u-\eta+\epsilon}{1-2\eta}$. Dans ce cas les bornes de Chernoff indiquent que le vérifieur rejette avec forte probabilité dans le cas 2 où $\boldsymbol{\delta} \cdot \boldsymbol{x} = 0$ et $\boldsymbol{\delta} \cdot \boldsymbol{y} = 1$. On a donc asymptotiquement :

$$P_1^{\text{acc}} \simeq 1, \quad P_2^{\text{acc}} \simeq 0, \quad P_3^{\text{acc}} \simeq 0, \quad P_4^{\text{acc}} \simeq 0 \ .$$

Le vérifieur accepte donc avec forte probabilité lorsque $\boldsymbol{\delta} \cdot \boldsymbol{x} = 0$ et $\boldsymbol{\delta} \cdot \boldsymbol{y} = 0$, et rejette avec forte probabilité dans tous les autres cas. On peut alors légèrement adapter l'attaque GRS.

En effet, supposons sans perte de généralité que $\boldsymbol{x}$ et $\boldsymbol{y}$ sont linéairement indépendants. Pour un $\boldsymbol{\delta}$ aléatoire, le cas 1 arrive avec probabilité $\frac{1}{4}$, si bien que l'attaquant peut trouver, en effectuant le test avec $\mathcal{O}(4k)$ vecteurs $\boldsymbol{\delta}$ aléatoires, $k-2$ vecteurs $\boldsymbol{\delta}$ indépendants tels que $\boldsymbol{\delta} \cdot \boldsymbol{x} = 0$ et $\boldsymbol{\delta} \cdot \boldsymbol{y} = 0$. Il est donc en mesure de retrouver l'espace vectoriel $\langle \boldsymbol{x}, \boldsymbol{y} \rangle$. Soient $\boldsymbol{c}_1$, $\boldsymbol{c}_2$ et $\boldsymbol{c}_3$ les trois vecteurs non nuls de cet espace. Une fois que l'attaquant les a retrouvés, il peut se faire passer pour le prouveur avec une probabilité de succès d'environ $\frac{1}{8}$ en choisissant deux vecteurs aléatoires parmi $\boldsymbol{c}_1$, $\boldsymbol{c}_2$ et $\boldsymbol{c}_3$ (disons $\boldsymbol{c}_1$ et $\boldsymbol{c}_2$), puis en fixant une valeur arbitraire pour $(\boldsymbol{b}, w)$ qu'il enverra à chaque tour, et en répondant par le bit $\boldsymbol{a} \cdot \boldsymbol{c}_1 \oplus \boldsymbol{b} \cdot \boldsymbol{c}_2$ à chaque tour. Il sera alors authentifié avec succès lorsque $(\boldsymbol{b} \cdot \boldsymbol{s} = w, \boldsymbol{c}_1 = \boldsymbol{x}, \boldsymbol{c}_2 = \boldsymbol{y})$ ou $(\boldsymbol{b} \cdot \boldsymbol{s} \neq w, \boldsymbol{c}_1 = \boldsymbol{y}, \boldsymbol{c}_2 = \boldsymbol{x})$, ce qui arrive avec probabilité $\frac{1}{8}$.

Mais l'attaquant peut aussi pousser l'attaque plus loin et identifier parmi les trois vecteurs $\boldsymbol{c}_1$, $\boldsymbol{c}_2$ et $\boldsymbol{c}_3$ celui qui est égal à $\boldsymbol{x} \oplus \boldsymbol{y}$. Pour cela, l'attaquant interroge le prouveur légitime avec des challenges $\boldsymbol{a}$ systématiquement égaux aux vecteurs $\boldsymbol{b}$ envoyés par le prouveur. Ainsi, la réponse du prouveur est toujours égale à $\boldsymbol{b} \cdot (\boldsymbol{x} \oplus \boldsymbol{y}) \oplus \nu$. L'attaquant peut donc en déduire que $\boldsymbol{x} \oplus \boldsymbol{y}$ est le vecteur $\boldsymbol{c}_i$ qui maximise le nombre de vecteurs $\boldsymbol{b}$ tels que $\boldsymbol{b} \cdot \boldsymbol{c}_i$ est égal à la réponse du prouveur. Une fois cette opération effectuée, l'attaquant connait l'ensemble non ordonné $\{\boldsymbol{x}, \boldsymbol{y}\}$. Ceci est suffisant pour se faire passer pour le prouveur avec une probabilité de succès de $\frac{1}{2}$. Supposons que le vecteur $\boldsymbol{c}_3$ ait été exclu comme étant le vecteur $\boldsymbol{x} \oplus \boldsymbol{y}$. L'attaquant fixe alors une valeur arbitraire pour $(\boldsymbol{b}, w)$ qu'il utilisera à chaque tour du protocole, et répond par le bit $\boldsymbol{a} \cdot \boldsymbol{c}_1 \oplus \boldsymbol{b} \cdot \boldsymbol{c}_2$ à

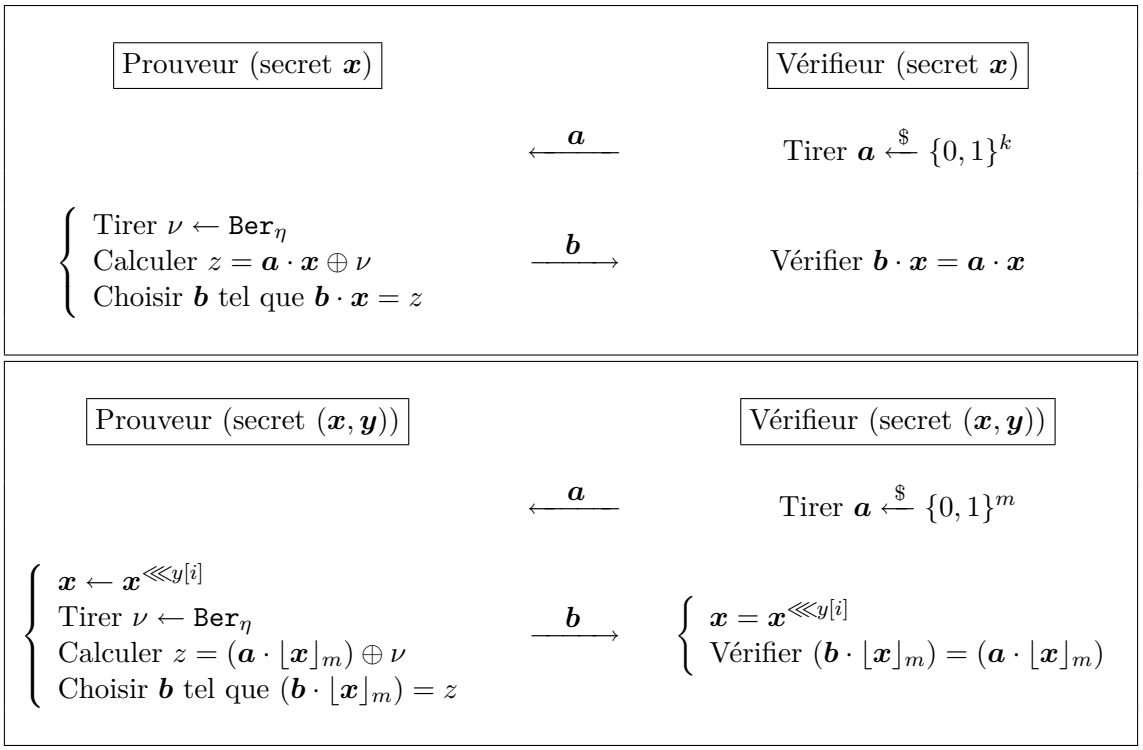

FIGURE 4.6 – Un tour du protocole HB-MP' (en haut) et du protocole HB-MP (en bas). Il est répété r fois et l'authentification est réussie si le nombre de tours tels que le test échoue est inférieur ou égal à un seuil t.

chaque tour. L'attaquant sera authentifié avec succès si $(\boldsymbol{b} \cdot \boldsymbol{s} = w, \boldsymbol{c}_1 = \boldsymbol{x}, \boldsymbol{c}_2 = \boldsymbol{y})$ ou $(\boldsymbol{b} \cdot \boldsymbol{s} \neq w, \boldsymbol{c}_1 = \boldsymbol{y}, \boldsymbol{c}_2 = \boldsymbol{x})$, ce qui arrive maintenant avec probabilité $\frac{1}{2}$. Remarquons que quel que soit le résultat de cette première tentative, l'attaquant pourra s'authentifier avec probabilité 1 à la tentative suivante. Si la première tentative était réussie il peut réutiliser le même $(\boldsymbol{b}, w)$ et répondre $\boldsymbol{a} \cdot \boldsymbol{c}_1 \oplus \boldsymbol{b} \cdot \boldsymbol{c}_2$ à chaque tour. Si la première tentative a échoué il réutilise le même $(\boldsymbol{b}, w)$ mais répond $\boldsymbol{a} \cdot \boldsymbol{c}_2 \oplus \boldsymbol{b} \cdot \boldsymbol{c}_1$ à chaque tour. La réponse sera toujours correcte et l'adversaire sera authentifié.

Situation 3 : $\eta_2 \simeq u$. Dans le dernier cas le résultat de l'authentification dans le cas 2 n'est pas sûr : les probabilités de rejet et d'acceptation sont toutes deux notables. On peut cependant remarquer que lorsque $\boldsymbol{\delta} \cdot \boldsymbol{x} = 1$, le vérifieur rejette toujours avec une probabilité exponentiellement proche de un. On peut donc déduire, lorsque le vérifieur accepte l'authentification, que $\boldsymbol{\delta} \cdot \boldsymbol{x} = 0$. Ceci permet de retrouver le vecteur $\boldsymbol{x}$ à l'aide d'au plus $\mathcal{O}(4k)$ vecteurs $\boldsymbol{\delta}$ indépendants, car le cas 1 arrive une fois sur quatre pour un vecteur $\boldsymbol{\delta}$ aléatoire. On peut alors procéder comme pour la situation 1.

4.8 Cryptanalyse de la variante HB-MP

Les variantes HB-MP' et HB-MP ont été proposées par Munilla et Peinado [MP07]. Contrairement à HB$^+$, chaque tour de ces protocoles n'utilise que deux passes entre le prouveur et le vérifieur. Ceci est illustré sur la figure 4.6 où un tour de chaque protocole est représenté. Dans le protocole HB-MP', le prouveur et le vérifieur partagent un vecteur secret $\boldsymbol{x}$ de longueur k. À chaque tour, le vérifieur tire un vecteur $\boldsymbol{a}$ aléatoire de longueur

k qu'il transmet au prouveur. Celui-ci calcule $z = \boldsymbol{a} \cdot \boldsymbol{x} \oplus \nu$, avec $\nu \leftarrow \mathtt{Ber}_\eta$, puis choisit aléatoirement un vecteur $\boldsymbol{b}$ tel que $\boldsymbol{b} \cdot \boldsymbol{x} = z$ qu'il renvoie au vérifieur. Ce dernier vérifie alors que $\boldsymbol{b} \cdot \boldsymbol{x} = \boldsymbol{a} \cdot \boldsymbol{x}$. À l'issue de r tours, le vérifieur accepte l'authentification si le nombre d'erreurs est inférieur ou égal à un seuil $t = ur$, avec $u \in [\eta, \frac{1}{2}[$.

Dans le protocole HB-MP, le prouveur et le vérifieur partagent un vecteur secret additionnel $\boldsymbol{y}$ de longueur k. Chaque tour du protocole se déroule de façon très similaire à HB-MP'. Le vérifieur tire un vecteur $\boldsymbol{a}$ aléatoire de longueur $m < k$ qu'il transmet au prouveur. Au i-ème tour, le prouveur met à jour le vecteur secret $\boldsymbol{x}$ en effectuant une rotation de ses bits de 0 ou 1 position vers la gauche selon la valeur du bit en position i de $\boldsymbol{y}$: $\boldsymbol{x} \leftarrow \boldsymbol{x}^{\lll y[i]}$. Puis il calcule $z = (\boldsymbol{a} \cdot \lfloor \boldsymbol{x} \rfloor_m) \oplus \nu$, où $\lfloor \boldsymbol{x} \rfloor_m$ dénote les m bits les plus à droite de $\boldsymbol{x}$ et $\nu \leftarrow \mathtt{Ber}_\eta$. Il choisit alors un vecteur $\boldsymbol{b}$ aléatoire tel que $(\boldsymbol{b} \cdot \lfloor \boldsymbol{x} \rfloor_m) = z$ qu'il renvoie au vérifieur. Celui-ci vérifie alors que $(\boldsymbol{b} \cdot \lfloor \boldsymbol{x} \rfloor_m) = (\boldsymbol{a} \cdot \lfloor \boldsymbol{x} \rfloor_m)$. À l'issue de r tours, le vérifieur accepte l'authentification si le nombre d'erreurs est inférieur ou égal à un seuil $t = ur$, avec $u \in [\eta, \frac{1}{2}[$.

Munilla et Peinado affirment dans leur article que HB-MP' est résistant aux attaques actives et que HB-MP est résistant aux attaques GRS. Nous allons voir cependant que ces deux protocoles sont vulnérables à une attaque passive, et donc sensiblement plus faibles que HB$^+$: un attaquant qui intercepte les r tours d'une authentification entre un prouveur et un vérifieur légitimes peut ensuite se faire passer pour le prouveur avec un probabilité proche de un. En effet, remarquons que le test fait par le vérifieur consiste à vérifier que $(\boldsymbol{a} \oplus \boldsymbol{b}) \cdot \lfloor \boldsymbol{x} \rfloor_m = 0$. Cette équation est toujours vraie lorsque $\boldsymbol{b} = \boldsymbol{a}$, et Munilla et Peinado précisent bien que le vérifieur doit rejeter l'authentification dès que le prouveur renvoie $\boldsymbol{a}$ à l'un des tours. Cependant, pour un attaquant qui a intercepté les r tours d'une exécution précédente du protocole, il est aisé de calculer un vecteur $\boldsymbol{b}$ différent de $\boldsymbol{a}$ et tel que $(\boldsymbol{a} \oplus \boldsymbol{b}) \cdot \lfloor \boldsymbol{x} \rfloor_m = 0$ avec forte probabilité.

L'attaquant enregistre simplement les r paires $(\boldsymbol{a}_i, \boldsymbol{b}_i)$ qui sont échangées lors d'une authentification entre le prouveur et le vérifieur. On a alors que $(\boldsymbol{a}_i \oplus \boldsymbol{b}_i) \cdot \lfloor \boldsymbol{x} \rfloor_m = 0$ avec probabilité $1 - \eta$. Par conséquent, pour tout challenge $\boldsymbol{a}'_i$, le vecteur $\boldsymbol{b}'_i = \boldsymbol{a}'_i \oplus \boldsymbol{a}_i \oplus \boldsymbol{b}_i$ est différent de $\boldsymbol{a}'_i$ (car $\boldsymbol{b}_i \neq \boldsymbol{a}_i$) et vérifie $(\boldsymbol{a}'_i \oplus \boldsymbol{b}'_i) \cdot \lfloor \boldsymbol{x} \rfloor_m = (\boldsymbol{a}_i \oplus \boldsymbol{b}_i) \cdot \lfloor \boldsymbol{x} \rfloor_m$. Par conséquent l'attaquant est authentifié dès que le prouveur l'a été lors de l'authentification interceptée. L'attaque fonctionne exactement de la même façon contre HB-MP'.

4.9 Les nouvelles variantes Trusted-HB et PUF-HB

Très récemment, deux nouvelles variantes du protocole HB$^+$ ont été proposées. Suite à la cryptanalyse de HB^{++}, Bringer et Chabanne ont proposé la variante Trusted-HB [BC08]. Ce protocole comporte deux phases. La première est identique à HB$^+$: r tours du protocole HB$^+$ sont effectués, au cours desquels le prouveur et le vérifieur échangent les vecteurs $\boldsymbol{b}_i$, $\boldsymbol{a}_i$ et le bit $z_i = \boldsymbol{a}_i \cdot \boldsymbol{x} \oplus \boldsymbol{b}_i \cdot \boldsymbol{y} \oplus \nu_i$, pour $i = 0$ à $r - 1$. Puis le prouveur envoie un dernier message constituant un MAC des échanges précédents utilisant une clé secrète partagée supplémentaire K (la construction du MAC est détaillée dans [BC08] et s'inspire d'un schéma dû à Krawczyk [Kra94]). Fondamentalement, l'authentification est réalisée par ce dernier message (en fait celui-ci est intrinsèquement suffisant pour authentifier le prouveur). La première phase du protocole permet en fait au vérifieur *d'identifier* secrètement le prouveur grâce aux vecteurs $\boldsymbol{x}$ et $\boldsymbol{y}$ sans qu'un attaquant puisse compromettre l'anonymat du prouveur : le vérifieur recherche simplement dans une base de données $(\boldsymbol{x}_i, \boldsymbol{y}_i)_{i \in [\![1,N]\!]}$ le couple $(\boldsymbol{x}_j, \boldsymbol{y}_j)$ vérifiant environ une fraction $1 - \eta$ des équations $z_i = \boldsymbol{a}_i \cdot \boldsymbol{x}_j \oplus \boldsymbol{b}_i \cdot \boldsymbol{y}_j$. Il y

a donc dans Trusted-HB un déplacement de fonctionnalité du protocole HB^+ de l'authentification vers l'identification « respectueuse de la vie privée » (traduction approximative du terme anglais *private*). Nous reviendrons sur cet aspect du protocole HB^+ dans la section 5.8. Cependant, Frumkin et Shamir ont récemment publié une cryptanalyse de Trusted-HB [FS09].

Hammouri et Sunar ont eu l'idée de combiner HB^+ avec les *fonctions physiquement inclonables*, ou PUF (*Physically Unclonable Function*), et ont baptisé leur proposition PUF-HB [HS08]. Ce protocole est similaire à HB^+ mais la façon dont le bit de réponse z est calculé est légèrement modifiée : le prouveur calcule $z = \boldsymbol{b} \cdot \boldsymbol{y} \oplus \mathrm{PUF}(\boldsymbol{a}) \oplus \nu$, où PUF est un fonction physiquement inclonable. Le vérifieur ne dispose pas de la fonction PUF qui par définition ne peut être implantée une seconde fois mais d'une modélisation à ϵ près PUF_ϵ, suffisante pour effectuer les vérifications nécessaires. PUF-HB est prouvé sûr contre les attaques actives, et les auteurs montrent que l'attaque GRS ne s'applique pas en raison de la non-linéarité de la fonction PUF, mais ne donnent pas de preuve de sécurité rigoureuse contre les attaques GRS. Remarquons qu'un attaquant man-in-the-middle pouvant facilement retrouver $\boldsymbol{y}$ en perturbant le vecteur de masquage $\boldsymbol{b}$, la sécurité de PUF-HB est ramenée au problème de modéliser la fonction $\mathrm{PUF}(\boldsymbol{a})$ en connaissant des exemples bruités de cette fonction ($\boldsymbol{a}$ étant éventuellement perturbé par l'attaquant).

Chapitre 5

HB$^{\#}$, une variante à sécurité prouvée contre les attaques GRS

Tâchons de résumer les forces et faiblesses du protocole HB^{+}. Du côté positif, il est très simple, ne mettant en jeu que des opérations élémentaires sur GF(2), et il possède une preuve de sécurité contre les attaques actives. Du côté négatif, le coût de transmission est très élevé car il faut échanger deux vecteurs de k bits pour calculer un seul bit de réponse. Par ailleurs, ce grand nombre de messages échangés donne beaucoup d'emprise à un attaquant man-in-the-middle, comme le montre l'attaque GRS. HB^{+} semble donc susceptible d'être amélioré sur le plan du coût de communication et de la sécurité.

Comme nous l'avons vu au chapitre précédent, les trois tentatives de variantes HB^{++}, HB* et HB-MP visant à rendre HB^{+} résistant aux attaques man-in-the-middle se sont avérées inopérantes. Cela ne signifie pourtant pas que toute entreprise d'amélioration de HB^{+} soit vouée à l'échec. Suite à la cryptanalyse de HB^{++}, HB* et HB-MP, nous avons proposé notre propre variante de HB^{+}. L'idée principale consiste à généraliser la forme des secrets : au lieu d'utiliser des vecteurs, nous proposons d'utiliser des matrices.

5.1 Description de RANDOM-HB$^{\#}$

Le protocole RANDOM-HB$^{\#}$ constitue une généralisation du protocole HB^{+}, dans laquelle les secrets ne sont plus des vecteurs $\boldsymbol{x}$ et $\boldsymbol{y}$ mais des matrices binaires X et Y de tailles respectives $k_X \times m$ et $k_Y \times m$. Le préfixe RANDOM indique que ces deux matrices sont tirées de façon aléatoire parmi l'ensemble des matrices binaires, contrairement à la variante HB$^{\#}$ où elles auront une forme particulière, comme nous le verrons à la section 5.5.

RANDOM-HB$^{\#}$ ne comporte qu'un seul tour représenté sur la figure 5.1 et se déroulant de la façon suivante : le prouveur envoie un vecteur aléatoire $\boldsymbol{b}$ de longueur k_Y au vérifieur ; celui-ci renvoie un vecteur aléatoire $\boldsymbol{a}$ de longueur k_X ; le prouveur calcule alors le *vecteur* réponse $\boldsymbol{z} = \boldsymbol{a} \cdot X \oplus \boldsymbol{b} \cdot Y \oplus \boldsymbol{\nu}$, de longueur m, où $\boldsymbol{\nu} \leftarrow \mathtt{Ber}_{m,\eta}$ est un vecteur de bruit donc chaque bit est tiré indépendamment selon $\mathtt{Ber}_{\eta}$, et l'envoie au vérifieur. La vérification consiste à comparer les deux vecteurs $\boldsymbol{z}$ et $\boldsymbol{a} \cdot X \oplus \boldsymbol{b} \cdot Y$: l'authentification est acceptée si et seulement si ces deux vecteurs diffèrent en au plus t positions, *i.e.* $\mathtt{Hwt}(\boldsymbol{a} \cdot X \oplus \boldsymbol{b} \cdot Y \oplus \boldsymbol{z}) \leq t$, où $t = um$ pour un paramètre $u \in [\eta, \frac{1}{2}[$. Le protocole RANDOM-HB$^{\#}$ peut donc être vu comme m itérations du protocole HB^{+} avec des paires de secrets $(\boldsymbol{x}, \boldsymbol{y})$ différentes et indépendantes correspondant aux colonnes de X et Y, et la même paire de vecteur de masquage et de vecteur challenge $(\boldsymbol{b}, \boldsymbol{a})$.

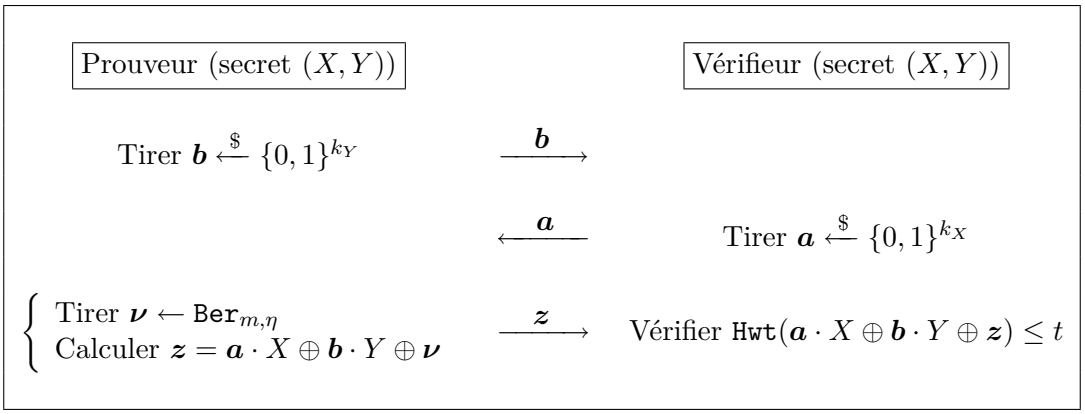

FIGURE 5.1 – Le protocole RANDOM-HB$^{\#}$. Il ne comporte qu'un seul tour et l'authentification est acceptée si le nombre de positions $i \in [\![1, m]\!]$ telles que $(\boldsymbol{a} \cdot X \oplus \boldsymbol{b} \cdot Y)[i] \neq \boldsymbol{z}[i]$ est inférieur ou égal à un seuil t.

Les probabilités de faux rejet et de fausse acceptation de RANDOM-HB$^{\#}$ sont données par des expressions similaires à celles de HB^{+}, à la substitution près du nombre de tours r de HB^{+} par le nombre de colonnes m des matrices secrètes X et Y :

$$P_{\mathrm{FR}} = \sum_{i=t+1}^{m} \binom{m}{i} \eta^i (1-\eta)^{m-i} \quad \text{et} \quad P_{\mathrm{FA}} = \frac{1}{2^m} \sum_{i=0}^{t} \binom{m}{i} \ .$$

Une grande partie du reste de ce chapitre est dédiée à l'étude de la sécurité du protocole RANDOM-HB$^{\#}$. Nous allons voir que tout comme HB^{+}, RANDOM-HB$^{\#}$ est sûr dans le modèle des attaques actives. Mais l'atout majeur de RANDOM-HB$^{\#}$ est qu'il est également sûr contre les attaques GRS, auxquelles HB^{+} est vulnérable avec une complexité linéaire.

Les preuves de sécurité qui vont suivre sont complexes, et réduire la sécurité de RANDOM-HB$^{\#}$ directement au problème LPN s'avère fastidieux. Comme le suggèrent Katz et Shin [KS06a], le protocole HB^{+} peut être vu comme un puzzle faiblement vérifiable, notion introduite par Canetti, Halevi et Steiner [CHS05] et que nous présentons dans la section 5.2. Informellement, elle désigne une classe de problèmes pour laquelle la justesse d'une réponse ne peut être vérifiée que par celui qui a généré l'instance. Ces mêmes auteurs ont étudié l'amplification de difficulté des puzzles faiblement vérifiables. L'amplification de difficulté (en anglais *hardness amplification*) est un sujet fondamental en cryptographie, et consiste à déterminer si résoudre plusieurs instances indépendantes d'un certain problème est plus dur que résoudre une unique instance de ce problème. Le résultat le plus célèbre dans ce domaine est sans doute le « XOR-Lemma » de Yao [Yao82] montrant que l'existence de fonctions à sens unique au sens faible implique l'existence de fonctions à sens unique au sens fort.

Canetti *et al.* ont montré que dans le cas des puzzles faiblement vérifiables, si aucun algorithme efficace ne peut résoudre une unique instance d'un puzzle avec une probabilité meilleure que ϵ, alors aucun algorithme efficace ne peut résoudre m instances indépendantes de ce puzzle avec une probabilité meilleure que ϵ^m. Nous allons utiliser ce résultat pour définir un puzzle (que nous nommerons *puzzle MHB*) et démontrer sa difficulté. Puis, nous réduirons la sécurité de RANDOM-HB$^{\#}$ dans le modèle des attaques actives à la difficulté du puzzle MHB. Enfin, nous réduirons la sécurité de RANDOM-HB$^{\#}$ dans le modèle des attaques GRS à sa sécurité dans le modèle des attaques actives.

5.2 Le puzzle MHB et sa difficulté

Nous commençons par donner les définitions de base concernant les puzzles faiblement vérifiables (nous parlerons simplement de puzzles par la suite, il sera toujours sous-entendu qu'ils sont faiblement vérifiables). Le lecteur intéressé pourra se référer à [CHS05] pour de plus amples détails.

Définition 5.1 (Puzzle faiblement vérifiable)
Un puzzle faiblement vérifiable est une paire d'algorithmes efficaces $\mathcal{Z} = (\mathcal{G}, \mathcal{V})$ tels que :
- L'algorithme de génération de puzzle $\mathcal{G}$ reçoit en entrée le paramètre de sécurité 1^k et retourne un puzzle aléatoire p et une information de contrôle c, $(p, c) \xleftarrow{\$} \mathcal{G}(1^k)$.
- L'algorithme de vérification $\mathcal{V}$ est un algorithme déterministe et efficace qui prend en entrée un puzzle p, une information de contrôle c et une réponse a, et retourne 0 lorsque la réponse est fausse et 1 lorsque la réponse est juste.

Un algorithme de résolution pour un puzzle est un algorithme efficace $\mathcal{S}$ qui prend en entrée un puzzle p et retourne une réponse a. Sa probabilité de succès est définie par :

$$\mathrm{succ}_{\mathcal{Z}}[\mathcal{S}] \stackrel{\text{def}}{=} \Pr_{\mathcal{G},\mathcal{S}}\left[(p, c) \xleftarrow{\$} \mathcal{G}(1^k), a \xleftarrow{\$} \mathcal{S}(p) \ : \ \mathcal{V}(p, c, a) = 1\right] \ ,$$

où l'aléa est entendu comme étant l'aléa interne de $\mathcal{G}$ et $\mathcal{S}$. ♦

Définition 5.2 (Difficulté d'un puzzle)
Soit $\epsilon : \mathbb{N} \to]0, 1[$ une fonction quelconque. Un puzzle $\mathcal{Z} = (\mathcal{G}, \mathcal{V})$ est dit $(1-\epsilon)$-dur si pour tout algorithme de résolution efficace $\mathcal{S}$ il existe une fonction négligeable du paramètre de sécurité $\mathtt{negl}$ telle que la probabilité de succès de $\mathcal{S}$ est inférieure à $\epsilon + \mathtt{negl}(k)$. ♦

Il nous faut définir la répétition d'un puzzle, ce qui se fait de manière très naturelle.

Définition 5.3 (Répétition d'un puzzle)
Soit $\mathcal{Z} = (\mathcal{G}, \mathcal{V})$ un puzzle et $m : \mathbb{N} \to \mathbb{N}$ une fonction quelconque du paramètre de sécurité. La répétition à l'ordre m du puzzle $\mathcal{Z}$, notée $\mathcal{Z}^m = (\mathcal{G}^m, \mathcal{V}^m)$, est le puzzle tel que $\mathcal{G}^m$, recevant en entrée le paramètre de sécurité 1^k, fait appel à $\mathcal{G}(1^k)$ m fois et retourne les m puzzles et leur information de contrôle respective :

$$((p_1, c_1), \ldots, (p_m, c_m)) \xleftarrow{\$} \mathcal{G}^m(1^k) \ .$$

Pour une réponse $(a_1, \ldots, a_m)$, l'algorithme de vérification $\mathcal{V}^m$ retourne 1 si et seulement si $\mathcal{V}(p_i, c_i, a_i) = 1$ pour tout $i \in [\![1, m]\!]$. ♦

Le résultat principal de Canetti *et al.* est l'expression de la difficulté de la répétition d'un puzzle en fonction de celle du puzzle répété : si un puzzle $\mathcal{Z}$ est $(1 - \epsilon)$-dur, alors sa répétition $\mathcal{Z}^m$ à l'ordre m est $(1 - \epsilon^m)$-dur. Plus précisément, on a le lemme suivant, que l'on peut obtenir en combinant le lemme 1 et la section 3.3 de l'article [CHS05] :

Lemme 5.1 ([CHS05], Lemme 1). *Soient $m : \mathbb{N} \to \mathbb{N}$ et ϵ, $\delta : \mathbb{N} \to]0, 1[$ des fonctions du paramètre de sécurité k calculables efficacement. Soit $\mathcal{Z} = (\mathcal{G}, \mathcal{V})$ un puzzle. S'il existe un algorithme de résolution $\mathcal{S}$ pour $\mathcal{Z}^m$, dont le temps de calcul est au plus T et ayant une probabilité de succès supérieure à $\epsilon^m + \delta$, alors il existe un algorithme de résolution $\mathcal{S}'$ pour $\mathcal{Z}$ ayant une probabilité de succès supérieure à $\epsilon + \frac{\delta}{8m}$ dont le temps de calcul est au plus T', où T' est polynomial en m, $\frac{1}{\delta}$, $\frac{1}{\epsilon^m + \delta}$, et les temps de calcul de $\mathcal{S}$, $\mathcal{G}$ et $\mathcal{V}$.* ▽

Nous allons maintenant voir comment appliquer ce résultat au protocole HB. On peut aisément exprimer le fait d'attaquer le protocole HB réduit à un seul tour comme un puzzle. C'est ce que nous appellerons le « puzzle HB » :

Définition 5.4 (Puzzle HB)
Soit $\eta \in]0, \frac{1}{2}[$, et soit q un polynôme en k. Le puzzle HB est le puzzle $(\mathcal{G}, \mathcal{V})$ tel que :
- L'algorithme de génération $\mathcal{G}$, lorsqu'il reçoit en entrée le paramètre de sécurité 1^k, tire un vecteur aléatoire $\boldsymbol{x}$ de longueur k, q vecteurs aléatoires $(\boldsymbol{a}_1, \ldots, \boldsymbol{a}_q)$ de longueur k, et calcule pour $1 \leq i \leq q$ les bits $z_i = \boldsymbol{a}_i \cdot \boldsymbol{x} \oplus \nu_i$, où $\nu_i = 1$ avec probabilité η. Il tire également un vecteur aléatoire $\boldsymbol{a}$ de longueur k constituant le challenge. Il retourne $\{(\boldsymbol{a}_i, z_i)\}_{1 \leq i \leq q}$ et $\boldsymbol{a}$ qui constituent le puzzle p.
- Un algorithme de résolution retourne un bit z.
- L'information de contrôle secrète est $\boldsymbol{x}$, et l'algorithme de vérification $\mathcal{V}$ retourne 1 si et seulement si $z = \boldsymbol{a} \cdot \boldsymbol{x}$. ♦

Le résultat de Juels et Weis sur la sécurité de HB [JW05a, Lemme 1] peut alors être exprimé en termes de difficulté du puzzle HB comme suit :

Lemme 5.2. *Supposons que le problème LPN est dur. Alors le puzzle HB est* $\left(1 - \frac{1}{2}\right)$*-dur,* i.e. *tout algorithme de résolution efficace* $\mathcal{S}$ *de ce puzzle a une probabilité de succès inférieure à* $\frac{1}{2} + \mathtt{negl}(k)$. ▽

Nous allons maintenant utiliser la théorie des puzzles afin de définir une généralisation matricielle du puzzle HB, le « puzzle MHB », et de démontrer sa difficulté.

Définition 5.5 (Puzzle MHB)
Soit $\eta \in]0, \frac{1}{2}[$, et soient m et q deux polynômes en k. Le puzzle MHB d'ordre m est $(\mathcal{G}, \mathcal{V})$ tel que :
- L'algorithme de génération $\mathcal{G}$, lorsqu'il reçoit en entrée le paramètre de sécurité 1^k, tire une matrice aléatoire X de taille $k \times m$, et q vecteurs aléatoires $(\boldsymbol{a}_1, \ldots, \boldsymbol{a}_q)$ de longueur k, et calcule pour $1 \leq i \leq q$ les vecteurs $\boldsymbol{z}_i = \boldsymbol{a}_i \cdot X \oplus \boldsymbol{\nu}_i$, où chaque bit de $\boldsymbol{\nu}_i$ vaut 1 avec probabilité η. Il tire également un vecteur aléatoire $\boldsymbol{a}$ de longueur k constituant le challenge. Il retourne $\{(\boldsymbol{a}_i, \boldsymbol{z}_i)\}_{1 \leq i \leq q}$ et $\boldsymbol{a}$ qui constituent le puzzle p.
- Un algorithme de résolution retourne un vecteur $\boldsymbol{z}$ de longueur m.
- L'information de contrôle secrète est X, et l'algorithme de vérification $\mathcal{V}$ retourne 1 si et seulement si $\boldsymbol{z} = \boldsymbol{a} \cdot X$. ♦

Remarquons que le puzzle MHB n'est pas exactement la répétition à l'ordre m du puzzle HB. En effet les m puzzles HB correspondant à chaque colonne de la matrice X du puzzle MHB partagent certaines données, à savoir les vecteurs $\boldsymbol{a}_i$ et le challenge $\boldsymbol{a}$. Par conséquent pour appliquer le lemme 5.1 nous devons tout d'abord généraliser la notion de répétition de puzzle au cas où les différents puzzles partagent des éléments communs. Pour cela nous supposerons que l'algorithme de génération de puzzle $\mathcal{G}$ se compose en fait de deux algorithmes $\mathcal{G} = (\mathcal{G}_f, \mathcal{G}_v)$ (pour *fixe* et *variable*). Lorsque $\mathcal{G}$ reçoit le paramètre de sécurité 1^k en entrée, il appelle $\mathcal{G}_f(1^k)$ qui renvoie aléatoirement une partie « fixe » du puzzle, $p_f \xleftarrow{\$} \mathcal{G}_f(1^k)$. Puis $\mathcal{G}$ appelle $\mathcal{G}_v(p_f)$ qui renvoie aléatoirement une partie « variable » du puzzle, ainsi que l'information de contrôle, $(p_v, c) \xleftarrow{\$} \mathcal{G}_v(p_f)$. $\mathcal{G}$ renvoie alors le puzzle $p = (p_f, p_v)$ et l'information de contrôle c. La probabilité de succès d'un algorithme de résolution est définie naturellement :

$$\mathrm{succ}_{\mathcal{Z}}[\mathcal{S}] = \Pr_{(\mathcal{G}_f, \mathcal{G}_v), \mathcal{S}}\left[((p_f, p_v), c) \xleftarrow{\$} \mathcal{G}(1^k), a \xleftarrow{\$} \mathcal{S}(p_f, p_v) : \mathcal{V}((p_f, p_v), c, a) = 1\right] .$$

Nous pouvons alors définir la pseudo-répétition $\widetilde{\mathcal{Z}^m}$ d'un puzzle $\mathcal{Z} = ((\mathcal{G}_f, \mathcal{G}_v), \mathcal{V})$ comme suit : le générateur de puzzle $\widetilde{\mathcal{G}^m}$, lorsqu'il reçoit en entrée le paramètre de sécurité

1^k, appelle $\mathcal{G}_f(1^k)$ une seule fois : $p_f \xleftarrow{\$} \mathcal{G}_f(1^k)$, puis appelle $\mathcal{G}_v$ m fois : $(p_v^i, c_i) \xleftarrow{\$} \mathcal{G}_v(p_f)$, $1 \leq i \leq m$, et retourne les m puzzles (p_f, p_v^i) avec leur information de contrôle :

$$(((p_f, p_v^1), c_1), \ldots, ((p_f, p_v^m), c_m)) \xleftarrow{\$} \widetilde{\mathcal{G}^m}(1^k) \ .$$

L'algorithme de vérification $\widetilde{\mathcal{V}^m}$ retourne 1 si et seulement si $\mathcal{V}$ retourne 1 pour la réponse à chacun des m puzzles. La probabilité de succès d'un algorithme de résolution pour le puzzle $\widetilde{\mathcal{Z}^m}$ est définie comme pour un puzzle habituel. Nous pouvons maintenant généraliser le lemme 5.1 à la pseudo-répétition d'un puzzle.

Lemme 5.3. *Soient $m : \mathbb{N} \to \mathbb{N}$ et ϵ, $\delta : \mathbb{N} \to]0,1[$ des fonctions du paramètre de sécurité k calculables efficacement. Soit $\mathcal{Z} = ((\mathcal{G}_f, \mathcal{G}_v, \mathcal{V})$ un puzzle tel que pour tout (p_f, p_v, c), une réponse aléatoire a une probabilité ϵ d'être acceptée par $\mathcal{V}$ [1]. S'il existe un algorithme de résolution $\mathcal{S}$ pour $\widetilde{\mathcal{Z}^m}$, dont le temps de calcul est au plus T, et ayant une probabilité de succès supérieure à $\epsilon^m + \delta$, alors il existe un algorithme de résolution $\mathcal{S}'$ pour $\mathcal{Z}$ ayant une probabilité de succès supérieure à $\epsilon + \frac{\delta^2}{32m}$ dont le temps de calcul est au plus T', où T' est polynomial en m, $\frac{1}{\delta}$, $\frac{1}{\epsilon^m + \delta}$, et les temps de calcul de $\mathcal{S}$, $\mathcal{G}$ et $\mathcal{V}$.* ▽

Démonstration. Notons (p_f, p_v) le puzzle que cherche à résoudre $\mathcal{S}'$. Soit E l'événement, défini sur l'espace des parties fixes p_f, consistant en ce que la probabilité, prise sur la séquence $(p_v^1, \ldots, p_v^m)$ et l'aléa de $\mathcal{S}$, que $\mathcal{S}$ résolve la pseudo-répétition du puzzle, soit supérieure à $\epsilon^m + \frac{\delta}{2m}$. D'après le lemme B.1, la probabilité, prise sur p_f, de l'événement E, est supérieure à $\frac{\delta}{2}$. Lorsque E se produit [2], on peut considérer $\mathcal{G}_v$ comme un générateur de puzzle classique (notons qu'il est crucial que l'information de contrôle soit générée par $\mathcal{G}_v$) auquel on peut appliquer le lemme 5.1 : il existe un algorithme de résolution $\mathcal{S}''$ résolvant le puzzle (p_f, p_v) avec une probabilité supérieure à $\epsilon + \frac{\delta}{16m}$, auquel $\mathcal{S}'$ fait appel pour calculer sa réponse. Dans le cas contraire, $\mathcal{S}'$ se contente de générer une réponse aléatoire, qui est correcte avec probabilité ϵ par hypothèse. La probabilité de succès totale de $\mathcal{S}'$ est donc supérieure à :

$$\frac{\delta}{2}\left(\epsilon + \frac{\delta}{16m}\right) + (1 - \frac{\delta}{2})\epsilon = \epsilon + \frac{\delta^2}{32m} \ .$$

Pour conclure, remarquons que le temps de calcul de $\mathcal{S}'$ est dominé par celui de $\mathcal{S}''$ et qu'il est donc polynomial en m, $\frac{1}{\delta}$, $\frac{1}{\epsilon^m + \delta}$, et les temps de calcul de $\mathcal{S}$, $\mathcal{G}$ et $\mathcal{V}$. ■

Munis du lemme 5.3, nous sommes en mesure de prouver la difficulté du puzzle MHB.

Théorème 5.4. *Supposons que le problème LPN est dur. Alors le puzzle MHB d'ordre m est $\left(1 - \frac{1}{2^m}\right)$-dur.* ♢

Démonstration. Raisonnons par contraposée en supposant l'existence d'un algorithme efficace $\mathcal{S}$ résolvant le puzzle MHB avec une probabilité supérieure à $\frac{1}{2^m} + \delta$, où δ est une fonction notable de k. D'après la définition de la pseudo-répétition d'un puzzle, on voit que le puzzle MHB est la pseudo-répétition à l'ordre m du puzzle HB, où $\mathcal{G}_f$ génère les vecteurs $(\boldsymbol{a}_1, \ldots, \boldsymbol{a}_q)$ et le challenge $\boldsymbol{a}$, et $\mathcal{G}_v$ génère le vecteur secret $\boldsymbol{x}$ et les bits de bruit $\nu_1, \ldots, \nu_q$. D'après le lemme 5.3, il existe donc un algorithme de résolution efficace du puzzle HB ayant une probabilité de succès supérieure à $\frac{1}{2} + \delta'$, où $\delta' = \frac{\delta^2}{32m}$ est une fonction notable de k. Par conséquent le puzzle HB n'est pas $\left(1 - \frac{1}{2}\right)$-dur, ce qui d'après le lemme 5.2 contredit la difficulté du problème LPN. ■

1. Notons que cette hypothèse est vraie pour le puzzle HB, avec $\epsilon = 1/2$.

2. D'après les bornes de Chernoff, $\mathcal{S}'$ peut évaluer la probabilité de succès de $\mathcal{S}$ pour p_f à δ près avec un nombre polynomial en $\frac{1}{\delta}$ d'exécutions de $\mathcal{S}$.

5.3 Sécurité de RANDOM-HB# dans le modèle actif

Nous allons maintenant utiliser le théorème 5.4 afin de démontrer, dans un premier temps, la sécurité de RANDOM-HB# contre les attaques actives. Ce résultat est exprimé par le théorème suivant et son corollaire.

Théorème 5.5. *Supposons qu'il existe un adversaire $\mathcal{A}$ attaquant le protocole* RANDOM-HB# *avec paramètres (k_X, k_Y, m, η, u) dans le modèle actif, interagissant avec le prouveur au plus q fois, dont le temps de calcul est inférieur à T, et possédant un avantage supérieur à δ. Alors il existe un algorithme de résolution $\mathcal{S}$ pour le puzzle MHB de paramètres (k_Y, m, η, q'), de temps de calcul inférieur à $T' = 2mLq(2 + \log q)T$, où*

$$q' = mLq(2 + \log q) \quad et \quad L = \mathcal{O}\left(\frac{512}{\delta^4(1-2u)^4}\log m\right) \ ,$$

et dont la probabilité de succès est supérieure à $\left(\frac{1}{2^m} + \frac{\delta}{4}\right)$. ♢

Corollaire 5.6. *Supposons le problème LPN dur. Alors le protocole* RANDOM-HB# *est sûr dans le modèle actif.* ▽

Démonstration. Notre preuve est une adaptation de la preuve de sécurité de Juels et Weis pour HB$^+$ [JW05b, Appendix C] au cas matriciel qui nous intéresse ici. Décrivons tout d'abord comment procède l'algorithme de résolution $\mathcal{S}$. Soit Y la matrice secrète utilisée par le générateur de puzzle. Nous noterons $\{(\boldsymbol{b}_i, \boldsymbol{z}_i)\}_{1\leq i\leq q'}$ les données du puzzle MHB que doit résoudre $\mathcal{S}$, et $\boldsymbol{b}$ le challenge pour lequel $\mathcal{S}$ cherche à retourner $\boldsymbol{z} = \boldsymbol{b} \cdot Y$. $\mathcal{S}$ utilise les données du puzzle pour simuler à l'adversaire $\mathcal{A}$ l'algorithme d'un prouveur $\mathcal{P}_{X,Y,\eta}$, où X est une matrice uniformément aléatoire « incorporant » le vecteur $\boldsymbol{z}$ dans l'une de ses lignes. Plus précisément, $\mathcal{S}$ procède comme suit.

Il divise les $q' = mLq(1+r)$ paires $\{(\boldsymbol{b}_i, \boldsymbol{z}_i)\}_{1\leq i\leq q'}$ en mL ensembles de $q(1+r)$ paires. Pour chaque position de bit $s = 1$ à m, il répète la procédure suivante L fois, en utilisant à chaque fois un nouveau jeu de $q(1+r)$ paires $\{(\boldsymbol{b}_i, \boldsymbol{z}_i)\}$:

1. Tirer une matrice aléatoire X' de taille $k_X \times m$ et j aléatoire dans $[\![1, k_X]\!]$. Par la suite nous noterons $X_{\boldsymbol{z},j}$ la matrice dont la j-ième ligne est égale au vecteur (inconnu) $\boldsymbol{z}$ et toutes les autres lignes sont nulles, et nous poserons $X = X' \oplus X_{\boldsymbol{z},j}$.
2. Simuler le prouveur lors des q interactions de l'adversaire $\mathcal{A}$ avec $\mathcal{P}$, pour $i = 1$ à q, de la façon suivante : tirer un bit aléatoire α_i (qui constitue un pari quant au j-ième bit du challenge $\boldsymbol{a}_i^+$ qui va être envoyé par l'adversaire $\mathcal{A}$). Si $\alpha_i = 0$, envoyer à $\mathcal{A}$ le vecteur de masquage $\boldsymbol{b}_i^+ = \boldsymbol{b}_i$, si $\alpha_i = 1$, envoyer à $\mathcal{A}$ le vecteur $\boldsymbol{b}_i^+ = \boldsymbol{b}_i \oplus \boldsymbol{b}$. $\mathcal{A}$ renvoie alors un challenge $\boldsymbol{a}_i^+$. Si le pari était bon, *i.e.* $\alpha_i = \boldsymbol{a}_i^+[j]$, répondre par le vecteur
$$\boldsymbol{z}_i^+ = \boldsymbol{a}_i^+ \cdot X' \oplus \boldsymbol{z}_i \ .$$
Sinon rembobiner l'adversaire $\mathcal{A}$ au début de sa i-ième requête et répéter la procédure avec une nouvelle paire $(\boldsymbol{b}_{i'}, \boldsymbol{z}_{i'})$ choisie parmi les rq paires supplémentaires.
3. Si les rq paires sont épuisées avant la fin de la phase de simulation des requêtes de $\mathcal{A}$, faire un vote aléatoire pour $\boldsymbol{z}[s]$ et terminer la procédure.
4. Sinon, appeler la phase de clonage de $\mathcal{A}$: $\mathcal{A}$ envoie un vecteur de masquage $\hat{\boldsymbol{b}}$. Tirer deux vecteurs challenge $\hat{\boldsymbol{a}}_1$ et $\hat{\boldsymbol{a}}_2$ aléatoires mais différant en leur j-ième bit. Transmettre $\hat{\boldsymbol{a}}_1$ à $\mathcal{A}$, enregistrer sa réponse $\hat{\boldsymbol{z}}_1$, puis rembobiner l'adversaire, transmettre $\hat{\boldsymbol{a}}_2$ à $\mathcal{A}$, et enregistrer sa réponse $\hat{\boldsymbol{z}}_2$.

5. Calculer le vote pour $\boldsymbol{z}[s]$ par :

$$v = \hat{\boldsymbol{z}}_1[s] \oplus \hat{\boldsymbol{z}}_2[s] \oplus \left((\hat{\boldsymbol{a}}_1 \oplus \hat{\boldsymbol{a}}_2) \cdot X'\right)[s] \ .$$

Une fois que L votes ont été faits pour chacun des m bits de $\boldsymbol{z}$, $\mathcal{S}$ prend le vote majoritaire pour chacun d'eux et retourne la réponse $\boldsymbol{z}$ correspondante.

Analysons maintenant la probabilité de succès de $\mathcal{S}$. Les mL appels répétés à $\mathcal{A}$ partagent une certaine part d'aléa, à savoir la matrice Y. Le reste de l'aléa est renouvelé à chaque appel et comprend les éléments suivants, que nous noterons ω :

- la matrice X (qui est uniformément aléatoire car X' est choisie aléatoirement à chaque nouvel appel),
- l'aléa utilisé pour simuler le prouveur et provenant du générateur de puzzle,
- l'aléa interne de $\mathcal{A}$,

ainsi que l'entier j et les challenges $\hat{\boldsymbol{a}}_1$ et $\hat{\boldsymbol{a}}_2$.

Par hypothèse, la probabilité, prise sur Y, ω, et un challenge $\hat{\boldsymbol{a}}$, pour que la réponse renvoyée par $\mathcal{A}$ soit correcte en au moins $m - t$ positions, est supérieure à $P_{\mathrm{FA}} + \delta$ (c'est exactement la probabilité que l'adversaire s'authentifie auprès du vérifieur). D'après le lemme B.1, pour une fraction des matrices Y supérieure à $P_{\mathrm{FA}} + \frac{\delta}{2}$, la probabilité, prise sur ω et $\hat{\boldsymbol{a}}$, que la réponse renvoyée par $\mathcal{A}$ soit correcte en au moins $m - t$ positions, est supérieure à $\frac{\delta}{2}$. Supposons donc que la matrice Y vérifie cette inégalité et montrons alors que $\mathcal{S}$ retourne le bon vecteur $\boldsymbol{z}$ en réponse au puzzle MHB avec une probabilité supérieure à $\frac{1}{2}$. Le théorème s'ensuivra car $P_{\mathrm{FA}} > \frac{2}{2^m}$ dès que $t > 1$, si bien que la probabilité de succès de $\mathcal{S}$ sera supérieure à

$$\frac{P_{\mathrm{FA}}}{2} + \frac{\delta}{4} > \frac{1}{2^m} + \frac{\delta}{4} \ .$$

Tout d'abord, nous allons montrer que durant la phase 2 de la procédure de simulation, $\mathcal{S}$ simule l'algorithme d'un prouveur $\mathcal{P}_{X,Y,\eta}$, avec $X = X' \oplus X_{\boldsymbol{z},j}$. En effet, observons que lorsque $\alpha_i = \boldsymbol{a}_i^+[j] = 0$, alors $\boldsymbol{a}_i^+ \cdot X_{\boldsymbol{z},j} = \boldsymbol{0}$, si bien que

$$\begin{aligned} \boldsymbol{z}_i^+ &= \boldsymbol{a}_i^+ \cdot X' \oplus \boldsymbol{b}_i \cdot Y \oplus \boldsymbol{\nu}_i \\ &= \boldsymbol{a}_i^+ \cdot X \oplus \boldsymbol{b}_i^+ \cdot Y \oplus \boldsymbol{\nu}_i \ , \end{aligned}$$

tandis que lorsque $\alpha_i = \boldsymbol{a}_i^+[j] = 1$, on a $\boldsymbol{a}_i^+ \cdot X_{\boldsymbol{z},j} = \boldsymbol{z} = \boldsymbol{b} \cdot Y$, si bien que

$$\begin{aligned} \boldsymbol{z}_i^+ &= \boldsymbol{a}_i^+ \cdot X' \oplus \boldsymbol{a}_i^+ \cdot X_{\boldsymbol{z},j} \oplus \boldsymbol{b} \cdot Y \oplus \boldsymbol{b}_i \cdot Y \oplus \boldsymbol{\nu}_i \\ &= \boldsymbol{a}_i^+ \cdot (X' \oplus X_{\boldsymbol{z},j}) \oplus (\boldsymbol{b}_i \oplus \boldsymbol{b}) \cdot Y \oplus \boldsymbol{\nu}_i \\ &= \boldsymbol{a}_i^+ \cdot X \oplus \boldsymbol{b}_i^+ \cdot Y \oplus \boldsymbol{\nu}_i \ . \end{aligned}$$

Dans tous les cas, les réponses reçues par $\mathcal{A}$ sont bien celles d'un prouveur $\mathcal{P}_{X,Y,\eta}$.

Analysons maintenant l'avantage dont bénéficie $\mathcal{S}$ lorsqu'il fait un vote pour un bit de $\boldsymbol{z}$. Tout d'abord, on peut majorer la probabilité que $\mathcal{S}$ entre dans la phase 3 de la procédure par la probabilité que l'une des q simulations de la phase 2 entraine le rejet de r paires $(\boldsymbol{b}_i, \boldsymbol{z}_i)$ parmi les rq paires supplémentaires, qui vaut $q2^{-r}$. Ainsi, en prenant $r = \log q + 1$ on obtient une probabilité inférieure à $\frac{1}{2}$.

Considérons la phase 4 pour une position de bit s fixée. Soit e_1 (respectivement e_2) le bit d'erreur en position s de la réponse de l'adversaire $\hat{\boldsymbol{z}}_1$ (respectivement $\hat{\boldsymbol{z}}_2$), c'est-à-dire :

$$\begin{aligned} \hat{\boldsymbol{z}}_1[s] &= (\hat{\boldsymbol{a}}_1 \cdot X)[s] \oplus (\hat{\boldsymbol{b}} \cdot Y)[s] \oplus e_1 \\ \hat{\boldsymbol{z}}_2[s] &= (\hat{\boldsymbol{a}}_2 \cdot X)[s] \oplus (\hat{\boldsymbol{b}} \cdot Y)[s] \oplus e_2 \ . \end{aligned}$$

En tenant compte du fait que $\hat{\boldsymbol{a}}_1[j] \oplus \hat{\boldsymbol{a}}_2[j] = 1$, le vote de $\mathcal{S}$ peut être réécrit comme :

$$\begin{aligned} v &= (\hat{\boldsymbol{a}}_1 \cdot X)[s] \oplus (\hat{\boldsymbol{b}} \cdot Y)[s] \oplus e_1 \oplus (\hat{\boldsymbol{a}}_2 \cdot X)[s] \oplus (\hat{\boldsymbol{b}} \cdot Y)[s] \oplus e_2 \oplus (\hat{\boldsymbol{a}}_1 \cdot X')[s] \oplus (\hat{\boldsymbol{a}}_2 \cdot X')[s] \\ &= ((\hat{\boldsymbol{a}}_1 \oplus \hat{\boldsymbol{a}}_2) \cdot (X \oplus X'))[s] \oplus e_1 \oplus e_2 \\ &= \boldsymbol{z}[s] \oplus e_1 \oplus e_2 \ . \end{aligned}$$

On voit donc que le vote de $\mathcal{S}$ est correct lorsque les bits $\hat{\boldsymbol{z}}_1[s]$ et $\hat{\boldsymbol{z}}_2[s]$ sont tous deux corrects, ou tous deux incorrects. Nous allons donc minorer la probabilité p' de cet événement. Pour ce faire, nous allons tout d'abord minorer la probabilité p, prise sur ω et le challenge $\hat{\boldsymbol{a}}$, pour que le s-ième bit de la réponse de $\mathcal{A}$ à ce challenge soit correct. Nous supposerons que cette probabilité est la même pour toutes les positions, hypothèse justifiée par le lemme suivant :

Lemme 5.7. *Tout adversaire $\mathcal{A}$ dans le modèle actif contre le protocole* RANDOM-*HB#* *peut être transformé en un adversaire $\mathcal{A}'$, possédant le même avantage, dont le temps de calcul est comparable, et tel que pour tout $s \in [\![1, m]\!]$, la probabilité que le bit en position s de la réponse de $\mathcal{A}'$ soit correct est égale pour toutes les positions.* ▽

Preuve. Supposons que $\mathcal{A}'$ interagisse avec un prouveur $\mathcal{P}_{X,Y,\eta}$. Soit σ une permutation de $[\![1, m]\!]$. Pour un vecteur $\boldsymbol{z}$ de longueur m (respectivement une matrice X à m colonnes), nous noterons $Q_\sigma(\boldsymbol{z})$ (respectivement $Q_\sigma(X)$) le vecteur obtenu en appliquant la permutation σ aux bits de $\boldsymbol{z}$ (respectivement aux colonnes de X). Remarquons que $Q_\sigma(\boldsymbol{a} \cdot X) = \boldsymbol{a} \cdot Q_\sigma(X)$. $\mathcal{A}'$ tire une permutation de l'ensemble $[\![1, m]\!]$ aléatoire σ et modifie les interactions entre le prouveur et l'adversaire $\mathcal{A}$ en changeant les réponses du prouveur $\boldsymbol{z}_i$ en $Q_\sigma(\boldsymbol{z}_i)$. Du point de vue de l'adversaire $\mathcal{A}$, tout se passe comme s'il interagissait avec un prouveur $\mathcal{P}_{Q_\sigma(X),Q_\sigma(Y),\eta}$. En effet :

$$\begin{aligned} Q_\sigma(\boldsymbol{z}_i) &= Q_\sigma\left(\boldsymbol{a}_i \cdot X \oplus \boldsymbol{b}_i \cdot Y \oplus \boldsymbol{\nu}_i\right) \\ &= \boldsymbol{a}_i \cdot Q_\sigma(X) \oplus \boldsymbol{b}_i \cdot Q_\sigma(Y) \oplus Q_\sigma(\boldsymbol{\nu}_i) \ . \end{aligned}$$

Le vecteur $Q_\sigma(\boldsymbol{\nu}_i)$ étant distribué selon $\mathtt{Ber}_{m,\eta}$, les réponses reçues par $\mathcal{A}$ sont bien celles d'un prouveur $\mathcal{P}_{Q_\sigma(X),Q_\sigma(Y),\eta}$.

Puis lors de la phase de clonage de $\mathcal{A}$, $\mathcal{A}'$ modifie la réponse $\hat{\boldsymbol{z}}$ renvoyée par $\mathcal{A}$ en $Q_{\sigma^{-1}}(\hat{\boldsymbol{z}})$. La probabilité de succès de $\mathcal{A}'$ et la même que celle de $\mathcal{A}$ car :

$$\mathtt{Hwt}\left(\hat{\boldsymbol{a}} \cdot X \oplus \hat{\boldsymbol{b}} \cdot Y \oplus Q_{\sigma^{-1}}(\hat{\boldsymbol{z}})\right) = \mathtt{Hwt}\left(\hat{\boldsymbol{a}} \cdot Q_\sigma(X) \oplus \hat{\boldsymbol{b}} \cdot Q_\sigma(Y) \oplus \hat{\boldsymbol{z}}\right) \ .$$

Soit p_i la probabilité (prise sur X, Y, et l'aléa interne du prouveur, de $\mathcal{A}$ et du vérifieur) que le bit en position i de la réponse $\hat{\boldsymbol{z}}$ de $\mathcal{A}$ soit correct. Clairement, le bit en position i de la réponse $Q_{\sigma^{-1}}(\hat{\boldsymbol{z}})$ de $\mathcal{A}'$ est correct si et seulement si le bit en position $\sigma(i)$ de la réponse de $\mathcal{A}$ est correct. Or la probabilité prise sur σ que $\sigma(i) = j$ est égale à $\frac{1}{m}$ pour tout i et j. Ainsi, pour tout i, la probabilité que le bit en position i de la réponse de $\mathcal{A}'$ soit correcte est égale à $\frac{1}{m}\sum_{j=1}^{m} p_j$. ▲

Nous pouvons alors minorer p comme suit. Supposons qu'une fois la réponse de $\mathcal{A}$ reçue, on tire une position *aléatoire* $s \in [\![1, m]\!]$. Clairement, le bit en position s de la réponse de $\mathcal{A}$ est correct avec probabilité p sur ω, $\hat{\boldsymbol{a}}$ et le tirage de s. Par ailleurs, on peut aussi calculer cette probabilité de la façon suivante : conditionné par le fait que plus de $m - t$ bits sont corrects (ce qui arrive avec un probabilité supérieure à $\frac{\delta}{2}$ par hypothèse), le s-ième bit de la réponse, pour un s aléatoire, est correct avec une probabilité supérieure à $(m-t)/m = 1-u$. Dans le cas contraire, le s-ième bit de la réponse est correct avec une

probabilité supérieure à $\frac{1}{2}$. Par conséquent, la probabilité p que le s-ième bit de la réponse soit correct est supérieure à

$$(1-u)\frac{\delta}{2} + \frac{1}{2}\left(1-\frac{\delta}{2}\right) = \frac{1}{2} + \frac{\delta}{2}\left(\frac{1}{2}-u\right) \ .$$

Donc $p \geq \frac{1}{2} + \epsilon$ avec $\epsilon = \frac{\delta}{2}\left(\frac{1}{2}-u\right)$.

Si les vecteurs $\hat{\boldsymbol{a}}_1$ et $\hat{\boldsymbol{a}}_2$ étaient choisis indépendamment, la probabilité p' que les bits en position s des réponses $\hat{\boldsymbol{z}}_1$ et $\hat{\boldsymbol{z}}_2$ soient à la fois corrects ou incorrects serait

$$\left(\frac{1}{2}+\epsilon\right)^2 + \left(\frac{1}{2}-\epsilon\right)^2 = \frac{1}{2} + 2\epsilon^2 \ .$$

Cependant ces deux vecteurs ne sont pas indépendants : en effet ils sont choisis tels qu'ils diffèrent en leur j-ième bit, pour un j aléatoirement choisi dans $[\![1, k_X]\!]$. Mais comme nous allons le voir, l'effet de cette dépendance est minime, c'est-à-dire que p' peut être minoré par une valeur proche de $\frac{1}{2} + 2\epsilon^2$. Plus précisément :

Lemme 5.8. *La probabilité p' que les bits en position s des réponses $\hat{\boldsymbol{z}}_1$ et $\hat{\boldsymbol{z}}_2$ soient à la fois corrects ou incorrects, conditionnée par le fait que $\hat{\boldsymbol{a}}_1$ et $\hat{\boldsymbol{a}}_2$ diffèrent en leur j-ième bit, pour un j aléatoirement choisi dans $[\![1, k_X]\!]$, est telle que*

$$p' \geq \frac{1}{2} + 2\epsilon^2 - \frac{1}{k_X} \ . \qquad \triangledown$$

Preuve. Tout d'abord remarquons que l'adversaire n'a aucun moyen de connaître j au cours de la simulation, par conséquent on peut considérer que j est tiré aléatoirement après que $\mathcal{A}$ a renvoyé ses deux réponses. Notons C_i l'événement que le bit $\hat{\boldsymbol{z}}_i[s]$ soit correct. Formellement nous voulons donc minorer la probabilité

$$p' = \Pr\left[\omega, \hat{\boldsymbol{a}}_1, \hat{\boldsymbol{a}}_2 \xleftarrow{\$} \{0,1\}^{k_X}, j \xleftarrow{\$} [\![1, k_X]\!] \,:\, (C_1 \wedge C_2) \vee (\bar{C_1} \wedge \bar{C_2}) \,|\, \hat{\boldsymbol{a}}_1[j] \neq \hat{\boldsymbol{a}}_2[j]\right] \ .$$

Pour un ω fixé, notons p_ω la probabilité, prise sur $\hat{\boldsymbol{a}}$, que le bit en position s de la réponse de $\mathcal{A}$ soit correct. Nous avons montré précédemment que $p = \mathbb{E}_\omega[p_\omega] \geq \frac{1}{2} + \epsilon$. De même, soit p'_ω la probabilité, pour un ω fixé, et prise sur le tirage de j, $\hat{\boldsymbol{a}}_1$, et $\hat{\boldsymbol{a}}_2$, que les bits en position s des réponses $\hat{\boldsymbol{z}}_1$ et $\hat{\boldsymbol{z}}_2$ soient à la fois corrects ou incorrects, conditionnée par $\hat{\boldsymbol{a}}_1[j] \neq \hat{\boldsymbol{a}}_2[j]$. On a également $p' = \mathbb{E}_\omega[p'_\omega]$. Nous allons minorer p'_ω en fonction de p_ω. Pour cela, nous remarquons que les événements $(C_1 \wedge C_2)$ et $(\bar{C_1} \wedge \bar{C_2})$ étant incompatibles, on a :

$$\begin{aligned}
p'_\omega &= \Pr\left[(C_1 \wedge C_2) \,|\, \hat{\boldsymbol{a}}_1[j] \neq \hat{\boldsymbol{a}}_2[j]\right] + \Pr\left[(\bar{C_1} \wedge \bar{C_2}) \,|\, \hat{\boldsymbol{a}}_1[j] \neq \hat{\boldsymbol{a}}_2[j]\right] \\
&= \frac{\Pr\left[\hat{\boldsymbol{a}}_1[j] \neq \hat{\boldsymbol{a}}_2[j] \,|\, (C_1 \wedge C_2)\right] \cdot \Pr\left[C_1 \wedge C_2\right]}{\Pr\left[\hat{\boldsymbol{a}}_1[j] \neq \hat{\boldsymbol{a}}_2[j]\right]} \\
&\qquad + \frac{\Pr\left[\hat{\boldsymbol{a}}_1[j] \neq \hat{\boldsymbol{a}}_2[j] \,|\, (\bar{C_1} \wedge \bar{C_2})\right] \cdot \Pr\left[\bar{C_1} \wedge \bar{C_2}\right]}{\Pr\left[\hat{\boldsymbol{a}}_1[j] \neq \hat{\boldsymbol{a}}_2[j]\right]} \\
&= 2p_\omega^2 \Pr\left[\hat{\boldsymbol{a}}_1[j] \neq \hat{\boldsymbol{a}}_2[j] \,|\, (C_1 \wedge C_2)\right] + 2(1-p_\omega)^2 \Pr\left[\hat{\boldsymbol{a}}_1[j] \neq \hat{\boldsymbol{a}}_2[j] \,|\, (\bar{C_1} \wedge \bar{C_2})\right] \ .
\end{aligned}$$

Il reste alors à minorer $\Pr\left[\hat{\boldsymbol{a}}_1[j] \neq \hat{\boldsymbol{a}}_2[j] \,|\, (C_1 \wedge C_2)\right]$ et $\Pr\left[\hat{\boldsymbol{a}}_1[j] \neq \hat{\boldsymbol{a}}_2[j] \,|\, (\bar{C_1} \wedge \bar{C_2})\right]$. Pour cela on peut remarquer qu'étant donné un ensemble A de vecteurs de $\{0,1\}^{k_X}$, la

probabilité, lorsqu'on tire deux vecteurs $\hat{\boldsymbol{a}}_1$ et $\hat{\boldsymbol{a}}_2$ de cet ensemble et un $j \xleftarrow{\$} [\![1, k_X]\!]$, que $\hat{\boldsymbol{a}}_1[j] = \hat{\boldsymbol{a}}_2[j]$, est maximisée lorsque A contient $\lfloor \log |A| \rfloor$ vecteurs dont les $k_X - \lfloor \log |A| \rfloor$ premiers bits sont égaux, et on peut donc majorer cette probabilité par $1 - \frac{\lceil \log |A| \rceil}{2k_X}$. Par conséquent, en prenant respectivement $|A| = 2^{k_X} p_\omega$ et $|A| = 2^{k_X}(1 - p_\omega)$, on obtient :

$$\Pr\left[\hat{\boldsymbol{a}}_1[j] \neq \hat{\boldsymbol{a}}_2[j] \,|\, (C_1 \wedge C_2)\right] \geq \frac{k_X + \log p_\omega - 1}{2k_X}$$
$$\Pr\left[\hat{\boldsymbol{a}}_1[j] \neq \hat{\boldsymbol{a}}_2[j] \,|\, (\bar{C_1} \wedge \bar{C_2})\right] \geq \frac{k_X + \log(1 - p_\omega) - 1}{2k_X} \quad .$$

Par conséquent, on a $p'_\omega \geq \phi(p_\omega)$ où ϕ est donnée par :

$$\phi(x) = x^2 \left(\frac{k_X + \log x - 1}{k_X}\right) + (1 - x)^2 \left(\frac{k_X + \log(1 - x) - 1}{k_X}\right) \quad .$$

Comme ϕ est convexe, on a, d'après l'inégalité de Jensen rappelée dans l'appendice B :

$$p' = \mathbb{E}_\omega[p'_\omega] \geq \mathbb{E}_\omega[\phi(p_\omega)] \geq \phi\left(\mathbb{E}_\omega[p_\omega]\right) = \phi(p) \geq \phi\left(\frac{1}{2} + \epsilon\right) \geq \frac{1}{2} + 2\epsilon^2 - \frac{1}{k_X} \quad ,$$

la dernière inégalité venant de $\inf_{x \in]0,1[} \left(x^2(\log x - 1) + (1 - x)^2(\log(1 - x) - 1)\right) = -1$. ▲

Comme $\mathcal{S}$ n'entre en phase 3 de la procédure de vote qu'avec une probabilité d'au plus $\frac{1}{2}$ (auquel cas son vote est aléatoire), la probabilité globale pour que le vote de $\mathcal{S}$ soit correct est minorée par

$$\frac{1}{4} + \frac{p'}{2} \geq \frac{1}{2} + \epsilon', \text{ avec } \epsilon' = \epsilon^2 - \frac{1}{2k_X} = \frac{\delta^2}{4}\left(\frac{1}{2} - u\right)^2 - \frac{1}{2k_X} \quad .$$

D'après les bornes de Chernoff (cf. appendice B), en répétant l'expérience L fois et en prenant le vote majoritaire, l'algorithme $\mathcal{S}$ devine le bit s correctement avec une probabilité supérieure à

$$\pi = \left(1 - e^{\frac{-L\epsilon'^2}{1+2\epsilon'}}\right) \geq \left(1 - e^{\frac{-L\epsilon'^2}{2}}\right) \quad .$$

Les m bits de la réponse de $\mathcal{S}$ seront donc corrects avec une probabilité supérieure à

$$\pi^m \geq \left(1 - e^{\frac{-L\epsilon'^2}{2}}\right)^m \quad .$$

Une probabilité de succès supérieure à $\frac{1}{2}$ est obtenue en prenant

$$L = \frac{2}{\epsilon'^2} \ln\left(\frac{1}{1 - e^{-\frac{\ln 2}{m}}}\right) = \mathcal{O}\left(\frac{512}{\delta^4(1 - 2u)^4} \log m\right) \quad .$$

La preuve du théorème s'ensuit en remarquant que le nombre total de paires $(\boldsymbol{b}_i, \boldsymbol{z}_i)$ utilisées par $\mathcal{S}$ est inférieur à $q' = mLqr = mLq(2 + \log q)$, et son temps de calcul, en tenant compte du fait que le nombre total de rembobinages au cours d'un vote est inférieur à $2qr$, est inférieur à $T' = 2mLqrT$.

La démonstration du corollaire est immédiate : tout adversaire efficace possédant un avantage notable contre le protocole RANDOM-HB# implique l'existence d'un algorithme de résolution efficace du puzzle MHB ayant une probabilité de succès supérieure à $\frac{1}{2^m} + \delta'$, où $\delta' = \delta/4$ est notable. D'après le théorème 5.4, ceci contredit l'hypothèse que le problème LPN est dur. ■

Remarque 5.1. Récemment, Impagliazzo *et al.* ont obtenu un résultat plus fort que celui de Canetti *et al.* concernant l'amplification de difficulté des puzzles faiblement vérifiables [IJK07]. Il est très certainement possible d'améliorer la qualité de la réduction de notre démonstration à l'aide de ce résultat. ∗

5.4 Sécurité de RANDOM-HB# dans le modèle GRS

La seconde étape de notre preuve de sécurité consiste à réduire la sécurité du protocole RANDOM-HB# dans le modèle GRS à sa sécurité dans le modèle actif. Plus précisément, on a le résultat suivant.

Théorème 5.9. *Soient (k_X, k_Y, m, η, u) des paramètres du protocole* RANDOM-*HB#* *vérifiant la condition :*

$$\exists\, c_1, c_2 > 0 \text{ tels que } 1 - \frac{k_X}{m} + H_2\left(\frac{d}{m}\right) \geq c_2 \tag{5.1}$$

où H_2 est la fonction entropie $H_2(x) = -x\log(x) - (1-x)\log(1-x)$ et

$$d = 1 + \left\lfloor \frac{(1+c_1)um - \eta m}{1-2\eta} \right\rfloor \quad .$$

Supposons qu'il existe un adversaire $\mathcal{A}$ attaquant le protocole RANDOM-*HB#* *dans le modèle GRS, perturbant au plus q exécutions du protocole, dont le temps de calcul est inférieur à T, et possédant un avantage supérieur à δ. Alors il existe un adversaire $\mathcal{A}'$ attaquant le protocole* RANDOM-*HB#* *dans le modèle actif, interagissant avec le prouveur au plus q fois, dont le temps de calcul est inférieur à $\mathcal{O}(T)$, et possédant un avantage supérieur à $(P_{FA} + \delta)(1 - q\epsilon)$, où $\epsilon = \mathtt{negl}(k)$.* ♢

Corollaire 5.10. *Supposons le problème LPN dur. Alors le protocole* RANDOM-*HB#* *avec des paramètres vérifiant la condition (5.1) est sûr dans le modèle GRS.* ▽

Démonstration. Nous allons décrire comment l'adversaire $\mathcal{A}'$ utilise l'adversaire $\mathcal{A}$ pour réaliser son attaque en lui simulant le prouveur et le vérifieur. Comme $\mathcal{A}'$ a accès à un prouveur authentique auquel il peut faire des requêtes, il est très simple de simuler le prouveur : $\mathcal{A}'$ transmet simplement les requêtes de $\mathcal{A}$ au prouveur et renvoie la réponse du prouveur à $\mathcal{A}$. Au contraire, $\mathcal{A}'$, qui est un adversaire dans le modèle actif, n'a pas accès au vérifieur et sa simulation est donc moins aisée. Nous allons voir cependant comment y parvenir.

Rappelons que dans le modèle GRS l'adversaire n'est autorisé à modifier que les messages du vérifieur vers le prouveur. $\mathcal{A}'$ commence donc par lancer la première phase de $\mathcal{A}$ et simule, pour $i = 1$ à q, le prouveur et le vérifieur de la façon suivante :

1. $\mathcal{A}'$ demande au prouveur $\mathcal{P}_{X,Y,\eta}$ un vecteur de masquage $\boldsymbol{b}_i$; $\mathcal{A}'$ envoie $\boldsymbol{b}_i$ comme vecteur de masquage du prouveur simulé au vérifieur simulé.
2. $\mathcal{A}'$ envoie un vecteur aléatoire $\boldsymbol{a}_i$ comme challenge du vérifieur simulé. $\mathcal{A}$ le modifie en $\boldsymbol{a}'_i = \boldsymbol{a}_i \oplus \boldsymbol{\alpha}_i$. $\mathcal{A}'$ transmet $\boldsymbol{a}'_i$ au prouveur authentique.
3. Le prouveur authentique renvoie une réponse $\boldsymbol{z}_i = \boldsymbol{a}'_i \cdot X \oplus \boldsymbol{b}_i \cdot Y \oplus \boldsymbol{\nu}_i$ à $\mathcal{A}'$ qui l'utilise comme réponse du prouveur simulé au vérifieur simulé.
4. $\mathcal{A}'$ simule la décision du vérifieur comme suit : si $\boldsymbol{\alpha}_i$ était le vecteur nul, $\mathcal{A}'$ accepte l'authentification, sinon il la rejette.

Après cette première phase, $\mathcal{A}'$ appelle la phase de clonage de $\mathcal{A}$ et réplique les messages de $\mathcal{A}$ au vérifieur réel.

Du point de vue de $\mathcal{A}$, le prouveur $\mathcal{P}_{X,Y,\eta}$ est parfaitement simulé par $\mathcal{A}'$. Soit $\mathtt{Sim}_i$ l'événement que le vérifieur $\mathcal{R}_{X,Y,u}$ soit correctement simulé par $\mathcal{A}$ lors de la i-ième exécution du protocole, et $\mathtt{Sim}$ l'événement que le vérifieur soit correctement simulé pour

chacune des q exécutions du protocole, $\texttt{Sim} = \bigwedge_{i=1}^{q} \texttt{Sim}_i$. La probabilité de succès conditionnelle de $\mathcal{A}'$ sachant l'événement $\texttt{Sim}$ réalisé est la même que celle de $\mathcal{A}$, *i.e.* $P_{\mathrm{FA}} + \delta$. Il nous faut donc maintenant minorer la probabilité de l'événement $\texttt{Sim}$.

Considérons la i-ième exécution du protocole. Lorsque $\boldsymbol{\alpha}_i = \mathbf{0}$, $\mathcal{A}'$ échoue à simuler correctement le vérifieur (c'est-à-dire qu'il accepte l'authentification au lieu de la rejeter) avec une probabilité égale à celle qu'a un vérifieur authentique de rejeter un prouveur authentique, *i.e.* P_{FR}. Pour le cas où $\boldsymbol{\alpha}_i \neq \mathbf{0}$ nous pouvons faire le raisonnement suivant. Supposons que le vecteur d'erreur $\boldsymbol{\alpha}_i \cdot X$ rajouté par $\mathcal{A}$ ait un poids de Hamming égal à d. Puisque ce vecteur est ajouté *avant* le bruit $\boldsymbol{\nu}_i$ ajouté par le prouveur, ce dernier est indépendant de $\boldsymbol{\alpha}_i \cdot X$. Par conséquent, le vecteur d'erreur total $\boldsymbol{\nu}_i \oplus \boldsymbol{\alpha}_i \cdot X$ a un poids de Hamming distribué comme la somme de d variables de Bernoulli égales à 1 avec probabilité $1 - \eta$ et 0 avec probabilité η, et $m - d$ variables de Bernoulli égales à 1 avec probabilité η et 0 avec probabilité $1 - \eta$. La valeur moyenne du poids de Hamming du vecteur d'erreur total est donc

$$\mu(d) = d(1-\eta) + (m-d)\eta = \eta m + (1-2\eta)d \ .$$

D'après les bornes de Chernoff (cf. appendice B), lorsque $\mu(d) > t$, le poids de Hamming du vecteur d'erreur total est inférieur à t avec une probabilité inférieure à $\exp(-\frac{(\mu-t)^2}{2\mu})$. De plus ceci reste vrai pour tout $d' \geq d$ car μ est une fonction croissante de d. Ainsi, si la matrice X est telle que pour tout $\boldsymbol{\alpha} \neq \mathbf{0}$, $\mathrm{Hwt}(\boldsymbol{\alpha} \cdot X)$ est suffisamment grand, rejeter l'authentification dès que $\boldsymbol{\alpha}_i \neq \mathbf{0}$ est une bonne stratégie de simulation. Nous formalisons ce raisonnement comme suit.

Soit $d_{\min}(X) = \min_{\boldsymbol{\alpha} \neq \mathbf{0}} (\mathrm{Hwt}(\boldsymbol{\alpha} \cdot X))$ la distance minimale de la matrice X. Rappelons tout d'abord le résultat classique de théorie des codes :

Lemme 5.11. *Soit d un entier compris dans $[\![1, \lfloor \frac{m}{2} \rfloor]\!]$ et soit H_2 la fonction entropie $H_2(x) = -x\log(x) - (1-x)\log(1-x)$. Alors*

$$\Pr_X[d_{\min}(X) \leq d] \leq 2^{-\left(1 - \frac{k_X}{m} - H_2\left(\frac{d}{m}\right)\right)m} \ . \qquad \triangledown$$

Preuve. C'est une simple conséquence de la borne supérieure suivante sur le nombre de vecteurs de longueur m de poids de Hamming inférieur à d (cf. appendice C) :

$$\sum_{i=0}^{d} \binom{m}{i} \leq 2^{mH_2\left(\frac{d}{m}\right)} \ .$$

Pour tout vecteur non nul $\boldsymbol{\alpha}$, $\boldsymbol{\alpha} \cdot X$ est uniformément distribué lorsque X est tirée aléatoirement, et par conséquent a un poids de Hamming inférieur à d avec une probabilité inférieure à $2^{m\left(H_2\left(\frac{d}{m}\right)-1\right)}$. Le lemme dérive alors de la borne de l'union. ▲

Soit $\tilde{d}$ le plus petit entier tel que

$$\mu(\tilde{d}) \geq t, \quad \textit{i.e.} \quad \tilde{d} = 1 + \left\lfloor \frac{t - \eta m}{1 - 2\eta} \right\rfloor \ .$$

On peut vérifier que $\tilde{d} < \frac{m}{2}$. Alors pour tout $d \in [\![\tilde{d}, \frac{m}{2}]\!]$, on a $\mu(d) \geq t$, si bien que lorsque

$\boldsymbol{\alpha}_i \neq \mathbf{0}$, on a :

$$\begin{aligned}\Pr_{X,\boldsymbol{\nu}_i}[\overline{\texttt{Sim}_i}] &= \Pr_{\boldsymbol{\nu}_i}[\overline{\texttt{Sim}_i} \mid d_{\min}(X) > d] \cdot \Pr_X[d_{\min}(X) > d] \\ &\qquad + \Pr_{\boldsymbol{\nu}_i}[\overline{\texttt{Sim}_i} \mid d_{\min}(X) \leq d] \cdot \Pr_X[\min(X) \leq d] \\ &\leq \Pr_{\boldsymbol{\nu}_i}[\overline{\texttt{Sim}_i} \mid d_{\min}(X) > d] + \Pr_X[d_{\min}(X) \leq d] \\ &\leq e^{-\frac{(\mu-t)^2}{2\mu}} + 2^{-\left(1-\frac{k_X}{m}-H_2(\frac{d}{m})\right)m} \ .\end{aligned}$$

On peut alors utiliser la condition (5.1) pour montrer que ce majorant est négligeable. En effet cette condition indique qu'on peut trouver deux constantes c_1 et c_2 telles qu'il existe un d (nécessairement supérieur à $\tilde{d}$) tel que

$$\mu(d) \geq (1+c_1)t \quad \text{et} \quad \left(1 - \frac{k_X}{m} - H_2\left(\frac{d}{m}\right)\right) \geq c_2 \ ,$$

ce qui en reportant dans le majorant précédant implique :

$$\Pr_{X,\boldsymbol{\nu}_i}[\overline{\texttt{Sim}_i}] \leq e^{-\frac{uc_1^2}{2(1+c_1)}m} + 2^{-c_2 m} \ ,$$

qui est bien négligeable en m, donc en k.

Ainsi on a que $\Pr[\overline{\texttt{Sim}_i}] \leq \epsilon$, où ϵ est une fonction négligeable du paramètre de sécurité définie par :

$$\epsilon = \max\left\{P_{\text{FR}}, \min_{d \in [\![\tilde{d}, \frac{m}{2}]\!]}\left(e^{-\frac{(\mu-t)^2}{2\mu}} + 2^{-\left(1-\frac{k_X}{m}-H_2\left(\frac{d}{m}\right)\right)m}\right)\right\} \ .$$

Par conséquent, $\Pr[\texttt{Sim}] \geq (1 - q\epsilon)$ si bien que $\mathcal{A}'$ a une probabilité de succès supérieure à $(P_{\text{FA}} + \delta)(1 - q\epsilon)$, ce qui conclut la démonstration du théorème.

La démonstration du corollaire s'en déduit en remarquant que si δ est notable, alors pour k suffisamment grand, $q\epsilon(P_{\text{FA}} + \delta) \leq \frac{\delta}{2}$, donc la probabilité de succès de $\mathcal{A}'$ est plus grande que $P_{\text{FA}} + \frac{\delta}{2}$. Or ceci contredit la sécurité de RANDOM-HB$^{\#}$ dans le modèle actif (corollaire 5.6). ∎

Remarque 5.2. La condition (5.1) est vérifiée pour des paramètres typiques pour lesquels m est grand devant k_X. Comme nous le verrons dans la section 5.7 on peut prendre $k_X = 80$. Des paramètres acceptables pour m et η (c'est-à-dire menant à des probabilités de fausse acceptation et de faux rejet négligeables) sont par exemple $m = 1\,164$, $\eta = 0,25$ et $m = 441$, $\eta = 0,125$. Dans le premier cas on a :

$$\tilde{d} = 229 \quad \text{et} \quad \left(1 - \frac{k_X}{m} - H_2\left(\frac{\tilde{d}}{m}\right)\right) = 0,216 \ ,$$

et dans le second :

$$\tilde{d} = 78 \quad \text{et} \quad \left(1 - \frac{k_X}{m} - H_2\left(\frac{\tilde{d}}{m}\right)\right) = 0,145 \ .$$

Dans les deux cas l'existence de constantes c_1 et c_2 permettant de majorer la probabilité d'échec de la simulation du vérifieur par une quantité négligeable est assurée. ∗

5.5 La proposition pratique HB$^{\#}$

RANDOM-HB$^{\#}$ constitue une avancée par rapport à HB^{+} sur deux points essentiels :
- alors que la sécurité de HB^{+} n'est prouvée que dans le modèle des attaques actives et qu'il existe une attaque GRS de complexité linéaire contre ce protocole, la sécurité de RANDOM-HB$^{\#}$ est prouvée dans le modèle des attaques GRS.
- le coût de communication est réduit de $r(k_X + k_Y + 1)$ bits pour HB^{+}, où r est le nombre de tours du protocole, à $(k_X + k_Y + m)$ bits pour RANDOM-HB$^{\#}$, où m est le nombre de colonnes des matrices X et Y. En fait r et m sont équivalents car ce sont ces deux paramètres qui, avec η, fixent les probabilités de fausse acceptation et de faux rejet pour chacun des protocoles. On voit donc qu'en supposant $r, m, k_X, k_Y = \Theta(k)$, le coût de communication est une fonction quadratique du paramètre de sécurité pour HB^{+} alors qu'il est linéaire pour RANDOM-HB$^{\#}$.

Cependant, un problème de taille apparaît : la clé secrète est constituée de deux matrices X, Y de tailles respectives $k_X \times m$ et $k_Y \times m$, soit un coût de stockage total de $m(k_X + k_Y)$ bits, fonction quadratique du paramètre de sécurité. Pour des valeurs typiques des paramètres, soit quelques centaines de bits, cela représente plusieurs dizaines de Kbits, ce qui exclut l'utilisation de RANDOM-HB$^{\#}$ pour des étiquettes RFID à bas-coût.

5.5.1 Utilisation de matrices de Toeplitz

Afin de remédier à cet inconvénient, nous proposons de modifier la forme des matrices X et Y. Au lieu d'utiliser des matrices uniformément aléatoires, nous proposons d'utiliser des matrices possédant une forme particulière, à savoir des matrices de Toeplitz.

Définition 5.6 (Matrice de Toeplitz)
Une matrice de Toeplitz de taille $k \times m$ est telle que tous ses coefficients situés sur une même anti-diagonale [3] sont égaux. Si $\boldsymbol{s} = (s_1, \ldots, s_{k+m-1})$ est un vecteur de taille $k+m-1$, nous noterons $\mathcal{T}_{\boldsymbol{s}} = (t_{ij})$ la matrice de Toeplitz définie par $t_{ij} = s_{i+j-1}$. Lorsque $m > k$, $\mathcal{T}_{\boldsymbol{s}}$ a la forme suivante :

$$\begin{pmatrix} s_1 & s_2 & s_3 & \cdots & s_{k-1} & s_k & s_{k+1} & \cdots & s_m \\ s_2 & s_3 & & \iddots & \iddots & \iddots & & \iddots & s_{m+1} \\ s_3 & & \iddots & \iddots & \iddots & & & \iddots & \vdots \\ \vdots & s_{k-1} & \iddots & \iddots & & & & & \\ s_{k-1} & s_k & s_{k+1} & & \iddots & \iddots & & & \vdots \\ s_k & s_{k+1} & \cdots & & s_m & s_{m+1} & & \cdots & s_{k+m-1} \end{pmatrix} \quad \blacklozenge$$

Le protocole HB$^{\#}$ est défini exactement comme RANDOM-HB$^{\#}$ à la différence près que les deux matrices X et Y ne sont plus uniformément aléatoires, mais choisies uniformément dans l'espace des matrices de Toeplitz : $X = \mathcal{T}_{\boldsymbol{x}}$ et $Y = \mathcal{T}_{\boldsymbol{y}}$, avec $\boldsymbol{x} \xleftarrow{\$} \{0,1\}^{k_X+m-1}$ et $\boldsymbol{y} \xleftarrow{\$} \{0,1\}^{k_Y+m-1}$. Par conséquent, la taille totale de la clé secrète pour HB$^{\#}$ est réduite à $(2m + k_X + k_Y - 2)$ bits, soit une fonction linéaire du paramètre de sécurité.

Les matrices de Toeplitz ont par exemple été utilisées par Krawczyk pour construire des codes d'authentification de messages [Kra94, Kra95] en raison de leurs propriétés

3. La définition classique est que tous les coefficients d'une même *diagonale* sont égaux, mais notre est définition est équivalente et simplifie nos notations.

statistiques. En effet, on peut montrer que la famille de fonctions

$$(\boldsymbol{a} \mapsto \boldsymbol{a} \cdot \mathcal{T}_{\boldsymbol{s}})_{\boldsymbol{s} \in \{0,1\}^{k+m-1}}$$

constitue une famille de fonctions de hachage universelle [CW79]. En d'autres termes, on a le résultat suivant, dont une démonstration peut être trouvée dans [MNT90] :

Lemme 5.12. *Soient $\boldsymbol{a} \neq \boldsymbol{0}$ et $\boldsymbol{z}$ deux vecteurs de longueurs respectives k et m. Alors*

$$\Pr\left[\boldsymbol{s} \xleftarrow{\$} \{0,1\}^{k+m-1} \,:\, \boldsymbol{a} \cdot \mathcal{T}_{\boldsymbol{s}} = \boldsymbol{z}\right] = \frac{1}{2^m} \ . \qquad \triangledown$$

Intuitivement, il s'agit de la propriété que l'on recherche pour assurer la sécurité du protocole : pour un adversaire qui n'a pas d'information sur les matrices X et Y, les vecteurs $\boldsymbol{a} \cdot X$ et $\boldsymbol{b} \cdot Y$ paraissent uniformément distribués.

Remarquons qu'il existe une façon alternative de décrire HB$^{\#}$. Pour un vecteur $\boldsymbol{a} = (a_1, \ldots, a_k) \in \{0,1\}^k$, nous noterons $M_{\boldsymbol{a}}$ la matrice de taille $m \times (m+k-1)$ définie par

$$M_{\boldsymbol{a}} = \begin{pmatrix} a_1 & a_2 & a_3 & \cdots & a_k & 0 & \cdots & \cdots & \cdots & \cdots & \cdots & 0 \\ 0 & a_1 & a_2 & \cdots & a_{k-1} & a_k & 0 & \cdots & \cdots & \cdots & \cdots & 0 \\ \vdots & \vdots & & & & \vdots & \vdots & & & & \vdots & \vdots \\ \vdots & \vdots & & & & \vdots & \vdots & & & & \vdots & \vdots \\ 0 & 0 & \cdots & \cdots & \cdots & \cdots & 0 & a_1 & a_2 & \cdots & \cdots & a_k \end{pmatrix} \qquad (5.2)$$

On peut alors vérifier que $\boldsymbol{a} \cdot \mathcal{T}_{\boldsymbol{x}} = (M_{\boldsymbol{a}} \cdot \boldsymbol{x})^t$. On peut donc réécrire la réponse d'un prouveur pour le protocole HB$^{\#}$ sous la forme

$$\boldsymbol{z} = \boldsymbol{a} \cdot \mathcal{T}_{\boldsymbol{x}} \oplus \boldsymbol{b} \cdot \mathcal{T}_{\boldsymbol{y}} \oplus \boldsymbol{\nu} = (M_{\boldsymbol{a}} \cdot \boldsymbol{x})^t \oplus (M_{\boldsymbol{b}} \cdot \boldsymbol{y})^t \oplus \boldsymbol{\nu} \ .$$

Par conséquent, le protocole HB$^{\#}$ peut être vu comme le protocole HB^{+}, avec deux secrets $\boldsymbol{x}$ et $\boldsymbol{y}$ de tailles respectives (k_X+m-1) et (k_Y+m-1), comportant m tours, mais où les m vecteurs de masquage $\boldsymbol{b}_i$ et les m vecteurs challenges $\boldsymbol{a}_i$ ne sont pas uniformément aléatoires mais ont une forme spéciale, à savoir $\boldsymbol{a}_i = \boldsymbol{a}^{\ggg i}$ et $\boldsymbol{b}_i = \boldsymbol{b}^{\ggg i}$, où $\boldsymbol{a}$ (respectivement $\boldsymbol{b}$) est un vecteur dont seules les k_X (respectivement k_Y) premières composantes sont non nulles.

5.5.2 Sécurité de HB$^{\#}$

La sécurité du protocole HB$^{\#}$ est malheureusement moins bien comprise que celle de RANDOM-HB$^{\#}$. Bien qu'il nous semble parfaitement raisonnable de conjecturer que HB$^{\#}$ est sûr dans le modèle actif, il ne nous a pas été possible de le démontrer. Un premier pas vers une preuve serait de démontrer que le puzzle « *Toeplitz-MHB* » est dur. Ce puzzle est défini exactement comme le puzzle MHB (cf. définition 5.5) à la différence près que la matrice secrète X est choisie aléatoirement parmi les matrices de Toeplitz de taille $k \times m$. Nous conjecturons que le puzzle Toeplitz-MHB est $\left(1 - \frac{1}{2^m}\right)$-dur. Si cette conjecture est vraie, alors on peut montrer par une transposition directe de la démonstration du théorème 5.5 que le protocole est sûr dans le modèle actif *lorsque la matrice Y est une matrice de Toeplitz et la matrice X une matrice uniformément aléatoire.* Pour montrer qu'il est également sûr de choisir la matrice X aléatoirement parmi les matrices de Toeplitz, il faudrait adapter la technique de simulation de la démonstration du théorème 5.5. En effet dans cette preuve on simule un prouveur $\mathcal{P}_{X,Y,\eta}$, où $X = X' \oplus X_{\boldsymbol{z},j}$, X' étant choisie aléatoirement et $X_{\boldsymbol{z},j}$ étant la matrice dont la j-ième ligne est égale au vecteur (inconnu)

$\boldsymbol{z}$ et toutes les autres lignes sont nulles. Or cette technique de simulation ne respecte pas la structure des matrices de Toeplitz. Il semble difficile de modifier la preuve de façon à incorporer le vecteur $\boldsymbol{z}$ dans une matrice de Toeplitz et non une matrice uniformément aléatoire. On peut cependant espérer contourner ce problème en utilisant les techniques de démonstration de Katz, Shin et Smith [KS06a, KS06b] qui simulent le prouveur de façon beaucoup plus simple.

Par contre, la sécurité de HB$^{\#}$ dans le modèle des attaques actives se transpose directement au modèle GRS, en vertu du théorème suivant, qui est le pendant du théorème 5.9 pour RANDOM-HB$^{\#}$.

Théorème 5.13. *Soient (k_X, k_Y, m, η, u) des paramètres du protocole HB$^{\#}$ vérifiant la condition (5.1). Supposons qu'il existe un adversaire $\mathcal{A}$ attaquant le protocole HB$^{\#}$ dans le modèle GRS, perturbant au plus q exécutions du protocole, dont le temps de calcul est inférieur à T, et possédant un avantage supérieur à δ. Alors il existe un adversaire $\mathcal{A}'$ attaquant le protocole HB$^{\#}$ dans le modèle actif, interagissant avec le prouveur au plus q fois, dont le temps de calcul est inférieur à $\mathcal{O}(T)$, et possédant un avantage supérieur à $(P_{FA} + \delta)(1 - q\epsilon)$, où ϵ est une fonction négligeable du paramètre de sécurité.* ♢

Corollaire 5.14. *Si le protocole HB$^{\#}$ avec des paramètres vérifiant la condition (5.1) est sûr dans le modèle actif, alors il est sûr dans le modèle GRS.* ▽

DÉMONSTRATION. La preuve est analogue à celle du théorème 5.9 et repose sur l'observation que le lemme 5.11 reste vrai dans le cas des matrices de Toeplitz, à savoir que pour tout entier $d \in [\![1, \lfloor \frac{m}{2} \rfloor]\!]$, on a

$$\Pr\left[\boldsymbol{x} \xleftarrow{\$} \{0,1\}^{k_X+m-1} \,:\, d_{\min}(\mathcal{T}_{\boldsymbol{x}}) \leq d\right] \leq 2^{-\left(1-\frac{k_X}{m}-H_2\left(\frac{d}{m}\right)\right)m} .$$

Ce résultat se démontre facilement à partir du lemme 5.12. ■

Remarque 5.3. Nous tenons à souligner que, même si nous pensons que le puzzle Toeplitz-MHB est dur, il est sans doute plus facile à résoudre que le puzzle MHB original. En effet, nous avons vu que l'on peut réécrire ce puzzle comme un problème LPN classique impliquant un unique vecteur $\boldsymbol{x}$ mais où les vecteurs $\boldsymbol{a}_i$ ont une forme spéciale donnée par la matrice (5.2). Or il n'est pas impossible que l'on puisse tirer parti de cette forme particulière pour accélérer les algorithmes de résolution du problème LPN tel que l'algorithme BKW, qui fonctionne en combinant les vecteurs $\boldsymbol{a}_i$ pour obtenir les vecteurs de base $\boldsymbol{e}_i$.∗

5.6 L'attaque man-in-the-middle de Ouafi *et al.*

Dans l'article original sur (RANDOM-)HB$^{\#}$, nous conjecturions que ces deux protocoles étaient résistants aux attaques man-in-the-middle générales (c'est-à-dire non-GRS et nécessitant la manipulation des messages du prouveur vers le vérifieur). Cependant, Ouafi, Overbeck et Vaudenay ont montré que pour certaines classes de paramètres, ce n'est pas le cas [OOV08]. Leur attaque tire parti du fait que lorsque l'on a obtenu passivement un échange entre un prouveur et un vérifieur légitimes constitué des vecteurs $\bar{\boldsymbol{b}}$, $\bar{\boldsymbol{a}}$ et $\bar{\boldsymbol{z}} = \bar{\boldsymbol{a}} \cdot X \oplus \bar{\boldsymbol{b}} \cdot Y \oplus \bar{\boldsymbol{\nu}}$, on peut « superposer » cet échange aux sessions suivantes du protocole et utiliser la décision du vérifieur pour « débruiter » $\bar{\boldsymbol{z}}$, c'est-à-dire retrouver la valeur exacte du vecteur $\bar{\boldsymbol{a}} \cdot X \oplus \bar{\boldsymbol{b}} \cdot Y$.

Plus précisément, lorsque l'on superpose le triplet $(\bar{\boldsymbol{b}}, \bar{\boldsymbol{a}}, \bar{\boldsymbol{z}})$ à une session ultérieure $(\boldsymbol{b}, \boldsymbol{a}, \boldsymbol{z})$, où $\boldsymbol{z} = \boldsymbol{a} \cdot X \oplus \boldsymbol{b} \cdot Y \oplus \boldsymbol{\nu}$, le vérifieur, lorsqu'il prend sa décision, indique si le poids de Hamming du vecteur d'erreur total $\boldsymbol{\nu} \oplus \bar{\boldsymbol{\nu}}$ est inférieur ou supérieur à t. En effectuant l'expérience un grand nombre de fois avec le même triplet $(\bar{\boldsymbol{b}}, \bar{\boldsymbol{a}}, \bar{\boldsymbol{z}})$, on peut ainsi estimer le poids de Hamming de $\bar{\boldsymbol{\nu}}$. L'attaque consiste alors à tester, pour chaque position de bit $i \in [\![1, m]\!]$, si, pour le vecteur $\bar{\boldsymbol{z}}'$ obtenu en inversant le bit en position i de $\bar{\boldsymbol{z}}$, la superposition du triplet $(\bar{\boldsymbol{b}}, \bar{\boldsymbol{a}}, \bar{\boldsymbol{z}}')$ fait augmenter ou diminuer la probabilité d'acceptation du vérifieur, ce qui indiquera respectivement que le bit d'erreur en position i du vecteur $\bar{\boldsymbol{\nu}}$ était 0 ou 1. On obtient ainsi des équations linéaires sur les matrices X et Y qui permettent de retrouver ces deux matrices lorsqu'on en a suffisamment. De nombreuses optimisations de cette attaque élémentaire sont décrites dans [OOV08].

La condition nécessaire et suffisante pour que la complexité de cette attaque soit polynomiale est que le test consistant à superposer le triplet $(\bar{\boldsymbol{b}}, \bar{\boldsymbol{a}}, \bar{\boldsymbol{z}})$ pour retrouver le poids de Hamming du vecteur d'erreur $\bar{\boldsymbol{\nu}}$ ait une sensibilité non négligeable. Pour cela, la superposition de deux vecteurs de bruit indépendants $\boldsymbol{\nu}$ et $\boldsymbol{\nu}'$ doit avoir un poids de Hamming inférieur au seuil t avec une probabilité suffisamment forte. En effet dans le cas contraire le vérifieur rejette avec une probabilité exponentiellement proche de un toutes les authentifications perturbées et la complexité du test devient exponentielle. Lorsque l'on superpose deux vecteurs de bruit indépendants, chaque bit du résultat vaut 1 avec une probabilité égale à $\eta' = 2\eta(1-\eta)$. Par conséquent, la condition pour que l'attaque soit polynomiale [4] est $t \geq 2\eta(1-\eta)m$.

Remarquons que cette attaque *n'est absolument pas liée au problème LPN* et que sa complexité est dominée par le nombre de sessions d'authentification perturbées (elle nécessite potentiellement de modifier un grand nombre de sessions entre le prouveur et le vérifieur authentique). Cette remarque sera importante lorsque nous discuterons du choix des paramètres pour HB$^{\#}$.

La question du statut de (RANDOM-)HB$^{\#}$ lorsque les paramètres η et u vérifient $u < 2\eta(1-\eta)$ reste ouverte : peut-on prouver la sécurité du protocole dans ce cas ? Malgré nos efforts, nous n'avons pu étendre la preuve de sécurité des théorèmes 5.9 et 5.13 au cas des attaques man-in-the-middle générales. Le principal obstacle consiste à savoir comment simuler le vérifieur lorsque l'attaquant modifie arbitrairement les communications.

5.7 Choix des paramètres

Lorsque l'on tente de choisir des paramètres concrets pour HB$^{\#}$, de nombreux compromis doivent être faits. En particulier :

- la probabilité de fausse acceptation P_{FA} dépend de m et du paramètre u déterminant le seuil $t = um$ (c'est une fonction croissante de u à m fixé et décroissante de m à u fixé) ;
- la probabilité de faux rejet P_{FR} dépend de m, u et η (c'est une fonction décroissante de u et m et croissante de η) ;
- la taille totale de la clé secrète ainsi que le coût de communication sont des fonctions croissantes de m, k_X et k_Y ;

4. Plus précisément, il est montré dans [OOV08] que l'attaque est polynomiale lorsque

$$t \geq 2\eta(1-\eta)m - \mathcal{O}\left(\sqrt{\ln |K|}\right)\sqrt{\eta(1-\eta)m},$$

où $|K|$ est la taille totale de la clé, et exponentielle dans le cas contraire.

HB$^{\#}$					P_{FR}	P_{FA}	Transmission	Taille de
k_X	k_Y	m	η	t			(bits)	clé (bits)
80	512	1 164	0,25	405	2^{-45}	2^{-83}	1 756	2 918
80	512	441	0,125	113	2^{-45}	2^{-83}	1 033	1 472
80	576	367	0,1	86	2^{-45}	2^{-83}	1 023	1 388

TABLE 5.1 – Paramètres pour le protocole HB$^{\#}$ offrant une sécurité de 2^{80} opérations contre les attaques GRS. Le troisième jeu de paramètres a été proposé par Levieil [Lev08].

– enfin, la sécurité globale du protocole doit prendre en compte les trois points suivants :

1. le problème LPN de paramètres (k_Y, η) (dont la difficulté augmente avec k_Y et η) doit être impossible à résoudre ;
2. il doit être impossible de retrouver k_X par recherche exhaustive ; en effet il n'est pas possible de retrouver le secret X en se ramenant à un problème LPN, si bien que la taille du paramètre k_X peut être sensiblement réduite par rapport à celle de k_Y ; ceci avait déjà été noté par Levieil et Fouque [LF06] ;
3. l'attaque de Ouafi *et al.* [OOV08], dont la complexité est principalement déterminée par les paramètres η, m et t, doit être impossible à mettre en œuvre.

Il existe de nombreuses méthodes pour fixer les paramètres de HB$^{\#}$. Si l'on ne tient pas compte de l'attaque de Ouafi *et al.* (par exemple si l'on ne cherche pas à se prémunir des attaques man-in-the-middle générales), une méthode possible est la suivante :

1. fixer les probabilités de faux rejet et de fausse acceptation maximales, typiquement $P_{\mathrm{FA}} < 2^{-80}$ et $P_{\mathrm{FR}} < 2^{-40}$;
2. fixer k_X selon le niveau de sécurité voulu (typiquement $k_X = 80$), et η et k_Y tels que le problème LPN de paramètres (k_Y, η) ne puisse être résolu par les meilleurs algorithmes avec une complexité inférieure au niveau de sécurité désiré (cf. section 3.5.7) ;
3. chercher le plus petit m pour lequel il existe un seuil t menant à des probabilités de fausse acceptation et de faux rejet inférieures aux valeurs maximales fixées.

Dans l'article original, nous proposions les deux premiers jeux de paramètres donnés dans la table 5.1. Ces paramètres garantissent la sécurité du protocole dans le modèle GRS contre tout attaquant effectuant moins de 2^{80} opérations, mais Ouafi *et al.* ont montré qu'ils étaient insuffisants pour offrir la même sécurité contre les attaques man-in-the-middle générales, rendant ces paramètres obsolètes. Un troisième jeu de paramètres, également vulnérable à l'attaque de Ouafi *et al.*, a été proposé par Levieil [Lev08].

Ouafi *et al.* ont montré que pour échapper à leur attaque, il n'y a guère d'autre possibilité que d'augmenter le paramètre m. Ainsi, ils estiment que pour obtenir une instance du protocole sûre contre un attaquant pouvant perturber jusqu'à 2^{80} sessions d'authentifications, il faut $m \geq 1\,700$ lorsque $\eta = 0,25$ et $m \geq 2\,900$ lorsque $\eta = 0,125$. Ces valeurs sont calculées pour une valeur du seuil égale à $t = \lceil \eta m \rceil$, ce qui implique une probabilité de faux rejet de 0,5 environ. Ces valeurs de m doivent donc être augmentées pour obtenir une probabilité de faux rejet acceptable. Afin de limiter cette nécessaire augmentation de m, on peut être tenté de modifier le protocole de façon à ce que le prouveur teste le vecteur de bruit avant utilisation et le génère à nouveau si son poids de Hamming est supérieur

au seuil de rejet t. Cependant Ouafi *et al.* ont montré [OOV08, Section 4] que cela mène à d'autres types d'attaques.

Nous tenons tout de même à souligner une différence importante entre une attaque fondée sur la résolution du problème LPN et l'attaque de Ouafi *et al.* Comme nous l'avons vu au chapitre 3, certains algorithmes de résolution du problème LPN, comme la variante de Lyubashevsky, sont capables d'utiliser un faible nombre de sessions d'authentifications. Il est donc naturel d'imposer une sécurité de 2^{80} *opérations* à des attaques fondées sur de tels algorithmes. En revanche, dans le cas de l'attaque de Ouafi *et al.*, il semble impossible de réduire le nombre de sessions requises. Par conséquent, il n'est pas déraisonnable de choisir des paramètres assurant une sécurité contre des attaquants capables de perturber seulement 2^{50} sessions environ, ce qui semble déjà une borne supérieure très suffisante. Dans ce cas la figure 2 de [OOV08] indique que $m = 900$ est suffisant pour $\eta = 0,25$.

5.8 Implémentation, optimisations et variantes

5.8.1 Implémentation

Le protocole HB$^{\#}$, tout comme HB^{+}, est très économique à implanter en hardware. Au niveau du prouveur (donc de l'étiquette dans un système RFID), il ne requiert que des portes AND et XOR et une source d'aléa. De plus, les bits du vecteur de masquage $\boldsymbol{b}$ peuvent être générés à la volée puis envoyés au vérifieur tandis que la valeur du produit $\boldsymbol{b} \cdot Y$ est mise à jour. De même la valeur du produit $\boldsymbol{a} \cdot X$ peut être calculée au fur et à mesure de la réception des bits du vecteur challenge $\boldsymbol{a}$ sans avoir à mémoriser la totalité du vecteur. L'aléa nécessaire à la génération des bits de bruit et du vecteur $\boldsymbol{b}$ peut être produit dans une puce à partir de phénomènes physiques tels que le bruit thermique, le bruit de grenaille (« *shot noise* »), le bruit d'avalanche dans une diode Zener, etc.

D'autre part la vérification, vraisemblablement implémentée en software dans un système RFID, peut être accélérée en tirant parti du fait que le produit d'une matrice de Toeplitz de taille $n \times n$ et d'un vecteur de taille n peut être implémentée en temps $\mathcal{O}(n \log n)$ (plutôt que $\mathcal{O}(n^2)$ pour une matrice générique) en utilisant la Transformée de Fourier Rapide [GL96].

5.8.2 Variantes et problèmes ouverts

Variante à deux passes.

Dans l'article original proposant HB^{+}, Juels et Weis suggéraient d'étudier la variante à deux passes dans laquelle le vérifieur commence par envoyer le vecteur challenge $\boldsymbol{a}$, puis le prouveur répond en envoyant simultanément le vecteur $\boldsymbol{b}$ et le bit de réponse z. Cette variante est également envisageable pour (RANDOM-)HB$^{\#}$. Nous ne savons pas si cela est sûr : nous ne connaissons pas d'attaque mais nous ne savons pas non plus s'il est possible d'étendre la preuve du théorème 5.5 à cette variante. En effet, la preuve de sécurité repose de façon essentielle sur le fait que le prouveur envoie le vecteur $\boldsymbol{b}$ *avant* de connaître $\boldsymbol{a}$ (c'est ce qui permet de rembobiner l'adversaire pour le faire répondre à deux challenges $\hat{\boldsymbol{a}}_1$ et $\hat{\boldsymbol{a}}_2$ différents pour le même vecteur de masquage $\hat{\boldsymbol{b}}$).

Matrices de Toeplitz fondées sur un LFSR.

Dans son étude des codes d'authentification de messages fondés sur les matrices de Toeplitz [Kra94], Krawczyk notait que l'on peut restreindre un peu plus l'espace des clés

secrètes en ne considérant que les matrices de Toeplitz $\mathcal{T}$ de taille $k \times m$ dont les lignes sont générées par les états successifs d'un registre à décalage à rétroaction linéaire (LFSR) de taille m. Krawczyk a montré que lorsque le polynôme de rétroaction est irréductible, la famille de fonctions $(\boldsymbol{a} \mapsto \boldsymbol{a} \cdot \mathcal{T})$, paramétrée par le polynôme de rétroaction et l'état initial du LFSR, conserve de bonnes propriétés de hachage universel. Plus précisément, en notant $\mathcal{F}$ l'ensemble des matrices ainsi définies, on a pour tous vecteurs $\boldsymbol{a} \neq \mathbf{0}$ et $\boldsymbol{z}$ de longueurs respectives k et m :

$$\Pr\left[\mathcal{T} \xleftarrow{\$} \mathcal{F} \,:\, \boldsymbol{a} \cdot \mathcal{T} = \boldsymbol{z}\right] \leq \frac{k}{2^{m-1}} \ .$$

Une telle variante optimise à la fois la mémoire requise pour stocker la clé secrète (définie par un polynôme de rétroaction, qui peut être vu comme un circuit plutôt que comme de véritables bits de mémoire, et un état initial du LFSR pour chaque matrice, soit $2m$ bits en tout) ainsi que le temps de calcul. Cependant les effets sur la sécurité restent à étudier.

Préservation de la vie privée.

Comme nous l'avons vu avec la proposition Trusted-HB [BC08] à la section 4.9, le protocole HB^{+} et ses variantes peuvent être utilisés en tant que protocoles d'identification respectueux de la vie privée et de l'anonymat. Pour cela le vérifieur entame une session sans connaitre a priori l'identité du prouveur avec lequel il communique et recherche parmi sa base de clés secrètes celle menant à un taux d'erreur proche de η sur les équations (les autres auront un taux d'erreur de $\frac{1}{2}$ environ). Un adversaire interceptant les communications ne peut pas retrouver la clé secrète du prouveur sans connaitre la base de secrets et ne peut donc pas « tracer » le prouveur. Il serait intéressant d'étudier plus rigoureusement cet aspect de HB^{+} et de ses variantes dans les divers modèles de *privacy* proposés à ce jour [ADO05, JW07, Vau07].

5.9 Conclusion et tableau récapitulatif

Même si l'attaque de Ouafi *et al.* est venue remettre en cause les espoirs que l'on pouvait fonder sur RANDOM-HB$^{\#}$ et HB$^{\#}$, ces deux protocoles nous semblent dignes de continuer à être étudiés à plus d'un titre.

Les deux problèmes ouverts les plus importants sont la compréhension de l'effet de l'utilisation de matrices de Toeplitz sur la sécurité des deux protocoles (plus fondamentalement la compréhension de la difficulté du puzzle Toeplitz-MHB, cf. section 5.5.2), et l'étude de la sécurité de RANDOM-HB$^{\#}$ et HB$^{\#}$ dans le modèle des attaques man-in-the-middle générales lorsque $u < 2\eta(1-\eta)$. Si de tels paramètres se révélaient sûrs, on pourrait alors espérer obtenir, par diverses optimisations, un protocole d'authentification prouvé sûr contre les attaques man-in-the-middle générales avec une taille de clé raisonnable (même si descendre sous le millier de bits de clé semble difficile).

Le problème LPN, par la simplicité des opérations qu'il met en jeu, nous semble un problème conjecturé difficile de choix pour concevoir des protocoles cryptographiques prouvés sûrs, et on peut se demander quelles autres fonctionnalités cryptographiques sont réalisables à l'aide de ce problème (authentification de message, authentification « zero-knowledge », etc.). Nous allons voir dans le prochain chapitre que l'on peut obtenir un schéma de chiffrement symétrique.

Protocole	Paramètres	Taille de clé (bits)	Communication (bits)	passif	actif	GRS	MITM
HB	(k, r, η, u)	k	$r(k+1)$	$\surd$	†	†	†
HB$^+$	(k_x, k_y, r, η, u)	$k_x + k_y$	$r(k_x + k_y + 1)$	$\surd$	$\surd$	†	†
RANDOM-HB$^\#$	(k_X, k_Y, m, η, u) $u \geq 2\eta(1-\eta)$	$m(k_X + k_Y)$	$k_X + k_Y + m$	$\surd$	$\surd$	$\surd$	†
RANDOM-HB$^\#$	(k_X, k_Y, m, η, u) $u < 2\eta(1-\eta)$	$m(k_X + k_Y)$	$k_X + k_Y + m$	$\surd$	$\surd$	$\surd$	?
HB$^\#$	(k_X, k_Y, m, η, u) $u \geq 2\eta(1-\eta)$	$k_X + k_Y + 2m - 2$	$k_X + k_Y + m$	?	?	?	†
HB$^\#$	(k_X, k_Y, m, η, u) $u < 2\eta(1-\eta)$	$k_X + k_Y + 2m - 2$	$k_X + k_Y + m$	?	?	?	?

TABLE 5.2 – Tableau récapitulatif des caractéristiques des principaux protocoles d'authentification étudiés dans cette partie. La signification des symboles pour les modèles d'attaque est la suivante : $\surd$ signifie que le protocole est prouvé sûr, † qu'il existe une attaque polynomiale, et ? que la sécurité du protocole est un problème ouvert.

Chapitre 6

LPN-C, un schéma de chiffrement à clé secrète à sécurité prouvée

Dans les précédents chapitres de cette partie, nous nous sommes concentrés sur les schémas d'authentification d'entité fondés sur le problème LPN. Il est cependant naturel de se demander s'il est possible de réaliser d'autres fonctionnalités courantes de la cryptographie à l'aide de ce même problème. Nous allons voir dans le présent chapitre qu'il est possible de construire un schéma de chiffrement à clé secrète (probabiliste) dont la sécurité peut être prouvée sous l'hypothèse de la difficulté du problème LPN. Le cryptosystème que nous proposons est dénommé LPN-C, pour *LPN-Cipher*. Après avoir rappelé les principales définitions liées aux schémas de chiffrement symétriques, nous décrivons LPN-C, puis étudions sa sécurité dans différents modèles. Nous suggérons au passage une technique permettant de construire un code d'authentification de message (MAC) également fondé sur le problème LPN.

6.1 Schémas de chiffrement à clé secrète

6.1.1 Définitions

Nous commençons par rappeler la définition d'un schéma de chiffrement symétrique, que nous adaptons (en la simplifiant légèrement) de [KY06].

Définition 6.1 (Schéma de chiffrement symétrique)
Un schéma de chiffrement probabiliste à clé secrète (ou symétrique) est un triplet d'algorithmes $\Gamma = (\mathcal{G}, \mathcal{E}, \mathcal{D})$ tel que :

- l'algorithme de génération de clé $\mathcal{G}$, prenant en entrée le paramètre de sécurité k, retourne une clé secrète aléatoire $K \in \mathcal{K}(k) : K \leftarrow \mathcal{G}(1^k)$;
- l'algorithme de chiffrement $\mathcal{E}$ est un algorithme probabiliste polynomial qui prend en entrée une clé secrète K et un texte en clair $\boldsymbol{x} \in \{0,1\}^*$ et retourne un chiffré $\boldsymbol{y}$: $\boldsymbol{y} \leftarrow \mathcal{E}_K(\boldsymbol{x})$;
- l'algorithme de déchiffrement $\mathcal{D}$ est un algorithme déterministe polynomial qui prend en entrée une clé secrète K et une chaîne de bits $\boldsymbol{y}$ et retourne soit le texte clair correspondant $\boldsymbol{x}$ ou un symbole spécial $\perp$: $\boldsymbol{x}' \leftarrow \mathcal{D}_K(\boldsymbol{y})$, $\boldsymbol{x}' \in \{0,1\}^* \cup \{\perp\}$. ♦

On requiert habituellement que $\mathcal{D}_K(\mathcal{E}_K(\boldsymbol{X})) = \boldsymbol{X}$ pour tout $\boldsymbol{X} \in \{0,1\}^*$. On peut cependant assouplir cette condition et tolérer des erreurs de déchiffrement, c'est-à-dire requérir seulement que $\mathcal{D}_K(\mathcal{E}_K(\boldsymbol{X})) = \boldsymbol{X}$, excepté avec une probabilité fonction négligeable du paramètre de sécurité k et de $|\boldsymbol{X}|$.

6.1.2 Modèles de sécurité pour le chiffrement symétrique

L'étude de la définition des notions de sécurité pour le chiffrement a principalement été faite dans le paradigme asymétrique, le travail fondateur étant la définition de la *sécurité sémantique* et de l'*indistinguabilité* par Goldwasser et Micali [GM84], puis plus tard celle de la *non-malléabilité* par Dolev, Dwork et Naor [DDN00]. Les relations entre ces différentes notions ont ensuite été étudiées de manière extensive [BDPR98, BS99, WSI03]. Informellement, l'indistinguabilité caractérise le secret offert par le schéma : un attaquant doit être incapable de distinguer les chiffrés de deux clairs choisis par l'attaquant lui-même. La non-malléabilité caractérise l'impossibilité pour un attaquant de « manipuler » les chiffrés : étant donné un chiffré $\boldsymbol{y}$, l'attaquant doit être incapable de générer un nouveau chiffré $\boldsymbol{y}'$ tel que les clairs correspondant à $\boldsymbol{y}$ et $\boldsymbol{y}'$ soit reliés d'une façon contrôlée par l'attaquant.

Dans le cadre symétrique, même si la définition du *secret parfait* par Shannon remonte à 1949 [Sha49], l'étude systématique des notions de sécurité pour le chiffrement est liée à celle des modes opératoires d'un chiffrement par bloc [BDJR97] [1]. Katz et Yung ont ensuite caractérisé et hiérarchisé de façon complète les différentes notions de sécurité et les différents modèles d'attaque [KY06]. Les définitions que nous présentons maintenant sont tirées de leur article.

Un adversaire (contre la propriété d'indistinguabilité IND ou de non-malléabilité NM) procède en deux phases et sera noté comme une paire d'algorithmes $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$. On distinguera différentes classes d'attaquants selon les oracles (chiffrement et/ou déchiffrement) auxquels ils ont accès durant chaque phase, ce que l'on notera PX-CY, où P désigne l'oracle de chiffrement et C l'oracle de déchiffrement, et X, $Y \in \{0, 1, 2\}$ indiquent durant quelle phase d'attaque $\mathcal{A}$ peut accéder à l'oracle correspondant :

- $X, Y = 0$: $\mathcal{A}$ n'a jamais accès à l'oracle correspondant ;
- $X, Y = 1$: $\mathcal{A}$ n'a accès à l'oracle correspondant que durant la première phase d'attaque (on parle aussi d'attaque à clairs ou chiffrés choisis *non-adaptative*) ;
- $X, Y = 2$: $\mathcal{A}$ a accès à l'oracle correspondant durant les deux phases d'attaque (on parle aussi d'attaque à clairs ou chiffrés choisis *adaptative*).

Ainsi, une attaque IND-P1-C0 désigne une attaque à clairs choisis non-adaptative contre l'indistinguabilité d'un schéma, et une attaque NM-P2-C2 désigne une attaque à clairs et chiffrés choisis adaptative contre la non-malléabilité d'un schéma.

Lors de la première phase d'attaque, $\mathcal{A}_1$ peut interagir librement avec les oracles auxquels il a accès. Puis il retourne une distribution sur l'espace des clairs (une paire $(\boldsymbol{x}_1, \boldsymbol{x}_2)$ dans le cas d'une attaque contre l'indistinguabilité, ou une distribution plus complexe dans le cas d'une attaque contre la non-malléabilité). Un chiffré $\boldsymbol{y}$ est alors tiré aléatoirement selon la distribution retournée par $\mathcal{A}_1$ et transmis pour la deuxième phase d'attaque à $\mathcal{A}_2$, et le succès de $\mathcal{A}$ est déterminé suivant le but de l'attaque (dans le cas d'une attaque sur l'indistinguabilité, il doit distinguer si le chiffré correspond à $\boldsymbol{x}_1$ ou $\boldsymbol{x}_2$).

Nous ne donnons la définition formelle que pour les attaques IND-PX-CY car ce sont les attaques qui vont principalement nous intéresser par la suite. Pour la définition rigoureuse de la non-malléabilité, nous renvoyons à [KY06].

1. En effet, la quasi-totalité des schémas de chiffrement symétriques proposés à ce jour sont soit des algorithmes de chiffrement à flot, soit le résultat de la combinaison d'un algorithme de chiffrement par bloc et d'un mode opératoire tel que CBC, CTR, etc.

Définition 6.2 (Attaque IND-PX-CY)
Soit $\Gamma = (\mathcal{G}, \mathcal{E}, \mathcal{D})$ un schéma de chiffrement indexé par un paramètre de sécurité k et $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ un adversaire. Pour $X, Y \in \{0, 1, 2\}$, l'avantage de $\mathcal{A}$ contre l'indistinguabilité de Γ est définie par :

$$\mathrm{Adv}^{\text{IND-PX-CY}}_{\mathcal{A},\Gamma}(k) \stackrel{\text{def}}{=} \left| \Pr\left[K \leftarrow \mathcal{G}(1^k); (\boldsymbol{x_0}, \boldsymbol{x_1}, s) \leftarrow \mathcal{A}_1^{\mathcal{O}_1,\mathcal{O}'_1}(1^k); \right.\right.$$
$$\left.\left. b \xleftarrow{\$} \{0,1\}; \boldsymbol{y} \leftarrow \mathcal{E}_K(\boldsymbol{x_b}) \,:\, \mathcal{A}_2^{\mathcal{O}_2,\mathcal{O}'_2}(1^k, s, \boldsymbol{y}) = b\right] - \frac{1}{2}\right| \ ,$$

où $(\mathcal{O}_1, \mathcal{O}_2)$ est égal respectivement à $(\emptyset, \emptyset)$, $(\mathcal{E}_K(\cdot), \emptyset)$, $(\mathcal{E}_K(\cdot), \mathcal{E}_K(\cdot))$ lorsque X est égal respectivement à 0, 1, 2 et $(\mathcal{O}'_1, \mathcal{O}'_2)$ est égal respectivement à $(\emptyset, \emptyset)$, $(\mathcal{D}_K(\cdot), \emptyset)$, $(\mathcal{D}_K(\cdot), \mathcal{D}_K(\cdot))$ lorsque Y est égal respectivement à 0, 1, 2, et s est une information transmise de $\mathcal{A}_1$ à $\mathcal{A}_2$. Les messages en clair retournés par $\mathcal{A}_1$ doivent être tels que $|\boldsymbol{x_0}| = |\boldsymbol{x_1}|$. De plus, lorsque $Y = 2$, $\mathcal{A}_2$ n'est pas autorisé à faire la requête $\mathcal{D}_K(\boldsymbol{y})$.
Nous dirons que Γ est sûr au sens IND-PX-CY si $\mathrm{Adv}^{\text{IND-PX-CY}}_{\mathcal{A},\Gamma}(k)$ est négligeable pour tout adversaire $\mathcal{A}$ polynomial. ♦

La comparaison des propriétés de sécurité IND-PX-CY et NM-PX-CY (implication équivalence, ou incomparabilité) a été exhaustivement effectuée par Katz et Yung [KY06]. Les deux relations qui vont nous être utiles sont les suivantes :

1. la sécurité contre les attaques à clairs choisis non-adaptatives implique la sécurité contre les attaques à clairs choisis adaptatives :

   $$\Gamma \text{ est sûr au sens IND-P1-C}Y \Rightarrow \Gamma \text{ est sûr au sens IND-P2-C}Y$$
   $$\Gamma \text{ est sûr au sens NM-P1-C}Y \Rightarrow \Gamma \text{ est sûr au sens NM-P2-C}Y$$

2. pour les attaques à clairs et chiffrés choisis adaptatives, indistinguabilité et non-malléabilité sont des notions équivalentes :

   $$\Gamma \text{ est sûr au sens IND-P2-C2} \Leftrightarrow \Gamma \text{ est sûr au sens NM-P2-C2}$$

6.2 Description de LPN-C

Nous en venons maintenant à la description du schéma de chiffrement LPN-C, qui est résumé sur la figure 6.1.

Soit $C : \{0,1\}^r \to \{0,1\}^m$ un code correcteur d'erreurs de paramètres $[m, r, d]$, c'est-à-dire de longueur m, de dimension r et de distance minimale d, de capacité de correction $t = \left\lfloor \frac{d-1}{2} \right\rfloor$. Ce code correcteur est supposé publiquement connu, et nous noterons C^{-1} la procédure de décodage. Soit M une matrice binaire secrète de taille $k \times m$, constituant la clé secrète du cryptosystème. Pour chiffrer un vecteur $\boldsymbol{x}$ de longueur r, l'émetteur tire un vecteur aléatoire $\boldsymbol{a}$ de longueur k et calcule

$$\boldsymbol{y} = C(\boldsymbol{x}) \oplus \boldsymbol{a} \cdot M \oplus \boldsymbol{\nu} \ ,$$

où $\boldsymbol{\nu} \leftarrow \mathtt{Ber}_{m,\eta}$ est un vecteur de bruit de longueur m tel que chacun de ses bits vaut indépendamment 1 avec probabilité η et 0 avec probabilité $1 - \eta$. Le texte chiffré est constitué de la paire $(\boldsymbol{a}, \boldsymbol{y})$.

À la réception de ce chiffré, le destinataire déchiffre en calculant :

$$\boldsymbol{y} \oplus \boldsymbol{a} \cdot M = C(\boldsymbol{x}) \oplus \boldsymbol{\nu} \ ,$$

Paramètres	Paramètre de sécurité k Polynômes (en k) m, r, d avec $m > r$ Paramètre de bruit $\eta \in]0, \frac{1}{2}[$
Composants publics	Un code correcteur $C : \{0,1\}^r \to \{0,1\}^m$ de paramètres $[m, r, d]$ et l'algorithme de décodage correspondant C^{-1}
Génération de la clé secrète	Entrée : 1^k Sortie : une matrice aléatoire binaire M de taille $k \times m$
Algorithme de chiffrement	Entrée : un vecteur $\boldsymbol{x}$ de taille r Tirer un vecteur aléatoire $\boldsymbol{a} \xleftarrow{\$} \{0,1\}^k$ Tirer un vecteur de bruit $\boldsymbol{\nu} \leftarrow \mathtt{Ber}_{m,\eta}$ Calculer $\boldsymbol{y} = C(\boldsymbol{x}) \oplus \boldsymbol{a} \cdot M \oplus \boldsymbol{\nu}$ Sortie : $(\boldsymbol{a}, \boldsymbol{y})$
Algorithme de déchiffrement	Entrée : $(\boldsymbol{a}, \boldsymbol{y})$ Calculer $\boldsymbol{y} \oplus \boldsymbol{a} \cdot M$ Décoder le résultat à l'aide de C^{-1} Sortie : la sortie de C^{-1} ou $\perp$ s'il est impossible de décoder

FIGURE 6.1 – Description du schéma de chiffrement LPN-C.

puis en décodant le résultat à l'aide de C^{-1}. Si le décodage n'est pas possible (ce qui peut arriver lorsque le code n'est pas parfait), l'algorithme de déchiffrement retourne $\perp$.

Si le message n'est pas de taille r, il est complété (par exemple avec un 1 suivi de 0) jusqu'à ce que sa taille soit égale au plus petit multiple de r strictement supérieur à $|\boldsymbol{x}|$ et chiffré bloc par bloc.

6.2.1 Erreurs de déchiffrement

Des erreurs de déchiffrement surviennent lorsque le poids de Hamming du vecteur de bruit $\boldsymbol{\nu}$ est supérieur à la capacité de correction t du code correcteur d'erreurs. Dans ce cas l'algorithme de déchiffrement retourne $\perp$, voire un clair différent du message réellement chiffré par l'émetteur. Lorsque le vecteur de bruit est tiré aléatoirement selon la distribution $\mathtt{Ber}_{m,\eta}$, la probabilité d'erreur de déchiffrement est donnée par :

$$P_{\text{ED}} = \sum_{i=t+1}^{m} \binom{m}{i} \eta^i (1-\eta)^{m-i} \ .$$

Afin d'éliminer ces erreurs de déchiffrement, le poids de Hamming du vecteur de bruit peut être testé avant que ce vecteur ne soit utilisé. Si $\mathtt{Hwt}(\boldsymbol{\nu}) > t$, l'émetteur tire un nouveau vecteur de bruit selon la distribution $\mathtt{Ber}_{m,\eta}$. Lorsque les paramètres sont choisis tels que $\eta m < (1-\epsilon)t$ pour un $\epsilon > 0$, cela n'arrive qu'avec une probabilité négligeable si bien que l'algorithme de chiffrement reste polynomial. Cependant cela pourrait avoir des conséquences sur la sécurité du schéma, car l'attaquant possède dans ce cas l'information supplémentaire que $\mathtt{Hwt}(\boldsymbol{\nu}) \leq t$.

6.2.2 Propriété de pseudo-homomorphisme

LPN-C possède une propriété inhabituelle pour un chiffrement symétrique et potentiellement intéressante : il est « presque » homomorphique. La propriété d'homomorphisme caractérise les schémas de chiffrement pour lesquels il est possible d'effectuer une opération algébrique sur les textes clairs en effectuant une opération algébrique sur les chiffrés correspondants. Cette notion s'avère extrêmement intéressante pour les schémas de chiffrement à clé publique et sert par exemple pour les protocoles de vote électronique [CRS05].

Nous supposerons ici que le code correcteur est linéaire. Considérons deux clairs $\boldsymbol{x}_1$ et $\boldsymbol{x}_2$ et deux chiffrés correspondants $(\boldsymbol{a}_1, \boldsymbol{y}_1)$ et $(\boldsymbol{a}_2, \boldsymbol{y}_2)$ avec :

$$\boldsymbol{y}_1 = C(\boldsymbol{x}_1) \oplus \boldsymbol{a}_1 \cdot M \oplus \boldsymbol{\nu}_1 \quad \text{et} \quad \boldsymbol{y}_2 = C(\boldsymbol{x}_2) \oplus \boldsymbol{a}_2 \cdot M \oplus \boldsymbol{\nu}_2 \ .$$

On a alors :

$$\boldsymbol{y}_1 \oplus \boldsymbol{y}_2 = C(\boldsymbol{x}_1 \oplus \boldsymbol{x}_2) \oplus (\boldsymbol{a}_1 \oplus \boldsymbol{a}_2) \cdot M \oplus (\boldsymbol{\nu}_1 \oplus \boldsymbol{\nu}_2) \ .$$

Par conséquent, si $\mathtt{Hwt}(\boldsymbol{\nu}_1 \oplus \boldsymbol{\nu}_2) \leq t$, $(\boldsymbol{a}_1 \oplus \boldsymbol{a}_2, \boldsymbol{y}_1 \oplus \boldsymbol{y}_2)$ est un chiffré valide pour le clair $\boldsymbol{x}_1 \oplus \boldsymbol{x}_2$. Il est aisé de voir que $\boldsymbol{\nu}' = \boldsymbol{\nu}_1 \oplus \boldsymbol{\nu}_2$ est un vecteur de bruit de paramètre de bruit $\eta' = 2\eta(1-\eta)$. Par conséquent si η et t sont tels que $\eta' m < (1-\epsilon)t$ pour un $\epsilon > 0$, $\mathtt{Hwt}(\boldsymbol{\nu}') \leq t$ avec forte probabilité et le schéma est donc homomorphique. Par contre on ne pourra pas additionner les chiffrés sans limite car le bruit augmente avec le nombre de chiffrés additionnés.

6.3 Sécurité de LPN-C

6.3.1 Preuve de sécurité dans le modèle IND-P2-C0

Nous commençons par prouver que LPN-C est sûr contre les attaques à clairs choisis adaptatives moyennant l'hypothèse de la difficulté du problème LPN (conjecture 3.1). Pour cela, nous utiliserons le lemme 3.3 rappelé au chapitre 3 et qui exprime que si le problème LPN est dur, aucun algorithme efficace ne peut distinguer avec une probabilité non négligeable les réponses d'un oracle $\Pi_{\boldsymbol{x},\eta}$ de vecteurs tirés selon la distribution uniforme U_{k+1} sur $\{0,1\}^{k+1}$.

Théorème 6.1. *Supposons qu'il existe un adversaire $\mathcal{A}$ attaquant le schéma LPN-C de paramètres (k, m, r, d, η) dans le modèle IND-P2-C0 avec un avantage δ, faisant au plus q requêtes à l'oracle de chiffrement et effectuant au plus T opérations. Alors il existe un algorithme $\mathcal{D}$ faisant $\mathcal{O}(q)$ requêtes d'oracle, effectuant $\mathcal{O}(T)$ opérations, et tel que :*

$$\left| \Pr\left[\boldsymbol{s} \xleftarrow{\$} \{0,1\}^k \ : \ \mathcal{D}^{\Pi_{\boldsymbol{s},\eta}}(1^k) = 1 \right] - \Pr\left[\mathcal{D}^{U_{k+1}}(1^k) = 1 \right] \right| \geq \frac{\delta}{m} \ . \qquad \diamondsuit$$

Corollaire 6.2. *Supposons le problème LPN dur. Alors LPN-C est sûr dans le modèle IND-P2-C0.* ▽

Démonstration. Comme nous l'avons déjà fait remarquer, la sécurité contre les attaques à clairs choisis non-adaptatives P1 implique la sécurité contre les attaques à clairs choisis adaptatives P2. On pourra donc se restreindre aux attaquants ne faisant de requêtes à l'oracle de chiffrement que pendant la première partie de l'attaque (avant de voir le chiffré constituant le challenge).

La preuve procède par un argument hybride. Nous allons tout d'abord définir les distributions suivantes sur $\{0,1\}^{k+m}$. Pour $j \in [\![0, m]\!]$, soit M' une matrice binaire de taille $k \times (m-j)$. La distribution de probabilité $\mathcal{P}_{j,M',\eta}$ est définie par :

$$\{\boldsymbol{a} \xleftarrow{\$} \{0,1\}^k; \boldsymbol{r} \xleftarrow{\$} \{0,1\}^j; \boldsymbol{\nu} \leftarrow \mathrm{Ber}_{(m-j),\eta} \,:\, \boldsymbol{a}\|\boldsymbol{r}\|(\boldsymbol{a}\cdot M' \oplus \boldsymbol{\nu})\} \ .$$

Un vecteur $\boldsymbol{a}\|\boldsymbol{b}$ distribué selon cette loi est tel que les j premiers bits de $\boldsymbol{b}$ sont uniformément aléatoires, tandis que les $(m-j)$ derniers bits sont distribués selon $(m-j)$ distributions LPN indépendantes reliées à la colonne respective de M'. Remarquons que $\mathcal{P}_{m,M',\eta} = U_{k+m}$.

Nous allons également définir l'oracle de chiffrement hybride $\mathcal{E}'_{j,M',\eta}$ associé à la matrice secrète M' et de paramètre de bruit η : lorsqu'il reçoit en entrée un vecteur $\boldsymbol{x}$ de r bits constituant le clair, l'oracle de chiffrement l'encode en $C(\boldsymbol{x})$, tire un vecteur $\boldsymbol{a}\|\boldsymbol{b}$ de longueur $(k+m)$ distribué selon $\mathcal{P}_{j,M',\eta}$, et retourne $(\boldsymbol{a}, C(\boldsymbol{x}) \oplus \boldsymbol{b})$.

Décrivons maintenant comment le distingueur $\mathcal{D}$ procède. Rappelons que $\mathcal{D}$ a accès à un oracle $\mathcal{O}$ et veut distinguer s'il s'agit de U_{k+1} ou $\Pi_{\boldsymbol{s},\eta}$. Lorsqu'il reçoit en entrée le paramètre de sécurité 1^k, $\mathcal{D}$ tire un entier $j \in [\![1, m]\!]$ uniformément distribué. Si $j < m$, il tire également une matrice binaire M'de taille $k \times (m-j)$. Puis il exécute la première phase de l'attaquant $\mathcal{A}_1$. Lorsque $\mathcal{A}_1$ demande le chiffrement d'un vecteur $\boldsymbol{x}$, $\mathcal{D}$ obtient une paire $(\boldsymbol{a}, z)$ par une requête à son oracle $\mathcal{O}$, tire un vecteur aléatoire $\boldsymbol{r} \xleftarrow{\$} \{0,1\}^{j-1}$ de longueur $(j-1)$, et tire un vecteur de bruit $\boldsymbol{\nu}$ de longueur $(m-j)$ distribué selon $\mathrm{Ber}_{(m-j),\eta}$. Puis il forme le vecteur de masquage $\boldsymbol{b} = \boldsymbol{r}\|z\|(\boldsymbol{a}\cdot M' \oplus \boldsymbol{\nu})$ et retourne $(\boldsymbol{a}, C(\boldsymbol{x}) \oplus \boldsymbol{b})$. L'algorithme $\mathcal{A}_1$ retourne ensuite deux clairs $\boldsymbol{x}_1$ et $\boldsymbol{x}_2$. Le distingueur $\mathcal{D}$ tire $\alpha \in \{1,2\}$ au hasard et retourne à l'algorithme $\mathcal{A}_2$ le chiffré correspondant à $\boldsymbol{x}_\alpha$, chiffré de la même manière que précédemment. Si la réponse de $\mathcal{A}_2$ est correcte (ce que nous noterons $\mathcal{A} = \mathrm{OK}$), $\mathcal{D}$ retourne 1, dans le cas contraire il retourne 0.

On peut facilement vérifier que lorsque l'oracle auquel accède $\mathcal{D}$ est U_{k+1}, $\mathcal{D}$ simule un oracle de chiffrement $\mathcal{E}'_{j,M',\eta}$, alors que lorsque l'oracle auquel accède $\mathcal{D}$ est $\Pi_{\boldsymbol{s},\eta}$, $\mathcal{D}$ simule un oracle de chiffrement $\mathcal{E}'_{j-1,M'',\eta}$ où $M'' = \boldsymbol{s}\|M'$ est la matrice obtenue par concaténation de $\boldsymbol{s}$ et M'. Ainsi, l'avantage du distingueur peut être exprimé comme :

$$\begin{aligned}
\mathrm{Adv} &= \left|\Pr\left[\boldsymbol{s} \xleftarrow{\$} \{0,1\}^k \,:\, \mathcal{D}^{\Pi_{\boldsymbol{s},\eta}}(1^k) = 1\right] - \Pr\left[\mathcal{D}^{U_{k+1}}(1^k) = 1\right]\right| \\
&= \frac{1}{m}\left|\sum_{j=0}^{m-1} \Pr\left[\mathcal{A}^{\mathcal{E}'_{j,M',\eta}} = \mathrm{OK}\right] - \sum_{j=1}^{m} \Pr\left[\mathcal{A}^{\mathcal{E}'_{j,M',\eta}} = \mathrm{OK}\right]\right| \\
&= \frac{1}{m}\left|\Pr\left[\mathcal{A}^{\mathcal{E}'_{0,M',\eta}} = \mathrm{OK}\right] - \Pr\left[\mathcal{A}^{\mathcal{E}'_{m,M',\eta}} = \mathrm{OK}\right]\right| \ .
\end{aligned}$$

Remarquons maintenant que l'oracle de chiffrement $\mathcal{E}'_{0,M',\eta}$ est exactement un oracle de chiffrement LPN-C réel. D'autre part l'oracle de chiffrement $\mathcal{E}'_{m,M',\eta}$ chiffre tous les clairs en les masquant avec des vecteurs $\boldsymbol{b}$ uniformément aléatoires, si bien que dans ce cas l'adversaire $\mathcal{A}$ ne peut pas faire mieux (ou pire) que deviner α au hasard, auquel cas sa probabilité de succès est de $\frac{1}{2}$. Ainsi, la quantité

$$\left|\Pr\left[\mathcal{A}^{\mathcal{E}'_{0,M',\eta}} = \mathrm{OK}\right] - \Pr\left[\mathcal{A}^{\mathcal{E}'_{m,M',\eta}} = \mathrm{OK}\right]\right|$$

est exactement l'avantage de l'adversaire $\mathcal{A}$, qui est supérieur à δ par hypothèse. Le théorème s'ensuit.

La preuve du corollaire découle directement du lemme 3.3. ∎

Ainsi, nous venons de montrer que LPN-C est sûr contre les attaques à clairs choisis adaptatives. Qu'en est-il des attaques plus puissantes à chiffrés choisis ? Remarquons que lorsque le code est linéaire, LPN-C est clairement malléable, même lorsque l'adversaire n'a accès ni à l'oracle de chiffrement ni à l'oracle de déchiffrement : LPN-C n'est pas NM-P0-C0. En effet, étant donné un chiffré $(\boldsymbol{a}, \boldsymbol{y})$ correspondant à un clair $\boldsymbol{x}$, un adversaire peut forger un nouveau chiffré correspondant au clair $\boldsymbol{x} \oplus \boldsymbol{x'}$ pour un $\boldsymbol{x'}$ de son choix en modifiant simplement le chiffré en $(\boldsymbol{a}, \boldsymbol{y} \oplus C(\boldsymbol{x'}))$. Le même genre d'attaque (en plus élaboré) s'applique probablement lorsque le code correcteur n'est pas linéaire.

Comme la sécurité au sens NM-P2-C2 est équivalente à la sécurité au sens IND-P2-C2, on en déduit que le schéma ne peut pas être sûr au sens IND-P2-C2. Reste le cas de la sécurité contre les attaques à chiffrés choisis non adaptatives (C1), que nous explorons dans la section suivante.

6.3.2 Une attaque dans le modèle IND-P0-C1

Nous allons maintenant montrer que le schéma LPN-C est vulnérable à une attaque contre son indistinguabilité lorsque l'adversaire à accès à l'oracle de déchiffrement, même de façon non-adaptative (attaque IND-P0-C1).

L'idée principale consiste à envoyer des requêtes répétées à l'oracle de déchiffrement de la forme $(\boldsymbol{a}, \boldsymbol{y}_i)$ pour un $\boldsymbol{a}$ fixé afin d'acquérir des équations linéaires approchées sur $\boldsymbol{a} \cdot M$. Considérons donc un adversaire qui envoie des requêtes $(\boldsymbol{a}, \boldsymbol{y}_i)$ à l'oracle de déchiffrement pour un $\boldsymbol{a}$ fixé quelconque et des $\boldsymbol{y}_i$ aléatoirement tirés. Lorsque $\boldsymbol{y}_i \oplus \boldsymbol{a} \cdot M$ est à une distance de Hamming d'un mot de code inférieure à t, l'oracle de déchiffrement va renvoyer un clair $\boldsymbol{x}_i$ tel que $\texttt{Hwt}(C(\boldsymbol{x}_i) \oplus \boldsymbol{y}_i \oplus \boldsymbol{a} \cdot M) \leq t$. Ceci donne une approximation de chaque bit de $\boldsymbol{a} \cdot M$ avec un paramètre de bruit inférieur à $\frac{t}{m}$. Plus précisément, nous allons montrer le lemme suivant :

Lemme 6.3. *Soit $\boldsymbol{a}$ un vecteur fixé et $j \in [\![1, m]\!]$ une position fixée. Soit p_j la probabilité, prise sur $\boldsymbol{y}_i$, et sachant qu'à la requête $(\boldsymbol{a}, \boldsymbol{y}_i)$ l'oracle de déchiffrement a retourné un vecteur $\boldsymbol{x}_i$ et non $\perp$, que le bit en position j de $\boldsymbol{a} \cdot M$ soit différent du bit en position j de $C(\boldsymbol{x}_i) \oplus \boldsymbol{y}_i$:*

$$p_j = \Pr_{\boldsymbol{y}_i \xleftarrow{\$} \{0,1\}^m} \left[(\boldsymbol{a} \cdot M)[j] \neq (C(\mathcal{D}_M(\boldsymbol{a}, \boldsymbol{y}_i)) \oplus \boldsymbol{y}_i)[j] \,\middle|\, \mathcal{D}_M(\boldsymbol{a}, \boldsymbol{y}_i) \neq \perp \right] .$$

Alors $p_j \leq \frac{t}{m}$. ▽

Démonstration. Considérons la variable aléatoire N définie comme le nombre d'erreurs $\texttt{Hwt}(C(\boldsymbol{x}_i) \oplus \boldsymbol{y}_i \oplus \boldsymbol{a} \cdot M)$. Lorsque $\mathcal{D}_M(\boldsymbol{a}, \boldsymbol{y}_i) \neq \perp$, cette variable est toujours inférieure à t. Par conséquent son espérance conditionnelle par rapport à l'événement $\mathcal{D}_M(\boldsymbol{a}, \boldsymbol{y}_i) \neq \perp$ est également inférieure à t. Or par linéarité de l'espérance on a que l'espérance est égale à la somme des p_j et par conséquent $\sum_{j=1}^{m} p_j \leq m$. Par symétrie (on peut facilement se ramener au cas où les probabilités p_j sont égales pour tout j comme dans le lemme 5.7), $p_j \leq \frac{t}{m}$. ■

On peut donc utiliser cette procédure pour obtenir des équations approchées avec le même vecteur $\boldsymbol{a}$ utilisables avec l'algorithme du lemme 3.2 pour retrouver chaque colonne de la matrice secrète M. Afin de calculer la complexité totale de la reconstruction de M, il convient de prendre en compte la probabilité que l'oracle de déchiffrement retourne bien

un clair et non $\perp$. Clairement, la probabilité qu'un vecteur $\boldsymbol{y}_i$ soit à une distance inférieure à t d'un mot de code est

$$\Pr_{\boldsymbol{y}_i \xleftarrow{\$} \{0,1\}^m} \left[\mathcal{D}_K(\boldsymbol{a}, \boldsymbol{y}_i) \neq \perp \right] = 2^r \sum_{i=0}^{t} \frac{\binom{m}{i}}{2^m} \simeq 2^{-(1-\frac{r}{m}-H_2(\frac{t}{m}))m} \ ,$$

où H_2 est la fonction entropie $H_2(x) = -x\log_2(x) - (1-x)\log_2(1-x)$. Remarquons que meilleur est le code utilisé, plus l'attaque est efficace.

Cette remarque suggère un moyen de contrer l'attaque, en « dégradant » le code. Supposons que l'on introduise un paramètre additionnel t' tel que $\eta m < t' < t$. Lorsque le nombre d'erreurs dans le vecteur $\boldsymbol{y} \oplus \boldsymbol{a} \cdot M$ est supérieur à t' (c'est-à-dire que $\boldsymbol{y} \oplus \boldsymbol{a} \cdot M$ est à une distance de Hamming plus grande que t' de n'importe quel mot de code), l'algorithme de déchiffrement retourne $\perp$. Si t' est choisi tel que $2^{-(1-\frac{r}{m}-H_2(\frac{t'}{m}))m}$ est négligeable, l'attaque devient inopérante. Cependant, il est fort probable qu'une telle variante soit peu efficace dans la pratique car elle impose de réduire énormément le bruit η pour éviter de trop fréquentes erreurs de déchiffrement. Il faut donc augmenter k pour éviter que le problème LPN ne devienne trop facile et les tailles de clés engendrées peuvent devenir prohibitives. Par ailleurs cette variante reste malléable et ne peut donc pas être sûre au sens IND-P2-C2, ce qui limite son intérêt. Le fait de savoir si elle peut être prouvée sûre au sens IND-P2-C1 est un problème ouvert.

6.3.3 Sécurité contre les attaques à clairs et chiffrés choisis

Le moyen le plus direct pour obtenir un schéma de chiffrement sûr contre les attaques à chiffrés choisis à partir d'un schéma sûr contre les attaques à clairs choisi est d'assurer l'authenticité des messages reçus par l'oracle de déchiffrement, par exemple en utilisant un Code d'Authentification de Message (MAC). Cette idée a été suggérée par [DDN00, KY06] et soigneusement étudiée par Bellare et Namprempre [BN00]. Ils ont exploré trois manières de procéder, dénommées *Encrypt-and-MAC*, *MAC-then-Encrypt* et *Encrypt-then-MAC*, et ont montré que cette dernière était la plus sûre.

Supposons que l'émetteur et le destinataire partagent une clé additionnelle K_m destinée à l'authentification de message, et soit $T(K_m, \cdot)$ une fonction de MAC sûre (c'est-à-dire fortement inforgeable contre des attaques à messages choisis). LPN-C est modifié de la façon suivante : soient $\boldsymbol{A} = (\boldsymbol{a}_1, \ldots, \boldsymbol{a}_n)$ les vecteurs utilisés pour chiffrer, et $\boldsymbol{Y} = (\boldsymbol{y}_1, \ldots, \boldsymbol{y}_n)$ les chiffrés à transmettre. Un code d'authentification $\boldsymbol{\tau}$ est ajouté à la transmission, calculé selon : $\boldsymbol{\tau} = T(K_m, \boldsymbol{A} \| \boldsymbol{Y})$. L'algorithme de déchiffrement est modifié pour retourner $\perp$ lorsque le code d'authentification n'est pas valide. Lorsque le schéma de départ est sûr au sens IND-P2-C0, les résultats génériques de [BN00] impliquent que le schéma modifié est sûr au sens IND/NM-P2-C2.

Cette méthode générique a l'inconvénient de reposer sur l'hypothèse supplémentaire de la sécurité de la fonction de MAC. On peut cependant aller une étape plus loin : nous proposons une construction de fonction de MAC reposant sur le problème LPN et une fonction de hachage. Soit M_2 une matrice binaire secrète de taille $\ell \times \ell'$, où ℓ et ℓ' sont des polynômes en k. Soit $H : \{0,1\}^* \to \{0,1\}^\ell$ une fonction de hachage. Pour $\boldsymbol{X} \in \{0,1\}^*$, définissons :

$$T(M_2, \boldsymbol{X}) = H(\boldsymbol{X}) \cdot M_2 \oplus \boldsymbol{\nu}' \ ,$$

où $\boldsymbol{\nu}'$ est un vecteur de bruit $\boldsymbol{\nu}' \leftarrow \mathtt{Ber}_{\ell',\eta}$. $\boldsymbol{\tau}$ est alors un code d'authentification valide pour un message $\boldsymbol{X}$ si $\mathtt{Hwt}(\boldsymbol{\tau} \oplus H(\boldsymbol{X}) \cdot M_2) \leq t$ pour un seuil $t > \eta\ell'$.

On peut montrer que cette fonction de MAC est sûre dans le modèle de l'oracle aléatoire pour H (remarquons que la portée d'un tel résultat est fortement limitée par l'existence de fonctions de MAC sûres bien plus simples dans le modèle de l'oracle aléatoire). Nous donnons simplement l'idée de la preuve, qui repose sur la difficulté du puzzle-MHB (cf. définition 5.5). Un attaquant essayant de forger un MAC peut dans une première phase faire des requêtes à un oracle de MAC pour des messages de son choix $\boldsymbol{X}_i$. Puisque H est modélisé comme un oracle aléatoire, les réponses reçues forment exactement les données d'un puzzle MHB $(\boldsymbol{a}_i, \boldsymbol{a}_i \cdot M_2 \oplus \boldsymbol{\nu}_i)$ associé à la matrice secrète M_2, avec $\boldsymbol{a}_i = H(\boldsymbol{X}_i)$. Forger un MAC valide pour un message $\boldsymbol{X}$ pour lequel aucun MAC valide n'a été demandé auparavant revient alors à résoudre le puzzle MHB pour le challenge aléatoire $\boldsymbol{a} = H(\boldsymbol{X})$, ce qui d'après le théorème 5.4 n'est possible qu'avec une probabilité négligeable (en supposant le problème LPN dur).

Remarquons qu'il est facile de produire un *nouveau* code d'authentification valide pour un message $\boldsymbol{X}$ pour lequel un MAC $\boldsymbol{\tau}$ a été demandé à l'oracle : il suffit d'inverser un unique bit quelconque de $\boldsymbol{\tau}$, et le code résultant $\boldsymbol{\tau}'$ sera encore un MAC valide pour $\boldsymbol{X}$ avec une forte probabilité.

6.4 Propositions de paramètres et optimisations pratiques

Nous discutons maintenant quelques exemples de paramètres pour LPN-C, ainsi que des variantes pratiques possibles. Nous définirons le facteur d'expansion du schéma comme le rapport entre la longueur du chiffré et du clair :

$$\sigma = \frac{|\text{chiffré}|}{|\text{clair}|} = \frac{m+k}{r} \quad .$$

La taille de la clé secrète est $|K| = k \cdot m$.

De nombreux compromis sont possibles lorsque l'on fixe la valeur des paramètres (k, η, m, r, d). Tout d'abord, la difficulté du problème LPN est régie par k et η, et elle est une fonction croissante de ces deux paramètres. Cependant augmenter k implique un plus grand facteur d'expansion ainsi qu'une plus grande taille de clé secrète, tandis qu'augmenter η implique d'utiliser un code avec une meilleure capacité de correction et une plus grande distance minimale, ce qui implique un plus grand facteur m/r, donc un facteur d'expansion supérieur. Enfin, les erreurs de déchiffrement sont également problématiques si ηm est trop proche de t.

Nous renvoyons à la section 3.5.7 pour des exemples de paramètres k et η menant à une sécurité donnée pour le problème LPN. Pour une sécurité de 80 bits, des paramètres adéquats sont $(k = 512, \eta = 0.125)$, ou $(k = 768, \eta = 0.05)$. Nous donnons dans le tableau 6.1 des exemples de paramètres fondés sur la table des meilleurs codes linéaires connus (*Best Known Linear Codes*) du logiciel MAGMA 2.13 [2].

Les optimisations pratiques possibles que nous entrevoyons sont au nombre de trois :

1. Si la bande passante est restreinte mais que la taille de la clé secrète importe moins, une première possibilité consiste à augmenter la taille de la matrice secrète M afin de diminuer la taille du facteur d'expansion σ. Ainsi, supposons que la matrice M est désormais de taille $k \times (n \cdot m)$ pour un entier $n > 1$. Il est alors possible de chiffrer

2. MAGMA Computational Algebra System, `http://magma.maths.usyd.edu.au/magma`

LPN-C					expansion	clé	clé	proba. erreur de
k	η	m	r	d	σ	$\lvert K\rvert$ (bits)	(Toeplitz)	déchiffrement P_{ED}
512	0.125	80	27	21	21.9	40,960	591	0.42
512	0.125	160	42	42	16	81,920	671	0.44
768	0.05	80	53	9	16	61,440	847	0.37
768	0.05	160	99	17	9.4	122,880	927	0.41
768	0.05	160	75	25	12.4	122,880	927	0.06

TABLE 6.1 – Exemples de paramètres pour le schéma LPN-C.

n blocs de r bits avec le même vecteur aléatoire $\boldsymbol{a}$. Le facteur d'expansion devient donc

$$\sigma = \frac{n \cdot m + k}{n \cdot r} \ .$$

Asymptotiquement, lorsque n croît, le facteur d'expansion tend vers celui du code correcteur m/r.

2. Une autre possibilité permettant d'économiser la bande passante consiste à partager les vecteurs $\boldsymbol{a}_i$ à l'avance entre l'émetteur et le récepteur, ou à les générer à partir d'une graine courte et d'un générateur pseudo-aléatoire. Le facteur d'expansion tombe alors à $\sigma = m/r$, cependant de nouveaux problèmes peuvent apparaître, comme la désynchronisation, etc.
3. Enfin, nous signalons la possibilité, déjà explorée dans le cas du protocole HB$^{\#}$, d'utiliser non pas une matrice secrète parfaitement aléatoire, mais une matrice de Toeplitz (cf. définition 5.6). La taille de la clé secrète est alors réduite à $k + m - 1$ bits, ce qui correspond environ à une division par un facteur 100 pour des paramètres typiques.

Remarque 6.1. Le schéma de chiffrement LPN-C a été indépendamment proposé dans deux articles récents. Applebaum *et al.* [ACPS09] ont montré que LPN-C était sûr contre les attaques à clairs choisis *dépendants de la clé* (sécurité au sens KDM, *Key-Dependent Message*). Dodis *et al.* [DKL09] ont proposé une légère variante de LPN-C et ont montré que sous une version plus forte de l'hypothèse de la difficulté du problème LPN, le schéma reste sûr même lorsque de l'information sur la clé secrète « fuit ». ∗

Bibliographie

[ACPS09] Benny Applebaum, David Cash, Chris Peikert, and Amit Sahai. Fast Cryptographic Primitives and Circular-Secure Encryption Based on Hard Learning Problems. In Shai Halevi, editor, *Advances in Cryptology - CRYPTO 2009*. Springer, 2009. To appear. 124

[ADO05] Gildas Avoine, Etienne Dysli, and Philippe Oechslin. Reducing Time Complexity in RFID Systems. In Bart Preneel and Stafford E. Tavares, editors, *Selected Areas in Cryptography - SAC 2005*, volume 3897 of *Lecture Notes in Computer Science*, pages 291–306. Springer, 2005. 112

[AKS01] Miklós Ajtai, Ravi Kumar, and D. Sivakumar. A sieve algorithm for the shortest lattice vector problem. In *Symposium on Theory of Computing - STOC 2001*, pages 601–610. ACM, 2001. 63

[AL87] Dana Angluin and Philip D. Laird. Learning From Noisy Examples. *Machine Learning*, 2(4):343–370, 1987. 58

[BB04a] Dan Boneh and Xavier Boyen. Secure Identity Based Encryption Without Random Oracles. In Matthew K. Franklin, editor, *Advances in Cryptology - CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 443–459. Springer, 2004. 13

[BB04b] Dan Boneh and Xavier Boyen. Short Signatures Without Random Oracles. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology - EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 56–73. Springer, 2004. 13

[BBP04] Mihir Bellare, Alexandra Boldyreva, and Adriana Palacio. An Uninstantiable Random-Oracle-Model Scheme for a Hybrid-Encryption Problem. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology - EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 171–188. Springer, 2004. 12

[BC08] Julien Bringer and Hervé Chabanne. Trusted-HB: A Low-Cost Version of HB^+ Secure Against Man-in-the-Middle Attacks. *IEEE Transactions on Information Theory*, 54(9):4339–4342, 2008. 91, 112

[BCD06] Julien Bringer, Hervé Chabanne, and Emmanuelle Dottax. HB^{++}: a Lightweight Authentication Protocol Secure against Some Attacks. In *International Workshop on Security, Privacy and Trust in Pervasive and Ubiquitous Computing - SecPerU 2006*, pages 28–33. IEEE Computer Society, 2006. 79, 81, 86, 87

[BDJR97] Mihir Bellare, Anand Desai, E. Jokipii, and Phillip Rogaway. A Concrete Security Treatment of Symmetric Encryption. In *Symposium on Foundations of Computer Science - FOCS '97*, pages 394–403. IEEE Computer Society, 1997. 13, 116

[BDPA07] Guido Bertoni, Joan Daemen, Michael Peeters, and Gilles Van Assche. Sponge Functions. ECRYPT Hash Workshop, May 2007. 24

[BDPA08] Guido Bertoni, Joan Daemen, Michael Peeters, and Gilles Van Assche. On the Indifferentiability of the Sponge Construction. In Nigel P. Smart, editor, *Advances in Cryptology - EUROCRYPT 2008*, volume 4965 of *Lecture Notes in Computer Science*, pages 181–197. Springer, 2008. 24

[BDPR98] Mihir Bellare, Anand Desai, David Pointcheval, and Phillip Rogaway. Relations Among Notions of Security for Public-Key Encryption Schemes. In Hugo Krawczyk, editor, *Advances in Cryptology - CRYPTO '98*, volume 1462 of *Lecture Notes in Computer Science*, pages 26–45. Springer, 1998. 116

[BF01] Dan Boneh and Matthew K. Franklin. Identity-Based Encryption from the Weil Pairing. In Joe Kilian, editor, *Advances in Cryptology - CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 213–229. Springer, 2001. 13

[BFJ+94] Avrim Blum, Merrick L. Furst, Jeffrey C. Jackson, Michael J. Kearns, Yishay Mansour, and Steven Rudich. Weakly learning DNF and characterizing statistical query learning using Fourier analysis. In *Symposium on Theory of Computing - STOC '94*, pages 253–262. ACM, 1994. 60

[BFKL93] Avrim Blum, Merrick L. Furst, Michael J. Kearns, and Richard J. Lipton. Cryptographic Primitives Based on Hard Learning Problems. In Douglas R. Stinson, editor, *Advances in Cryptology - CRYPTO '93*, volume 773 of *Lecture Notes in Computer Science*, pages 278–291. Springer, 1993. 69

[BGK+06] Lejla Batina, Jorge Guajardo, Tim Kerins, Nele Mentens, Pim Tuyls, and Ingrid Verbauwhede. An Elliptic Curve Processor Suitable For RFID-Tags. Workshop on Information and System Security - WISSec 2006, 2006. 72

[BGM06] Côme Berbain, Henri Gilbert, and Alexander Maximov. Cryptanalysis of Grain. In Matthew J. B. Robshaw, editor, *Fast Software Encryption - FSE 2006*, volume 4047 of *Lecture Notes in Computer Science*, pages 15–29. Springer, 2006. 66, 68

[BHK+99] John Black, Shai Halevi, Hugo Krawczyk, Ted Krovetz, and Phillip Rogaway. UMAC: Fast and Secure Message Authentication. In Michael J. Wiener, editor, *Advances in Cryptology - CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science*, pages 216–233. Springer, 1999. 86

[Bih94] Eli Biham. New Types of Cryptanalytic Attacks Using Related Keys. *Journal of Cryptology*, 7(4):229–246, 1994. 13

[BKL+07] Andrey Bogdanov, Lars R. Knudsen, Gregor Leander, Christof Paar, Axel Poschmann, Matthew J. B. Robshaw, Yannick Seurin, and Charlotte Vikkelsoe. PRESENT: An Ultra-Lightweight Block Cipher. In Pascal Paillier and Ingrid Verbauwhede, editors, *Cryptographic Hardware and Embedded Systems - CHES 2007*, volume 4727 of *Lecture Notes in Computer Science*, pages 450–466. Springer, 2007. 72

[BKN09] Alex Biryukov, Dmitry Khovratovich, and Ivica Nikolić. Distinguisher and Related-Key Attack on the Full AES-256. In Shai Halevi, editor, *Advances in Cryptology - CRYPTO 2009*. Springer, 2009. To appear. 14

[BKR00] Mihir Bellare, Joe Kilian, and Phillip Rogaway. The Security of the Cipher Block Chaining Message Authentication Code. *Journal of Computer and System Sciences*, 61(3):362–399, 2000. 15

[BKW03] Avrim Blum, Adam Kalai, and Hal Wasserman. Noise-tolerant learning, the parity problem, and the statistical query model. *Journal of ACM*, 50(4):506–519, 2003. Preliminary version in *Proceedings of STOC 2000*. 60, 63, 66

[Bla06] John Black. The Ideal-Cipher Model, Revisited: An Uninstantiable Block-cipher-Based Hash Function. In Matthew J. B. Robshaw, editor, *Fast Software Encryption - FSE '06*, volume 4047 of *Lecture Notes in Computer Science*, pages 328–340. Springer, 2006. 14

[BLP08] Daniel J. Bernstein, Tanja Lange, and Christiane Peters. Attacking and Defending the McEliece Cryptosystem. In Johannes Buchmann and Jintai Ding, editors, *Post-Quantum Cryptography - PQCrypto 2008*, volume 5299 of *Lecture Notes in Computer Science*, pages 31–46. Springer, 2008. 67

[BMvT78] Elwyn R. Berlekamp, Robert J. McEliece, and Henk C. A. van Tilborg. On the Inherent Intractability of Certain Coding Problems. *IEEE Transactions on Information Theory*, 24(3):384–386, 1978. 56

[BN00] Mihir Bellare and Chanathip Namprempre. Authenticated Encryption: Relations among Notions and Analysis of the Generic Composition Paradigm. In Tatsuaki Okamoto, editor, *Advances in Cryptology - ASIACRYPT 2000*, volume 1976 of *Lecture Notes in Computer Science*, pages 531–545. Springer, 2000. 122

[Bon98] Dan Boneh. The Decision Diffie-Hellman Problem. In Joe Buhler, editor, *Algorithmic Number Theory Symposium - ANTS '98*, volume 1423 of *Lecture Notes in Computer Science*, pages 48–63. Springer, 1998. 13

[BPR00] Mihir Bellare, David Pointcheval, and Phillip Rogaway. Authenticated Key Exchange Secure against Dictionary Attacks. In Bart Preneel, editor, *Advances in Cryptology - EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 139–155. Springer, 2000. 14

[BR93] Mihir Bellare and Phillip Rogaway. Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. In *ACM Conference on Computer and Communications Security*, pages 62–73, 1993. 12

[BR94] Mihir Bellare and Phillip Rogaway. Optimal Asymmetric Encryption. In Alfredo De Santis, editor, *Advances in Cryptology - EUROCRYPT '94*, volume 950 of *Lecture Notes in Computer Science*, pages 92–111. Springer, 1994. 12

[BR96] Mihir Bellare and Phillip Rogaway. The Exact Security of Digital Signatures - How to Sign with RSA and Rabin. In Ueli M. Maurer, editor, *Advances in Cryptology - EUROCRYPT '96*, volume 1070 of *Lecture Notes in Computer Science*, pages 399–416. Springer, 1996. 12

[BR06] Mihir Bellare and Thomas Ristenpart. Multi-Property-Preserving Hash Domain Extension and the EMD Transform. In Xuejia Lai and Kefei Chen, editors, *Advances in Cryptology - ASIACRYPT 2006*, volume 4284 of *Lecture Notes in Computer Science*, pages 299–314. Springer, 2006. 24, 42

[BRS02] John Black, Phillip Rogaway, and Thomas Shrimpton. Black-Box Analysis of the Block-Cipher-Based Hash-Function Constructions from PGV. In Moti Yung, editor, *Advances in Cryptology - CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 320–335. Springer, 2002. 14, 24

[BS99] Mihir Bellare and Amit Sahai. Non-malleable Encryption: Equivalence between Two Notions, and an Indistinguishability-Based Characterization. In

Michael J. Wiener, editor, *Advances in Cryptology - CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science*, pages 519–536. Springer, 1999. 116

[Cam94] Peter J. Cameron. *Combinatorics: Topics, Techniques, Algorithms*. Cambridge University Press, 1994. 55

[Can00] Ran Canetti. Security and Composition of Multiparty Cryptographic Protocols. *Journal of Cryptology*, 13(1):143–202, 2000. 18

[Can01] Ran Canetti. Universally Composable Security: A New Paradigm for Cryptographic Protocols. In *Symposium on Foundations of Computer Science - FOCS 2001*, pages 136–145. IEEE Computer Society, 2001. 18

[CC94] Anne Canteaut and Hervé Chabanne. A further improvement of the work factor in an attempt at breaking McEliece's cryptosystem. In Pascale Charpin, editor, *EUROCODE '94*, 1994. 67

[CC95] Anne Canteaut and Florent Chabaud. A New Algorithm for Finding Minimum-Weight Words in a Linear Code: Application to Narrow-Sense BCH Codes of Length 511. Technical Report 2685, INRIA, october 1995. 67

[CC98] Anne Canteaut and Florent Chabaud. A New Algorithm for Finding Minimum-Weight Words in a Linear Code: Application to McEliece's Cryptosystem and to Narrow-Sense BCH Codes of Length 511. *IEEE Transactions on Information Theory*, 44(1):367–378, 1998. 67

[CDMP05] Jean-Sébastien Coron, Yevgeniy Dodis, Cécile Malinaud, and Prashant Puniya. Merkle-Damgård Revisited: How to Construct a Hash Function. In Victor Shoup, editor, *Advances in Cryptology - CRYPTO 2005*, volume 3621 of *Lecture Notes in Computer Science*, pages 430–448. Springer, 2005. 19, 23, 24, 25

[CFS01] Nicolas Courtois, Matthieu Finiasz, and Nicolas Sendrier. How to Achieve a McEliece-Based Digital Signature Scheme. In Colin Boyd, editor, *Advances in Cryptology - ASIACRYPT 2001*, volume 2248 of *Lecture Notes in Computer Science*, pages 157–174. Springer, 2001. 69

[CGH98] Ran Canetti, Oded Goldreich, and Shai Halevi. The Random Oracle Methodology, Revisited (Preliminary Version). In *Symposium on Theory of Computing - STOC '98*, pages 209–218. ACM, 1998. Full version available at `http://arxiv.org/abs/cs.CR/0010019`. 12, 18, 23, 26, 47

[CGH04] Ran Canetti, Oded Goldreich, and Shai Halevi. On the Random-Oracle Methodology as Applied to Length-Restricted Signature Schemes. In Moni Naor, editor, *Theory of Cryptography Conference - TCC 2004*, volume 2951 of *Lecture Notes in Computer Science*, pages 40–57. Springer, 2004. 12

[Cha94] Florent Chabaud. On the Security of Some Cryptosystems Based on Error-correcting Codes. In Alfredo De Santis, editor, *Advances in Cryptology - EUROCRYPT '94*, volume 950 of *Lecture Notes in Computer Science*, pages 131–139. Springer, 1994. 67

[CHS05] Ran Canetti, Shai Halevi, and Michael Steiner. Hardness Amplification of Weakly Verifiable Puzzles. In Joe Kilian, editor, *Theory of Cryptography Conference - TCC 2005*, volume 3378 of *Lecture Notes in Computer Science*, pages 17–33. Springer, 2005. 94, 95

[CJM02] Philippe Chose, Antoine Joux, and Michel Mitton. Fast Correlation Attacks: An Algorithmic Point of View. In Lars R. Knudsen, editor, *Advances in Cryptology - EUROCRYPT 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 209–221. Springer, 2002. 66, 68

[CJP02] Jean-Sébastien Coron, Antoine Joux, and David Pointcheval. Equivalence Between The Random Oracle Model and the Random Cipher Model. Dagstuhl Seminar, 2002. 27

[CJS94] James M. Crawford, Michael. J.Kearns, and Robert E. Shapire. The Minimal Disagreement Parity Problem as a Hard Satisfiability Problem. Technical Report, February 1994. 56

[CLNY06] Donghoon Chang, Sangjin Lee, Mridul Nandi, and Moti Yung. Indifferentiable Security Analysis of Popular Hash Functions with Prefix-Free Padding. In Xuejia Lai and Kefei Chen, editors, *Advances in Cryptology - ASIACRYPT 2006*, volume 4284 of *Lecture Notes in Computer Science*, pages 283–298. Springer, 2006. 24

[CM97] Christian Cachin and Ueli M. Maurer. Unconditional Security Against Memory-Bounded Adversaries. In Burton S. Kaliski Jr., editor, *Advances in Cryptology - CRYPTO '97*, volume 1294 of *Lecture Notes in Computer Science*, pages 292–306. Springer, 1997. 11

[CMPP05] Benoît Chevallier-Mames, Duong Hieu Phan, and David Pointcheval. Optimal Asymmetric Encryption and Signature Paddings. In John Ioannidis, Angelos D. Keromytis, and Moti Yung, editors, *Applied Cryptography and Network Security - ACNS 2005*, volume 3531 of *Lecture Notes in Computer Science*, pages 254–268. Springer, 2005. 14

[CN08] Donghoon Chang and Mridul Nandi. Improved Indifferentiability Security Analysis of chopMD Hash Function. In Kaisa Nyberg, editor, *Fast Software Encryption - FSE 2008*, volume 5086 of *Lecture Notes in Computer Science*, pages 429–443. Springer, 2008. 24

[CPS08] Jean-Sébastien Coron, Jacques Patarin, and Yannick Seurin. The Random Oracle Model and the Ideal Cipher Model Are Equivalent. In David Wagner, editor, *Advances in Cryptology - CRYPTO 2008*, volume 5157 of *Lecture Notes in Computer Science*, pages 1–20. Springer, 2008. 9, 25, 42

[CRS05] David Chaum, Peter Y. A. Ryan, and Steve A. Schneider. A practical voter-verifiable election scheme. In Sabrina De Capitani di Vimercati, Paul F. Syverson, and Dieter Gollmann, editors, *European Symposium on Research in Computer Security - ESORICS 2005*, volume 3679 of *Lecture Notes in Computer Science*, pages 118–139. Springer, 2005. 119

[CS98a] Anne Canteaut and Nicolas Sendrier. Cryptoanalysis of the Original McEliece Cryptosystem. In Kazuo Ohta and Dingyi Pei, editors, *Advances in Cryptology - ASIACRYPT '98*, volume 1514 of *Lecture Notes in Computer Science*, pages 187–199. Springer, 1998. 67

[CS98b] Ronald Cramer and Victor Shoup. A Practical Public Key Cryptosystem Provably Secure Against Adaptive Chosen Ciphertext Attack. In Hugo Krawczyk, editor, *Advances in Cryptology - CRYPTO '98*, volume 1462 of *Lecture Notes in Computer Science*, pages 13–25. Springer, 1998. 13

[CTIN08] Jose Carrijo, Rafael Tonicelli, Hideki Imai, and Anderson C. A. Nascimento. A Novel Probabilistic Passive Attack on the Protocols HB and HB$^+$. ePrint

Archive Report 2008/231, 2008. Available at `http://eprint.iacr.org/2008/231.pdf`. 63, 68

[CW79] Larry Carter and Mark N. Wegman. Universal Classes of Hash Functions. *Journal of Computer and System Sciences*, 18(2):143–154, 1979. 107

[Dam89] Ivan Damgård. A Design Principle for Hash Functions. In Gilles Brassard, editor, *Advances in Cryptology - CRYPTO '89*, volume 435 of *Lecture Notes in Computer Science*, pages 416–427. Springer, 1989. 15, 24

[DDN00] Danny Dolev, Cynthia Dwork, and Moni Naor. Nonmalleable Cryptography. *SIAM Journal of Computing*, 30(2):391–437, 2000. 116, 122

[Des00] Anand Desai. The Security of All-or-Nothing Encryption: Protecting against Exhaustive Key Search. In Mihir Bellare, editor, *Advances in Cryptology - CRYPTO 2000*, volume 1880 of *Lecture Notes in Computer Science*, pages 359–375. Springer, 2000. 14

[DK07] Dang N. Duc and Kwangjo Kim. Securing HB^+ against GRS Man-in-the-Middle Attack. In *Symposium on Cryptography and Information Security - SCIS 2007*. Institute of Electronics, Information and Communication Engineers, 2007. 79, 87

[DKL09] Yevgeniy Dodis, Yael Tauman Kalai, and Shachar Lovett. On cryptography with auxiliary input. In Michael Mitzenmacher, editor, *Symposium on Theory of Computing - STOC 2009*, pages 621–630. ACM, 2009. 124

[Doo03] Matthew W. Dood. *Applications of the Discrete Fourier Transform in Information Theory and Cryptology*. PhD thesis, University of London, 2003. 66

[DP06] Yevgeniy Dodis and Prashant Puniya. On the Relation Between the Ideal Cipher and the Random Oracle Models. In Shai Halevi and Tal Rabin, editors, *Theory of Cryptography Conference - TCC 2006*, volume 3876 of *Lecture Notes in Computer Science*, pages 184–206. Springer, 2006. 26, 45, 46

[DP07] Yevgeniy Dodis and Prashant Puniya. Feistel Networks Made Public, and Applications. In Moni Naor, editor, *Advances in Cryptology - EUROCRYPT 2007*, volume 4515 of *Lecture Notes in Computer Science*, pages 534–554. Springer, 2007. 26

[DPP08] Yevgeniy Dodis, Krzysztof Pietrzak, and Prashant Puniya. A New Mode of Operation for Block Ciphers and Length-Preserving MACs. In Nigel P. Smart, editor, *Advances in Cryptology - EUROCRYPT 2008*, volume 4965 of *Lecture Notes in Computer Science*, pages 198–219. Springer, 2008. 24, 50

[DRS09] Yevgeniy Dodis, Thomas Ristenpart, and Thomas Shrimpton. Salvaging Merkle-Damgård for Practical Applications. In Antoine Joux, editor, *Advances in Cryptology - EUROCRYPT 2009*, volume 5479 of *Lecture Notes in Computer Science*, pages 371–388. Springer, 2009. 23

[EKP+07] Thomas Eisenbarth, Sandeep Kumar, Christof Paar, Axel Poschmann, and Leif Uhsadel. A Survey of Lightweight-Cryptography Implementations. *IEEE Design & Test of Computers*, 24(6):522–533, 2007. 73

[EM97] Shimon Even and Yishay Mansour. A Construction of a Cipher from a Single Pseudorandom Permutation. *Journal of Cryptology*, 10(3):151–162, 1997. 14, 49

[FDW04] Martin Feldhofer, Sandra Dominikus, and Johannes Wolkerstorfer. Strong Authentication for RFID Systems Using the AES Algorithm. In Marc Joye and Jean-Jacques Quisquater, editors, *Cryptographic Hardware and Embedded Systems - CHES 2004*, volume 3156 of *Lecture Notes in Computer Science*, pages 357–370. Springer, 2004. 72

[Fei73] Horst Feistel. Cryptography and computer privacy. *Scientific American*, 228(5):15–23, 1973. 16

[FLP08] Marc Fischlin, Anja Lehmann, and Krzysztof Pietrzak. Robust Multi-property Combiners for Hash Functions Revisited. In Ivan Damgård, editor, *International Colloquium on Automata, Languages and Programming - ICALP 2008, Track C: Security and Cryptography Foundations*, volume 5126 of *Lecture Notes in Computer Science*, pages 655–666. Springer, 2008. 24

[FMI+06] Marc P. C. Fossorier, Miodrag J. Mihaljevic, Hideki Imai, Yang Cui, and Kanta Matsuura. An Algorithm for Solving the LPN Problem and Its Application to Security Evaluation of the HB Protocols for RFID Authentication. In Rana Barua and Tanja Lange, editors, *Progress in Cryptology - INDOCRYPT 2006*, volume 4329 of *Lecture Notes in Computer Science*, pages 48–62. Springer, 2006. 68

[FS86] Amos Fiat and Adi Shamir. How to Prove Yourself: Practical Solutions to Identification and Signature Problems. In Andrew M. Odlyzko, editor, *Advances in Cryptology - CRYPTO '86*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194. Springer, 1986. 12, 71

[FS09] Dmitry Frumkin and Adi Shamir. Un-Trusted-HB: Security Vulnerabilities of Trusted-HB. ePrint Archive Report 2009/044, 2009. Available at `http://eprint.iacr.org/2009/044.pdf`. 92

[GK03] Shafi Goldwasser and Yael Tauman Kalai. On the (In)security of the Fiat-Shamir Paradigm. In *Symposium on Foundations of Computer Science - FOCS 2003*, pages 102–115. IEEE Computer Society, 2003. 12

[GL89] Oded Goldreich and Leonid A. Levin. A Hard-Core Predicate for all One-Way Functions. In *Symposium on Theory of Computing - STOC '89*, pages 25–32. ACM, 1989. 58

[GL96] Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, Baltimore, 1996. Third edition. 111

[GM84] Shafi Goldwasser and Silvio Micali. Probabilistic Encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984. 116

[GM01] Henri Gilbert and Marine Minier. New Results on the Pseudorandomness of Some Blockcipher Constructions. In Mitsuru Matsui, editor, *Fast Software Encryption - FSE 2001*, volume 2355 of *Lecture Notes in Computer Science*, pages 248–266. Springer, 2001. 50

[GMR89] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The Knowledge Complexity of Interactive Proof Systems. *SIAM Journal of Computing*, 18(1):186–208, 1989. 19

[GMZZ08] Zbigniew Golebiewski, Krzysztof Majcher, Filip Zagorski, and Marcin Zawada. Practical Attacks on HB and HB+ Protocols. ePrint Archive Report 2008/241, 2008. Available at `http://eprint.iacr.org/2008/241.pdf`. 63

[Gol01] Oded Goldreich. *Foundations of Cryptography - Volume 1, Basic Tools*. Cambridge University Press, 2001. 16, 58, 71

[GPS06] Marc Girault, Guillaume Poupard, and Jacques Stern. On the Fly Authentication and Signature Schemes Based on Groups of Unknown Order. *Journal of Cryptology*, 19(4):463–487, 2006. 72

[GQ88] Louis C. Guillou and Jean-Jacques Quisquater. A Practical Zero-Knowledge Protocol Fitted to Security Microprocessor Minimizing Both Transmission and Memory. In Christof G. Günther, editor, *Advances in Cryptology - EUROCRYPT '88*, volume 330 of *Lecture Notes in Computer Science*, pages 123–128. Springer, 1988. 12, 71

[GR04] Craig Gentry and Zulfikar Ramzan. Eliminating Random Permutation Oracles in the Even-Mansour Cipher. In Pil Joong Lee, editor, *Advances in Cryptology - ASIACRYPT 2004*, volume 3329 of *Lecture Notes in Computer Science*, pages 32–47. Springer, 2004. 49

[Gra02] Louis Granboulan. Short Signatures in the Random Oracle Model. In Yuliang Zheng, editor, *Advances in Cryptology - ASIACRYPT 2002*, volume 2501 of *Lecture Notes in Computer Science*, pages 364–378. Springer, 2002. 14

[GRS05] Henri Gilbert, Matthew J. B. Robshaw, and Hervé Sibert. An active attack against HB^+: a provably secure lightweight authentication protocol. *IEE Electronics Letters*, 41(21):1169–1170, 2005. 77, 79

[GRS08a] Henri Gilbert, Matthew J. B. Robshaw, and Yannick Seurin. Good Variants of HB^+ Are Hard to Find. In Gene Tsudik, editor, *Financial Cryptography and Data Security - FC 2008*, volume 5143 of *Lecture Notes in Computer Science*, pages 156–170. Springer, 2008. 53, 71

[GRS08b] Henri Gilbert, Matthew J. B. Robshaw, and Yannick Seurin. $HB^\#$: Increasing the Security and Efficiency of HB^+. In Nigel P. Smart, editor, *Advances in Cryptology - EUROCRYPT 2008*, volume 4965 of *Lecture Notes in Computer Science*, pages 361–378. Springer, 2008. 53

[GRS08c] Henri Gilbert, Matthew J. B. Robshaw, and Yannick Seurin. How to Encrypt with the LPN Problem. In Ivan Damgård, editor, *International Colloquium on Automata, Languages and Programming - ICALP 2008, Track C: Security and Cryptography Foundations*, volume 5126 of *Lecture Notes in Computer Science*, pages 679–690. Springer, 2008. 53

[Han05] Gerhard Hancke. A practical relay attack on ISO 14443 proximity cards. Available at `http://www.rfidblog.org.uk/hancke-rfidrelay.pdf`, 2005. 79

[Hås01] Johan Håstad. Some optimal inapproximability results. *Journal of ACM*, 48(4):798–859, 2001. 56, 57

[HB01] Nicholas J. Hopper and Manuel Blum. Secure Human Identification Protocols. In Colin Boyd, editor, *Advances in Cryptology - ASIACRYPT 2001*, volume 2248 of *Lecture Notes in Computer Science*, pages 52–66. Springer, 2001. 75

[Hir04] Shoichi Hirose. Provably Secure Double-Block-Length Hash Functions in a Black-Box Model. In Choonsik Park and Seongtaek Chee, editors, *Information Security and Cryptology - ICISC 2004*, volume 3506 of *Lecture Notes in Computer Science*, pages 330–342. Springer, 2004. 14

[Hir06] Shoichi Hirose. Some Plausible Constructions of Double-Block-Length Hash Functions. In Matthew J. B. Robshaw, editor, *Fast Software Encryption - FSE 2006*, volume 4047 of *Lecture Notes in Computer Science*, pages 210–225. Springer, 2006. 14

[HS08] Ghaith Hammouri and Berk Sunar. PUF-HB: A Tamper-Resilient HB Based Authentication Protocol. In Steven M. Bellovin, Rosario Gennaro, Angelos D. Keromytis, and Moti Yung, editors, *Applied Cryptography and Network Security – ACNS 2008*, volume 5037 of *Lecture Notes in Computer Science*, pages 346–365, 2008. 92

[HWKS98] Chris Hall, David Wagner, John Kelsey, and Bruce Schneier. Building PRFs from PRPs. In Hugo Krawczyk, editor, *Advances in Cryptology - CRYPTO '98*, volume 1462 of *Lecture Notes in Computer Science*, pages 370–389. Springer, 1998. 50

[IJK07] Russell Impagliazzo, Ragesh Jaiswal, and Valentine Kabanets. Chernoff-Type Direct Product Theorems. In *Advances in Cryptology - CRYPTO 2007*, volume 4622 of *Lecture Notes in Computer Science*, pages 500–516. Springer, 2007. 102

[IL90] Russell Impagliazzo and Leonid A. Levin. No Better Ways to Generate Hard NP Instances than Picking Uniformly at Random. In *Symposium on Foundations of Computer Science - FOCS '90*, pages 812–821. IEEE Computer Society, 1990. 69

[IYYK01] Tetsu Iwata, Tomonobu Yoshino, Tomohiro Yuasa, and Kaoru Kurosawa. Round Security and Super-Pseudorandomness of MISTY Type Structure. In Mitsuru Matsui, editor, *Fast Software Encryption - FSE 2001*, volume 2355 of *Lecture Notes in Computer Science*, pages 233–247. Springer, 2001. 50

[IZ89] Russell Impagliazzo and David Zuckerman. How to Recycle Random Bits. In *Symposium on Foundations of Computer Science - FOCS '89*, pages 248–253. IEEE Computer Society, 1989. 67

[Jac03] Jeffrey Jackson. On the Efficiency of Noise-Tolerant PAC Algorithms Derived from Statistical Queries. *Ann. Math. Artif. Intell.*, 39(3):291–313, 2003. 66

[JJ99a] Thomas Johansson and Fredrik Jönsson. Fast Correlation Attacks Based on Turbo Code Techniques. In Michael J. Wiener, editor, *Advances in Cryptology - CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science*, pages 181–197. Springer, 1999. 68

[JJ99b] Thomas Johansson and Fredrik Jönsson. Improved Fast Correlation Attacks on Stream Ciphers via Convolutional Codes. In Jacques Stern, editor, *Advances in Cryptology - EUROCRYPT '99*, volume 1592 of *Lecture Notes in Computer Science*, pages 347–362. Springer, 1999. 68

[JJ00] Thomas Johansson and Fredrik Jönsson. Fast Correlation Attacks through Reconstruction of Linear Polynomials. In Mihir Bellare, editor, *Advances in Cryptology - CRYPTO 2000*, volume 1880 of *Lecture Notes in Computer Science*, pages 300–315. Springer, 2000. 68

[JJV02] Éliane Jaulmes, Antoine Joux, and Frédéric Valette. On the Security of Randomized CBC-MAC Beyond the Birthday Paradox Limit: A New Construction. In Joan Daemen and Vincent Rijmen, editors, *Fast Software Encryption - FSE 2002*, volume 2365 of *Lecture Notes in Computer Science*, pages 237–251. Springer, 2002. 14

[Jou00] Antoine Joux. A One Round Protocol for Tripartite Diffie-Hellman. In Wieb Bosma, editor, *Algorithmic Number Theory Symposium - ANTS 2000*, volume 1838 of *Lecture Notes in Computer Science*, pages 385–394. Springer, 2000. 13

[Jou02] Antoine Joux. The Weil and Tate Pairings as Building Blocks for Public Key Cryptosystems. In Claus Fieker and David R. Kohel, editors, *Algorithmic*

Number Theory Symposium - ANTS '02, volume 2369 of *Lecture Notes in Computer Science*, pages 20–32. Springer, 2002. 13

[Jou04] Antoine Joux. Multicollisions in Iterated Hash Functions. Application to Cascaded Constructions. In Matthew K. Franklin, editor, *Advances in Cryptology - CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 306–316. Springer, 2004. 49

[JW05a] Ari Juels and Stephen A. Weis. Authenticating Pervasive Devices with Human Protocols. In Victor Shoup, editor, *Advances in Cryptology - CRYPTO 2005*, volume 3621 of *Lecture Notes in Computer Science*, pages 293–308. Springer, 2005. 76, 77, 96, 134

[JW05b] Ari Juels and Stephen A. Weis. Authenticating Pervasive Devices with Human Protocols. Available at `http://saweis.net/pdfs/lpn-paper.pdf`, 2005. Full version of [JW05a]. 79, 98

[JW07] Ari Juels and Stephen A. Weis. Defining Strong Privacy for RFID. In *IEEE International Conference on Pervasive Computing and Communications - PerCom Workshops 2007*, pages 342–347. IEEE Computer Society, 2007. 112

[Kar72] Richard M. Karp. Reducibility Among Combinatorial Problems. In Raymond E. Miller and James W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. New York: Plenum, 1972. 56

[Kea98] Michael J. Kearns. Efficient Noise-Tolerant Learning from Statistical Queries. *Journal of ACM*, 45(6):983–1006, 1998. 59, 60

[KM07] Neal Koblitz and Alfred Menezes. Another Look at "Provable Security". *Journal of Cryptology*, 20(1):3–37, 2007. 12

[KR96] Joe Kilian and Phillip Rogaway. How to Protect DES Against Exhaustive Key Search. In Neal Koblitz, editor, *Advances in Cryptology - CRYPTO '96*, volume 1109 of *Lecture Notes in Computer Science*, pages 252–267. Springer, 1996. 14

[KR07] Lars R. Knudsen and Vincent Rijmen. Known-Key Distinguishers for Some Block Ciphers. In Kaoru Kurosawa, editor, *Advances in Cryptology - ASIACRYPT 2007*, volume 4833 of *Lecture Notes in Computer Science*, pages 315–324. Springer, 2007. 26, 48

[Kra94] Hugo Krawczyk. LFSR-based Hashing and Authentication. In Yvo Desmedt, editor, *Advances in Cryptology - CRYPTO '94*, volume 839 of *Lecture Notes in Computer Science*, pages 129–139. Springer, 1994. 91, 106, 111

[Kra95] Hugo Krawczyk. New Hash Functions For Message Authentication. In Louis C. Guillou and Jean-Jacques Quisquater, editors, *Advances in Cryptology - EUROCRYPT '95*, volume 921 of *Lecture Notes in Computer Science*, pages 301–310. Springer, 1995. 106

[KS01] Ravi Kumar and D. Sivakumar. On polynomial approximation to the shortest lattice vector length. In *Symposium on Discrete Algorithms - SODA 2001*, pages 126–127, 2001. 63

[KS06a] Jonathan Katz and Ji Sun Shin. Parallel and Concurrent Security of the HB and HB^+ Protocols. In Serge Vaudenay, editor, *Advances in Cryptology - EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 73–87. Springer, 2006. 60, 76, 77, 94, 108

[KS06b] Jonathan Katz and Adam Smith. Analysing the HB and HB$^+$ Protocols in the "Large Error" Case. ePrint Archive Report 2006/326, 2006. Available at `http://eprint.iacr.org/2006/326.pdf`. 76, 77, 108

[KW05] Ziv Kfir and Avishai Wool. Picking Virtual Pockets using Relay Attacks on Contactless Smartcard. In *Security and Privacy for Emerging Areas in Communications Networks - SecureComm 2005*, pages 47–58. IEEE Computer Society, Sept. 2005. 79

[KW06] Ilan Kirschenbaum and Avishai Wool. How to build a low-cost, extended-range RFID skimmer. In *USENIX Security Symposium*, pages 43–57. USENIX Association, 2006. 79

[KY06] Jonathan Katz and Moti Yung. Characterization of Security Notions for Probabilistic Private-Key Encryption. *Journal of Cryptology*, 19(1):67–95, 2006. Preliminary version in *Proceedings of STOC 2000*. 115, 116, 117, 122

[KYS05] Jens-Peter Kaps, Kaan Yüksel, and Berk Sunar. Energy Scalable Universal Hashing. *IEEE Trans. Computers*, 54(12):1484–1495, 2005. 86

[Lai88] Philip D. Laird. *Learning from good and bad data*. Kluwer International Series In Engineering And Computer Science. Kluwer Academic Publishers, 1988. 59

[LB88] Pil Joong Lee and Ernest F. Brickell. An Observation on the Security of McEliece's Public-Key Cryptosystem. In Christof G. Günther, editor, *Advances in Cryptology - EUROCRYPT '88*, volume 330 of *Lecture Notes in Computer Science*, pages 275–280. Springer, 1988. 67

[Leo88] Jeffrey S. Leon. A probabilistic algorithm for computing minimum weights of large error-correcting codes. *IEEE Transactions on Information Theory*, 34(5):1354–1359, 1988. 67

[Lev08] Éric Levieil. *Contributions à l'étude cryptographique de protocoles et de primitives à clé secrète*. PhD thesis, Université Paris 7, 2008. 66, 68, 69, 110

[LF06] Éric Levieil and Pierre-Alain Fouque. An Improved LPN Algorithm. In Roberto De Prisco and Moti Yung, editors, *Security and Cryptography for Networks - SCN 2006*, volume 4116 of *Lecture Notes in Computer Science*, pages 348–359. Springer, 2006. 66, 110

[LR86] Michael Luby and Charles Rackoff. Pseudo-random Permutation Generators and Cryptographic Composition. In *Symposium on Theory of Computing - STOC '86*, pages 356–363. ACM, 1986. 13

[LR88] Michael Luby and Charles Rackoff. How to Construct Pseudorandom Permutations from Pseudorandom Functions. *SIAM Journal of Computing*, 17(2):373–386, 1988. 17, 25, 27

[Lub96] Michael Luby. *Pseudorandomness and Cryptographic Applications*. Princeton Computer Science Notes. Princeton University Press, 1996. 58

[Luc00] Stefan Lucks. The Sum of PRPs Is a Secure PRF. In Bart Preneel, editor, *Advances in Cryptology - EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 470–484. Springer, 2000. 50

[Mat97] Mitsuru Matsui. New Block Encryption Algorithm MISTY. In Eli Biham, editor, *Fast Software Encryption - FSE '97*, volume 1267 of *Lecture Notes in Computer Science*, pages 54–68. Springer, 1997. 50

[Mau92] Ueli M. Maurer. A Simplified and Generalized Treatment of Luby-Rackoff Pseudorandom Permutation Generator. In Rainer A. Rueppel, editor, *Advances in Cryptology - EUROCRYPT '92*, volume 658 of *Lecture Notes in Computer Science*, pages 239–255. Springer, 1992. 17

[Mau99] Ueli M. Maurer. Information-Theoretic Cryptography. In Michael J. Wiener, editor, *Advances in Cryptology - CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science*, pages 47–64. Springer, 1999. 11

[McE78] Robert J. McEliece. A public-key cryptosystem based on algebraic coding theory. DSN progress report, Jet Propulsion Laboratory, Pasadena, California, 1978. 67, 69

[Mer89] Ralph C. Merkle. One Way Hash Functions and DES. In Gilles Brassard, editor, *Advances in Cryptology - CRYPTO '89*, volume 435 of *Lecture Notes in Computer Science*, pages 428–446. Springer, 1989. 14, 15, 24

[MFI00] Miodrag J. Mihaljevic, Marc P. C. Fossorier, and Hideki Imai. A Low-Complexity and High-Performance Algorithm for the Fast Correlation Attack. In Bruce Schneier, editor, *Fast Software Encryption - FSE 2000*, volume 1978 of *Lecture Notes in Computer Science*, pages 196–212. Springer, 2000. 68

[MFI01] Miodrag J. Mihaljevic, Marc P. C. Fossorier, and Hideki Imai. Fast Correlation Attack Algorithm with List Decoding and an Application. In Mitsuru Matsui, editor, *Fast Software Encryption - FSE 2001*, volume 2355 of *Lecture Notes in Computer Science*, pages 196–210. Springer, 2001. 68

[MNT90] Yishay Mansour, Noam Nisan, and Prasoon Tiwari. The Computational Complexity of Universal Hashing. In *Symposium on Theory of Computing - STOC '90*, pages 235–243. ACM, 1990. 107

[MP03] Ueli M. Maurer and Krzysztof Pietrzak. The Security of Many-Round Luby-Rackoff Pseudo-Random Permutations. In Eli Biham, editor, *Advances in Cryptology - EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 544–561. Springer, 2003. 17

[MP07] J. Munilla and A. Peinado. HB-MP: A further step in the HB-family of lightweight authentication protocols. *Computer Networks*, 51(9):2262–2267, 2007. 79, 90

[MR07] Máire McLoone and Matthew J. B. Robshaw. Public Key Cryptography and RFID Tags. In Masayuki Abe, editor, *Topics in Cryptology - CT-RSA 2007*, volume 4377 of *Lecture Notes in Computer Science*, pages 372–384. Springer, 2007. 72

[MRH04] Ueli M. Maurer, Renato Renner, and Clemens Holenstein. Indifferentiability, Impossibility Results on Reductions, and Applications to the Random Oracle Methodology. In Moni Naor, editor, *Theory of Cryptography Conference-TCC 2004*, volume 2951 of *Lecture Notes in Computer Science*, pages 21–39. Springer, 2004. 18, 19, 20, 23

[MS77] Florence J. MacWilliams and Neil J. A. Sloane. *The Theory of Error-Correcting Codes*. North-Holland Mathematical Library, 1977. 145

[MS88] Willi Meier and Othmar Staffelbach. Fast Correltaion Attacks on Stream Ciphers (Extended Abstract). In Christof G. Günther, editor, *Advances in Cryptology - EUROCRYPT '88*, volume 330 of *Lecture Notes in Computer Science*, pages 301–314. Springer, 1988. 68

[MS89] Willi Meier and Othmar Staffelbach. Fast Correlation Attacks on Certain Stream Ciphers. *Journal of Cryptology*, 1(3):159–176, 1989. 68

[MT07] Ueli M. Maurer and Stefano Tessaro. Domain Extension of Public Random Functions: Beyond the Birthday Barrier. In Alfred Menezes, editor, *Advances in Cryptology - CRYPTO 2007*, volume 4622 of *Lecture Notes in Computer Science*, pages 187–204. Springer, 2007. 24

[MvOV96] Alfred Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996. 13, 72

[Nat95] National Institute of Standards and Technology. FIPS 180-1: Secure Hash Standard, April 1995. 12

[Nat01] National Institute of Standards and Technology. FIPS 197: Advanced Encryption Standard, November 2001. 13

[Nie86] Harald Niederreiter. Knapsack-type crytosystems and algebraic coding theory. *Problems of Control and Information Theory*, 15(2):157–166, 1986. 69

[Nie02] Jesper Buus Nielsen. Separating Random Oracle Proofs from Complexity Theoretic Proofs: The Non-committing Encryption Case. In Moti Yung, editor, *Advances in Cryptology - CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 111–126. Springer, 2002. 12

[NR99] Moni Naor and Omer Reingold. On the Construction of Pseudorandom Permutations: Luby-Rackoff Revisited. *Journal of Cryptology*, 12(1):29–66, 1999. 17

[NY89] Moni Naor and Moti Yung. Universal One-Way Hash Functions and their Cryptographic Applications. In *Symposium on Theory of Computing - STOC '89*, pages 33–43. ACM, 1989. 13

[Oka92] Tatsuaki Okamoto. Provably Secure and Practical Identification Schemes and Corresponding Signature Schemes. In Ernest F. Brickell, editor, *Advances in Cryptology - CRYPTO '92*, volume 740 of *Lecture Notes in Computer Science*, pages 31–53. Springer, 1992. 12

[OOV08] Khaled Ouafi, Raphael Overbeck, and Serge Vaudenay. On the Security of HB$^{\#}$ Against a Man-in-the-Middle Attack. In Josef Pieprzyk, editor, *Advances in Cryptology - ASIACRYPT 2008*, volume 5350 of *Lecture Notes in Computer Science*, pages 108–124. Springer, 2008. 108, 109, 110, 111

[Pat90] Jacques Patarin. Pseudorandom Permutations Based on the DES Scheme. In Gérard D. Cohen and Pascale Charpin, editors, *EUROCODE '90*, volume 514 of *Lecture Notes in Computer Science*, pages 193–204. Springer, 1990. 17, 25, 27

[Pat91] Jacques Patarin. New Results on Pseudorandom Permutation Generators Based on the DES Scheme. In Joan Feigenbaum, editor, *Advances in Cryptology - CRYPTO '91*, volume 576 of *Lecture Notes in Computer Science*, pages 301–312. Springer, 1991. 17

[Pat98] Jacques Patarin. About Feistel Schemes with Six (or More) Rounds. In Serge Vaudenay, editor, *Fast Software Encryption - FSE '98*, volume 1372 of *Lecture Notes in Computer Science*, pages 103–121. Springer, 1998. 17

[Pat03] Jacques Patarin. Luby-Rackoff: 7 Rounds Are Enough for $2^{n(1-\epsilon)}$ Security. In Dan Boneh, editor, *Advances in Cryptology - CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 513–529. Springer, 2003. 17

[Pat04] Jacques Patarin. Security of Random Feistel Schemes with 5 or More Rounds. In Matthew K. Franklin, editor, *Advances in Cryptology - CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 106–122. Springer, 2004. 17

[PGV93] Bart Preneel, René Govaerts, and Joos Vandewalle. Hash Functions Based on Block Ciphers: A Synthetic Approach. In Douglas R. Stinson, editor, *Advances in Cryptology - CRYPTO '93*, volume 773 of *Lecture Notes in Computer Science*, pages 368–378. Springer, 1993. 24

[Pir06a] Selwyn Piramuthu. HB and Related Lightweight Authentication Protocols for Secure RFID Tag/Reader Authentication. CollECTeR Europe Conference, June 2006. 82

[Pir06b] Gilles Piret. Luby-Rackoff Revisited: On the Use of Permutations as Inner Functions of a Feistel Scheme. *Designs, Codes and Cryptography*, 39(2):233–245, 2006. 50

[PP03] Duong Hieu Phan and David Pointcheval. Chosen-Ciphertext Security without Redundancy. In Chi-Sung Laih, editor, *Advances in Cryptology - ASIACRYPT 2003*, volume 2894 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 2003. 14, 49

[PS00] David Pointcheval and Jacques Stern. Security Arguments for Digital Signatures and Blind Signatures. *Journal of Cryptology*, 13(3):361–396, 2000. 12, 143

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *Symposium on Theory of Computing - STOC 2005*, pages 84–93. ACM, 2005. 60, 61, 69

[RPLP08] Carsten Rolfes, Axel Poschmann, Gregor Leander, and Christof Paar. Ultra-Lightweight Implementations for Smart Devices - Security for 1000 Gate Equivalents. In Gilles Grimaud and François-Xavier Standaert, editors, *Smart Card Research and Advanced Applications - CARDIS 2008*, volume 5189 of *Lecture Notes in Computer Science*, pages 89–103. Springer, 2008. 72

[RR00] Zulfikar Ramzan and Leonid Reyzin. On the Round Security of Symmetric-Key Cryptographic Primitives. In Mihir Bellare, editor, *Advances in Cryptology - CRYPTO 2000*, volume 1880 of *Lecture Notes in Computer Science*, pages 376–393. Springer, 2000. 25

[RST01] Ronald L. Rivest, Adi Shamir, and Yael Tauman. How to Leak a Secret. In Colin Boyd, editor, *Advances in Cryptology - ASIACRYPT 2001*, volume 2248 of *Lecture Notes in Computer Science*, pages 552–565. Springer, 2001. 14

[Sch89] Claus-Peter Schnorr. Efficient Identification and Signatures for Smart Cards. In Gilles Brassard, editor, *Advances in Cryptology - CRYPTO '89*, volume 435 of *Lecture Notes in Computer Science*, pages 239–252. Springer, 1989. 71

[Sch91] Claus-Peter Schnorr. Efficient Signature Generation by Smart Cards. *Journal of Cryptology*, 4(3):161–174, 1991. 12

[Sha49] Claude Shannon. Communication Theory of Secrecy Systems. *Bell System Technical Journal*, 28(4):656–715, 1949. 13, 116

[Sha08] Adi Shamir. SQUASH - A New MAC with Provable Security Properties for Highly Constrained Devices Such as RFID Tags. In Kaisa Nyberg, editor, *Fast Software Encryption - FSE 2008*, volume 5086 of *Lecture Notes in Computer Science*, pages 144–157. Springer, 2008. 73

[Sie84] Thomas Siegenthaler. Correlation-immunity of nonlinear combining functions for cryptographic applications. *IEEE Transactions on Information Theory*, 30(5):776–780, 1984. 68

[Sie85] Thomas Siegenthaler. Decrypting a Class of Stream Ciphers Using Ciphertext Only. *IEEE Transactions on Computers*, 34(1):81–85, 1985. 68

[Sim98] Daniel R. Simon. Finding Collisions on a One-Way Street: Can Secure Hash Functions Be Based on General Assumptions? In Kaisa Nyberg, editor, *Advances in Cryptology - EUROCRYPT '98*, volume 1403 of *Lecture Notes in Computer Science*, pages 334–345. Springer, 1998. 13

[Ste88] Jacques Stern. A method for finding codewords of small weight. In Gérard D. Cohen and Jacques Wolfmann, editors, *Coding Theory and Applications*, volume 388 of *Lecture Notes in Computer Science*, pages 106–113. Springer, 1988. 67

[Ste93] Jacques Stern. A New Identification Scheme Based on Syndrome Decoding. In Douglas R. Stinson, editor, *Advances in Cryptology - CRYPTO '93*, volume 773 of *Lecture Notes in Computer Science*, pages 13–21. Springer, 1993. 69

[Ste07] John P. Steinberger. The Collision Intractability of MDC-2 in the Ideal-Cipher Model. In Moni Naor, editor, *Advances in Cryptology - EUROCRYPT 2007*, volume 4515 of *Lecture Notes in Computer Science*, pages 34–51. Springer, 2007. 14

[Sti01] Douglas R. Stinson, 2001. Available at `http://www.cacr.math.uwaterloo.ca/~dstinson/papers/polemic.ps`. 12

[Val84] Leslie G. Valiant. A Theory of the Learnable. *Commun. ACM*, 27(11):1134–1142, 1984. 58

[Vau03] Serge Vaudenay. Decorrelation: A Theory for Block Cipher Security. *Journal of Cryptology*, 16(4):249–286, 2003. 17

[Vau07] Serge Vaudenay. On Privacy Models for RFID. In Kaoru Kurosawa, editor, *Advances in Cryptology - ASIACRYPT 2007*, volume 4833 of *Lecture Notes in Computer Science*, pages 68–87. Springer, 2007. 112

[vT88] Johan van Tilburg. On the McEliece Public-Key Cryptosystem. In Shafi Goldwasser, editor, *Advances in Cryptology - CRYPTO '88*, volume 403 of *Lecture Notes in Computer Science*, pages 119–131. Springer, 1988. 67

[Wat05] Brent Waters. Efficient Identity-Based Encryption Without Random Oracles. In Ronald Cramer, editor, *Advances in Cryptology - EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 114–127. Springer, 2005. 13

[Win84] Robert S. Winternitz. A Secure One-Way Hash Function Built from DES. In *IEEE Symposium on Security and Privacy*, pages 88–90, 1984. 14

[WSI03] Yodai Watanabe, Junji Shikata, and Hideki Imai. Equivalence between Semantic Security and Indistinguishability against Chosen Ciphertext Attacks. In Yvo Desmedt, editor, *Public Key Cryptography - PKC 2003*, volume 2567 of *Lecture Notes in Computer Science*, pages 71–84. Springer, 2003. 116

[Yao82] Andrew C. Yao. Theory and Applications of Trapdoor Functions (Extended Abstract). In *Symposium on Foundations of Computer Science - FOCS '82*, pages 80–91. IEEE Computer Society, 1982. 94

Annexe A

Vocabulaire Asymptotique

Nous introduisons dans cet appendice les différentes définitions concernant le comportement asymptotique d'une fonction utilisées dans ce mémoire.

Par abus de langage, une fonction f de $\mathbb{N}$ vers $\mathbb{R}^+$ est dite polynomiale si elle est bornée à l'infini par un monôme :

$$\exists\, c > 0, \exists\, n_0 \in \mathbb{N} \text{ tels que } f(n) \leq n^c \text{ pour tout } n \geq n_0 \ .$$

On notera alors $f \in \texttt{poly}(n)$.

Une fonction f de $\mathbb{N}$ vers $\mathbb{R}^+$ est dite négligeable si elle décroit plus vite que n'importe quel polynôme :

$$\forall\, c > 0, \exists\, n_0 \in \mathbb{N} \text{ tel que } f(n) \leq \frac{1}{n^c} \text{ pour tout } n \geq n_0 \ .$$

On notera alors $f \in \texttt{negl}(n)$.

Une fonction f de $\mathbb{N}$ vers $\mathbb{R}^+$ est dite notable si pour n suffisamment grand elle est supérieure à l'inverse d'un monôme :

$$\exists\, c > 0, \exists\, n_0 \in \mathbb{N} \text{ tels que } f(n) \geq \frac{1}{n^c} \text{ pour tout } n \geq n_0 \ .$$

On a l'équivalence f est notable $\Leftrightarrow 1/f \in \texttt{poly}(n)$.

Remarquons qu'une fonction non-négligeable n'est pas forcément notable ! Ainsi, la fonction $\mu : n \mapsto (n \mod 2)$ n'est ni notable ni négligeable. Une fonction est non-négligeable s'il existe $c > 0$ tel que $f(n) > \frac{1}{n^c}$ pour une infinité de $n \in \mathbb{N}$.

Étant données deux fonction f et g de $\mathbb{N}$ vers $\mathbb{R}^+$, on dira que :
- f est asymptotiquement dominée par g, noté $f = \mathcal{O}(g)$, s'il existe $c > 0$ et $n_0 \in \mathbb{N}$ tels que $f(n) \leq c \cdot g(n)$ pour tout $n \geq n_0$;
- f domine asymptotiquement g, noté $f = \Omega(g)$, s'il existe $c > 0$ et $n_0 \in \mathbb{N}$ tels que $f(n) \geq c \cdot g(n)$ pour tout $n \geq n_0$;
- f est asymptotiquement équivalente à g, noté $f = \Theta(g)$, si $f = \mathcal{O}(g)$ et $f = \Omega(g)$;
- f est asymptotiquement négligeable par rapport à g, noté $f = o(g)$, si pour tout $c > 0$, il existe $n_0 \in \mathbb{N}$ tel que $f(n) < c \cdot g(n)$ pour tout $n \geq n_0$;
- f est asymptotiquement prépondérante par rapport à g, noté $f = \omega(g)$, si pour tout $c > 0$, il existe $n_0 \in \mathbb{N}$ tel que $f(n) > c \cdot g(n)$ pour tout $n \geq n_0$.

Annexe B

Probabilités

B.1 Lemme de séparation des espaces de probabilité

Nous utilisons souvent dans ce mémoire le lemme suivant, connu sous le nom de *Splitting Lemma* [PS00], permettant lors d'une expérience aléatoire de séparer les « bonnes» instances des « mauvaises » :

Lemme B.1. *Soit $\Omega = A \times B$ un espace de probabilité et* Pr *une probabilité sur cet espace. Soit $E \subset \Omega$ un événement tel que* $\Pr[E] \geq \delta$*. Soient ϵ_1, ϵ_2 deux réels tels que $\delta = \epsilon_1 + \epsilon_2$. Alors il existe $A' \subset A$ tel que :*

$$\begin{cases} \Pr_{(a,b)\leftarrow\Omega}[a \in A'] \geq \epsilon_1 \\ \Pr_{(a,b)\leftarrow\Omega}[E \,|\, a \in A'] \geq \epsilon_2 \ . \end{cases} \qquad \triangledown$$

B.2 Bornes de Chernoff

Les bornes de Chernoff sont très utiles pour majorer la probabilité des « queues de distributions » de la somme de variables de Bernoulli indépendantes.

Soient $X_1, \ldots, , X_n$ des variables de Bernoulli indépendantes telles que $\Pr[X_i = 1] = p_i$ et $\Pr[X_i = 0] = 1 - p_i$. Soit $X = \sum_{i=1}^{n} X_i$. Nous noterons μ la valeur moyenne de la variable aléatoire X :

$$\mu = \sum_{i=1}^{n} p_i \ .$$

Soit $\delta \in]0, 1[$. Les bornes de Chernoff permettent de majorer la probabilité que X soit supérieur à $(1+\delta)\mu$ ou inférieur à $(1-\delta)\mu$. Elles s'expriment de la façon suivante :

$$\Pr[X \leq (1-\delta)\mu] \leq e^{-\frac{\mu\delta^2}{2}} \quad \text{et} \quad \Pr[X \geq (1+\delta)\mu] \leq e^{-\frac{\mu\delta^2}{3}} \ .$$

On utilisera également souvent l'écart en valeur absolue :

$$\Pr[|X - \mu| \geq \delta\mu] \leq 2e^{-\frac{\mu\delta^2}{3}} \ .$$

Une forme équivalente est que pour tout $t < \mu$ et $t' > \mu$,

$$\Pr[X \leq t] \leq e^{-\frac{(\mu-t)^2}{2\mu}} \quad \text{et} \quad \Pr[X \geq t'] \leq e^{-\frac{(t'-\mu)^2}{3\mu}} \ .$$

Il est possible de symétriser ces bornes en considérant les variables $X'_i = 1 - X_i$ et en appliquant les deux inégalités précédentes à $X' = \sum_{i=1}^{n} X'_i$. On obtient alors

$$\Pr[X \leq t] \leq \min\left(e^{-\frac{(\mu-t)^2}{2\mu}}, e^{-\frac{(\mu-t)^2}{3(n-\mu)}}\right) \quad \text{et} \quad \Pr[X \geq t'] \leq \min\left(e^{-\frac{(t'-\mu)^2}{3\mu}}, e^{-\frac{(t'-\mu)^2}{2(n-\mu)}}\right) \ .$$

Application à l'estimation de la probabilité de succès du vote majoritaire. Considérons le problème suivant. On a accès à un oracle $\mathcal{O}_a$ qui retourne $a = 0$ ou 1, mais de façon biaisée : l'oracle $\mathcal{O}_0$ retourne 0 avec probabilité $1/2 + \epsilon$ et 1 avec probabilité $1/2 - \epsilon$ et l'oracle $\mathcal{O}_1$ retourne 1 avec probabilité $1/2 + \epsilon$ et 0 avec probabilité $1/2 - \epsilon$ (pour $\epsilon \in]0, \frac{1}{2}[$). On veut déterminer si l'on a accès à $\mathcal{O}_0$ ou $\mathcal{O}_1$. Pour ce faire, on effectue n requêtes à l'oracle et on procède à un vote majoritaire : si on a obtenu plus de 0 on parie que l'on a accès à $\mathcal{O}_0$, sinon à $\mathcal{O}_1$.

La probabilité d'erreur de cette stratégie est donnée par les bornes de Chernoff. En effet, supposons que l'on ait accès à $\mathcal{O}_0$ (le raisonnement est similaire pour $\mathcal{O}_1$). Si X_i désigne la réponse de l'oracle à la i-ième requête, la moyenne de la variable $X = \sum_{i=1}^{n} X_i$ vaut $\mu = \left(\frac{1}{2} - \epsilon\right) n$, et le vote majoritaire entraine une erreur lorsque $X > n/2$. Par conséquent la probabilité d'erreur est inférieure à

$$\min\left(e^{-\frac{2n\epsilon^2}{3(1-2\epsilon)}}, e^{-\frac{n\epsilon^2}{(1+2\epsilon)}}\right) \ .$$

Un majorant commun simple pour les deux termes est

$$e^{-\frac{n\epsilon^2}{2}} \ .$$

On voit donc que la probabilité d'erreur décroît exponentiellement avec le nombre de requêtes. Pour obtenir une probabilité d'erreur inférieure à η, il suffit de prendre

$$n = -2\epsilon^{-2} \ln \eta \ .$$

B.3 Inégalité de Jensen

Soit f une fonction convexe de I dans $\mathbb{R}$, I étant un intervalle de $\mathbb{R}$.

Soient $(x_1, \ldots, x_n) \in I^n$ et soit $(\lambda_1, \ldots, \lambda_n)$ une famille de réels de $[0, 1]$ vérifiant $\sum_{i=1}^{n} \lambda_i = 1$. Alors l'inégalité de Jensen exprime que :

$$f\left(\sum_{i=1}^{n} \lambda_i x_i\right) \leq \sum_{i=1}^{n} \lambda_i f(x_i) \ .$$

Application aux probabilités. Soit f une fonction convexe, et X une variable aléatoire. Alors on a :

$$f(\mathbb{E}[X]) \leq \mathbb{E}[f(X)] \ .$$

Annexe C

Estimation des coefficients binomiaux et de leur somme

Soit H_2 la fonction entropie

$$H_2(x) = -x\log(x) - (1-x)\log(1-x) \ .$$

Soit n un entier et $\lambda \in]0,1[$ tel que λn soit un entier. On a alors l'encadrement suivant :

$$\frac{2^{nH_2(\lambda)}}{\sqrt{8n\lambda(1-\lambda)}} \leq \binom{n}{\lambda n} \leq \frac{2^{nH_2(\lambda)}}{\sqrt{2\pi n\lambda(1-\lambda)}} \ .$$

On en déduit, pour $\lambda \in \left]\frac{1}{2},1\right[$ et $\mu \in \left]0,\frac{1}{2}\right[$, les encadrements suivants pour les sommes marginales de coefficients binomiaux :

$$\frac{2^{nH_2(\lambda)}}{\sqrt{8n\lambda(1-\lambda)}} \leq \sum_{i=\lambda n}^{n} \binom{n}{i} \leq 2^{nH_2(\lambda)}$$

et

$$\frac{2^{nH_2(\mu)}}{\sqrt{8n\mu(1-\mu)}} \leq \sum_{i=0}^{\mu n} \binom{n}{i} \leq 2^{nH_2(\mu)} \ .$$

La démonstration de ces résultats peut être trouvée au chapitre 10 de [MS77].

Annexe D

Pseudo-code du simulateur de la construction de Luby-Rackoff à 10 tours

Algorithme 1 Simulateur

for $i = 1$ to 10 **do**
  $F_i \leftarrow \emptyset$
end for
$N \leftarrow 0$ {nombre total d'appels à $\texttt{CompleteChain}_2$ et $\texttt{CompleteChain}_9$ déjà effectués}
À la réception d'une requête (i, U) du distingueur **do** $\texttt{Query}(i, U)$

Algorithme 2 $\texttt{Query}(i, U)$

if $U \notin F_i$ **then**
  $F_i(U) \xleftarrow{\$} \{0,1\}^n$
  if $i = 2$ **then**
    $\Omega_2 \leftarrow \{U\}$, $\Omega_9 \leftarrow \emptyset$
    $\texttt{CompleteExtCh}(\Omega_2, \Omega_9)$
  else if $i = 5$ **then**
    $\Omega_5 \leftarrow \{U\}$, $\Omega_6 \leftarrow \emptyset$
    $\texttt{CompleteCenter}(\Omega_5, \Omega_6)$
  else if $i = 6$ **then**
    $\Omega_5 \leftarrow \emptyset$, $\Omega_6 \leftarrow \{U\}$
    $\texttt{CompleteCenter}(\Omega_5, \Omega_6)$
  else if $i = 9$ **then**
    $\Omega_2 \leftarrow \emptyset$, $\Omega_9 \leftarrow \{U\}$
    $\texttt{CompleteExtCh}(\Omega_2, \Omega_9)$
  end if
end if
return $F_i(U)$

Algorithme 3 `CompleteExtCh`(Ω_2, Ω_9)

$\Omega_5 \leftarrow \emptyset$
$\Omega_6 \leftarrow \emptyset$
for all $W \in \Omega_2$ **do**
  for all $(R, S, D) \in \texttt{Chain}_2(W)$ **do**
    $(\omega_5, \omega_6) \leftarrow \texttt{CompleteChain}_2(W, R, S, D)$ $\{\omega_5 = \emptyset$ ou $\{Z\}$, $\omega_6 = \emptyset$ ou $\{A\}\}$
    $\Omega_5 \leftarrow \Omega_5 \cup \omega_5$
    $\Omega_6 \leftarrow \Omega_5 \cup \omega_6$
  end for
end for
for all $D \in \Omega_9$ **do**
  for all $(S, R, W) \in \texttt{Chain}_9(D)$ **do**
    $(\omega_5, \omega_6) \leftarrow \texttt{CompleteChain}_9(D, S, R, W)$ $\{\omega_5 = \emptyset$ ou $\{Z\}$, $\omega_6 = \emptyset$ ou $\{A\}\}$
    $\Omega_5 \leftarrow \Omega_5 \cup \omega_5$
    $\Omega_6 \leftarrow \Omega_6 \cup \omega_6$
  end for
end for
if $\Omega_5 \neq \emptyset$ or $\Omega_6 \neq \emptyset$ **then**
  `CompleteCenter`(Ω_5, Ω_6)
end if

Algorithme 4 `CompleteCenter`(Ω_5, Ω_6)

$\Omega_2 \leftarrow \emptyset$
$\Omega_9 \leftarrow \emptyset$
for all $Z \in \Omega_5$ **do**
  for all $A \in \texttt{Chain}_5(Z)$ **do**
    $(\omega_2, \omega_9) \leftarrow \texttt{CompleteChain}_5(Z, A)$ $\{\omega_2 = \emptyset$ ou $\{W\}$, $\omega_9 = \emptyset$ ou $\{D\}\}$
    $\Omega_2 \leftarrow \Omega_2 \cup \omega_2$
    $\Omega_9 \leftarrow \Omega_9 \cup \omega_9$
  end for
end for
for all $A \in \Omega_6$ **do**
  for all $Z \in \texttt{Chain}_6(A)$ **do**
    $(\omega_2, \omega_9) \leftarrow \texttt{CompleteChain}_6(A, Z)$ $\{\omega_2 = \emptyset$ ou $\{W\}$, $\omega_9 = \emptyset$ ou $\{D\}\}$
    $\Omega_2 \leftarrow \Omega_2 \cup \omega_2$
    $\Omega_9 \leftarrow \Omega_9 \cup \omega_9$
  end for
end for
if $\Omega_2 \neq \emptyset$ or $\Omega_9 \neq \emptyset$ **then**
  `CompleteExtCh`(Ω_2, Ω_9)
end if

Algorithme 5 $\texttt{CompleteChain}_2(W, R, S, D)$

if $N \geq q$ **then**
  exit with error OUT_OF_BOUND
end if
$\omega_5 \leftarrow \emptyset$
$\omega_6 \leftarrow \emptyset$
$C \leftarrow S \oplus F_9(D)$
if $C \notin F_8$ **then**
  $F_8(C) \xleftarrow{\$} \{0,1\}^n$
end if
$B \leftarrow D \oplus F_8(C)$
if $B \notin F_7$ **then**
  $F_7(B) \xleftarrow{\$} \{0,1\}^n$
end if
$A \leftarrow C \oplus F_7(B)$
if $A \notin F_6$ **then**
  $F_6(A) \xleftarrow{\$} \{0,1\}^n$
  $\omega_6 \leftarrow \{A\}$
end if
$Z \leftarrow B \oplus F_6(A)$
if $Z \notin F_5$ **then**
  $F_5(Z) \xleftarrow{\$} \{0,1\}^n$
  $\omega_5 \leftarrow \{Z\}$
end if
$Y \leftarrow A \oplus F_5(Z)$
$X \leftarrow R \oplus F_2(W)$
if $X \in F_3$ or $Y \in F_4$ **then**
  exit with error UNABLE_TO_ADAPT
end if
$F_2(X) \leftarrow W \oplus Y$
$F_3(Y) \leftarrow X \oplus Z$
$N \leftarrow N + 1$
return (ω_5, ω_6)

Algorithme 6 $\texttt{CompleteChain}_5(Z, A)$

$\omega_2 \leftarrow \emptyset$
$\omega_9 \leftarrow \emptyset$
$B \leftarrow Z \oplus F_6(A)$
if $B \notin F_7$ **then**
  $F_7(B) \xleftarrow{\$} \{0,1\}^n$
end if
$C \leftarrow A \oplus F_7(B)$
if $C \notin F_8$ **then**
  $F_8(C) \xleftarrow{\$} \{0,1\}^n$
end if
$D \leftarrow B \oplus F_8(C)$
if $D \notin F_9$ **then**
  $F_9(D) \xleftarrow{\$} \{0,1\}^n$
  $\omega_9 \leftarrow \{D\}$
end if
$S \leftarrow C \oplus F_9(D)$
if $S \notin F_{10}$ **then**
  $F_{10}(S) \xleftarrow{\$} \{0,1\}^n$
end if
$T \leftarrow D \oplus F_{10}(S)$
$L \| R \leftarrow \boldsymbol{P}^{-1}(S \| T)$
if $R \notin F_1$ **then**
  $F_1(R) \xleftarrow{\$} \{0,1\}^n$
end if
$W \leftarrow L \oplus F_1(R)$
if $W \notin F_2$ **then**
  $F_2(W) \xleftarrow{\$} \{0,1\}^n$
  $\omega_2 \leftarrow \{W\}$
end if
$X \leftarrow R \oplus F_2(W)$
$Y \leftarrow A \oplus F_5(Z)$
if $X \in F_3$ or $Y \in F_4$ **then**
  exit with error UNABLE_TO_ADAPT
end if
$F_2(X) \leftarrow W \oplus Y$
$F_3(Y) \leftarrow X \oplus Z$
return (ω_2, ω_9)

Algorithme 7 $\texttt{CompleteChain}_6(A, Z)$

$\omega_2 \leftarrow \emptyset$
$\omega_9 \leftarrow \emptyset$
$Y \leftarrow A \oplus F_5(Z)$
if $Y \notin F_4$ **then**
  $F_4(Y) \xleftarrow{\$} \{0,1\}^n$
end if
$X \leftarrow Z \oplus F_4(Y)$
if $X \notin F_3$ **then**
  $F_3(X) \xleftarrow{\$} \{0,1\}^n$
end if
$W \leftarrow Y \oplus F_3(X)$
if $W \notin F_2$ **then**
  $F_2(W) \xleftarrow{\$} \{0,1\}^n$
  $\omega_2 \leftarrow \{W\}$
end if
$R \leftarrow X \oplus F_2(W)$
if $R \notin F_1$ **then**
  $F_1(R) \xleftarrow{\$} \{0,1\}^n$
end if
$L \leftarrow X \oplus F_1(R)$
$S \| T \leftarrow \boldsymbol{P}(L \| R)$
if $S \notin F_{10}$ **then**
  $F_{10}(S) \xleftarrow{\$} \{0,1\}^n$
end if
$D \leftarrow T \oplus F_{10}(S)$
if $D \notin F_9$ **then**
  $F_9(D) \xleftarrow{\$} \{0,1\}^n$
  $\omega_9 \leftarrow \{D\}$
end if
$C \leftarrow S \oplus F_9(D)$
$B \leftarrow Z \oplus F_6(A)$
if $B \in F_7$ or $C \in F_8$ **then**
  exit with error UNABLE_TO_ADAPT
end if
$F_7(B) \leftarrow A \oplus C$
$F_8(C) \leftarrow B \oplus D$
return (ω_2, ω_9)

Algorithme 8 $\texttt{CompleteChain}_9(D, S, R, W)$

if $N \geq q$ **then**
  exit with error OUT_OF_BOUND
end if
$\omega_5 \leftarrow \emptyset$
$\omega_6 \leftarrow \emptyset$
$X \leftarrow R \oplus F_2(W)$
if $X \notin F_3$ **then**
  $F_3(X) \xleftarrow{\$} \{0,1\}^n$
end if
$Y \leftarrow W \oplus F_3(X)$
if $Y \notin F_4$ **then**
  $F_4(Y) \xleftarrow{\$} \{0,1\}^n$
end if
$Z \leftarrow X \oplus F_4(Y)$
if $Z \notin F_5$ **then**
  $F_5(Z) \xleftarrow{\$} \{0,1\}^n$
  $\omega_5 \leftarrow \{Z\}$
end if
$A \leftarrow Y \oplus F_5(Z)$
if $A \notin F_6$ **then**
  $F_6(A) \xleftarrow{\$} \{0,1\}^n$
  $\omega_6 \leftarrow \{A\}$
end if
$B \leftarrow Z \oplus F_6(A)$
$C \leftarrow S \oplus F_9(D)$
if $B \in F_7$ or $C \in F_8$ **then**
  exit with error UNABLE_TO_ADAPT
end if
$F_7(B) \leftarrow A \oplus C$
$F_8(C) \leftarrow B \oplus D$
$N \leftarrow N + 1$
return (ω_5, ω_6)

Résumé

Nous abordons deux aspects complémentaires de la sécurité prouvée en cryptographie.

Dans une première partie, nous étudions la relation existant entre les deux modèles idéalisés les plus utilisés pour les preuves de sécurité : le modèle de l'oracle aléatoire et le modèle du chiffrement par blocs idéal. Nous montrons que ces deux modèles sont en fait équivalents : l'existence d'un cryptosystème sûr dans l'un des modèles implique l'existence d'un cryptosystème réalisant la même fonctionnalité et sûr dans l'autre modèle. En particulier, nous montrons dans le cadre de la théorie de l'indifférentiabilité que si un cryptosystème utilisant un chiffrement par blocs idéal est sûr, alors le cryptosystème reste sûr en remplaçant le chiffrement par blocs par la construction de Luby-Rackoff à 6 tours où les fonctions internes sont des oracles aléatoires publiquement accessibles.

Dans une seconde partie, nous étudions les protocoles cryptographiques fondés sur le problème LPN (*Learning Parity with Noise*). La proposition du schéma d'authentification HB^+ par Juels et Weis à CRYPTO 2005 a suscité un très grand intérêt et de nombreuses variantes cherchant à renforcer la sécurité de ce protocole contre les attaques *man-in-the-middle* ont été proposées par la suite. Nous présentons des cryptanalyses de trois de ces variantes (HB^{++}, HB^*, et HB-MP), puis proposons les protocoles $\mathrm{HB}^\#$ et RANDOM-$\mathrm{HB}^\#$. La sécurité de ce dernier contre une classe restreinte d'attaques *man-in-the-middle* peut être prouvée sous l'hypothèse de la difficulté du problème LPN. Nous proposons également un schéma de chiffrement probabiliste symétrique dont la sécurité contre les attaques à clairs choisis peut être réduite à la difficulté du problème LPN.

Mots-clés : sécurité prouvée, oracle aléatoire, chiffrement par blocs idéal, indifférentiabilité, problème LPN, protocole d'authentification, schéma de chiffrement symétrique.

Abstract

We address two complementary aspects of provable security in cryptography.

Firstly, we study the relation between two idealised models widely used in security proofs, the random oracle model and the ideal block cipher model. We prove that these two models are in fact equivalent: the existence of a cryptosystem secure in one of the models implies the existence of a cryptosystem realising the same functionality and secure in the other model. In particular, we prove in the indifferentiability framework that if a cryptosystem using an ideal block cipher is secure, then this cryptosystem remains secure when the block cipher is replaced by the Luby-Rackoff construction with 6 rounds where the inner functions are publicly accessible random oracles.

Then, we study cryptographic protocols based on the LPN problem (*Learning Parity with Noise*). The authentication protocol HB^+ proposed by Juels and Weis at CRYPTO 2005 aroused much interest and several variants seeking to reinforce the security of this protocol against man-in-the-middle attacks were subsequently proposed. We present a cryptanalysis of three of these variants (HB^{++}, HB^*, and HB-MP), and then we propose the protocols $\mathrm{HB}^\#$ and RANDOM-$\mathrm{HB}^\#$. The security of RANDOM-$\mathrm{HB}^\#$ against a limited class of man-in-the-middle attacks can be proven under the assumption of the difficulty of the LPN problem. We also propose a probabilistic symmetric encryption scheme whose security against chosen plaintext attacks can be reduced to the difficulty of the LPN problem.

Keywords: provable security, random oracle, ideal block cipher, indifferentiability, LPN problem, authentication protocol, symmetric encryption scheme.