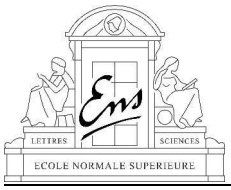
Mécanismes cryptographiques pour la génération de clefs et l'authentification.

Sébastien Zimmer



École normale supérieure
Département d'informatique

École doctorale de l'École polytechnique
Direction du 3e cycle

Mécanismes cryptographiques pour la génération de clefs et l'authentification

THÈSE

présentée et soutenue publiquement le 22 septembre 2008

pour l'obtention du

Doctorat de l'École polytechnique

(spécialité informatique)

par

Sébastien Zimmer

Composition du jury

Rapporteurs :	Reynald Lercier (*CELAR et Université de Rennes 1*) Bart Preneel (*Katholieke Universiteit Leuven, Belgique*)
Directeur de thèse :	David Pointcheval (*École normale supérieure et CNRS*)
Examinateurs :	Pierre-Alain Fouque (*École normale supérieure*) Adi Shamir (*Weizmann Institute, Israël*) Jacques Stern (*École normale supérieure et Ingenico*) Jean-Marc Steyaert (*École polytechnique*) Serge Vaudenay (*École polytechnique de Lausanne, Suisse*)
Invité :	Guillaume Poupard (*Ministère de la défense*)

Laboratoire d'Informatique de l'École normale supérieure, 45 rue d'Ulm, 75230 Paris Cedex 05

Remerciements

De nombreuses personnes ont concouru, de manière directe ou indirecte, à l'élaboration de cette thèse. Voici quelques temps déjà que je souhaite leur exprimer ma reconnaissance et je profite de l'occasion que m'offre ce mémoire pour leur formuler ces quelques sincères remerciements.

Ma plume s'adresse d'abord conjointement à David Pointcheval, mon directeur de thèse, et à Pierre-Alain Fouque, qui a guidé mon travail au quotidien. Travailler avec vous deux a été à la fois un plaisir et une chance : un plaisir car vous m'avez témoigné votre amitié, une chance car j'ai bénéficié de vos deux savoir-faire scientifiques complémentaires. J'ai trouvé en vous plus que des collègues et j'ai beaucoup appris à vos côtés.

Je souhaite aussi exprimer ma gratitude à tous les membres de l'équipe crypto de l'École normale supérieure, passés et présents, permanents, post-doc, thésards et stagiaires. Vous avez tous contribué à la bonne rédaction de ce mémoire, par vos apports scientifiques et en me permettant de travailler dans la bonne humeur. J'adresse une pensée particulière à Vivien Dubois, Jean-Baptiste Note et Nicolas Gama avec qui j'ai partagé de nombreux repas, passé des heures entières à discuter et même, des fois, travaillé.

Je veux aussi remercier tous mes co-auteurs, sans lesquels aucune de mes contributions scientifiques n'aurait vu le jour, en particulier Charles Bouillaguet pour m'avoir invité à participer aux travaux qu'il avait entamés sur les attaques en seconde préimage.

Je veux exprimer des remerciements chaleureux à Reynald Lercier pour avoir rempli le rôle de parrain au sein de la DGA et m'avoir prodigué ses précieux conseils lorsque j'étais perdu. Je le remercie également, ainsi que Bart Preneel, pour avoir assumé la lourde tâche de rapporteurs. Vous avez eu le courage de vous perdre dans le dédale des preuves pour vérifier leur exactitude et je vous en suis reconnaissant.

Je veux également adresser ma gratitude à Guillaume Poupard, Adi Shamir, Jacques Stern, Jean-Marc Steyaert et Serge Vaudenay qui me font l'amitié de participer à mon jury.

Je remercie Joëlle Isnard et Valérie Mongiat pour leur soutien logistique efficace.

J'aimerais aussi remercier le RER C, qui est un lieu propice à la réflexion de Pierre-Alain et qui a servi de pépinière à de nombreuses idées présentes ici. Combien de fois l'ai-je entendu arriver le matin en disant : « J'ai pensé ce matin dans le RER… ».

Merci à Gwenaëlle et Nelly pour m'avoir prêté leur compagnon respectif lors des veillées précédant les soumissions.

Enfin, je me tourne vers Alice, la dernière de ces quelques lignes mais la première dans mon cœur. Merci pour ton soutien inconditionnel, même lors des conférences à l'étranger, et merci pour ta relecture du manuscrit alors que tu n'y comprenais goutte. Ce mémoire te doit plus que je ne saurais le dire.

Que tous ceux que j'aurais pu involontairement oublier me pardonnent et trouvent ici exprimée ma reconnaissance.

Merci à ceux qui auront le courage de tourner ces pages, je vous souhaite une bonne lecture.

À Alice (et tant pis pour Bob),

À mon grand-père.

Table des matières

Table des figures

Notations

$\#M$	Cardinal de l'ensemble M
$\wedge$ et $\vee$	Respectivement 'et' et 'ou' logique
$\oplus$	Addition bit à bit modulo 2, appelé "xor"
$x\|\|x'$ ou $x.x'$	Concaténation de x et x'
$x \sqsubset x'$	Signifie x est un prefixe de x'
$\|x\|$	Taille de x en bits
$\|x\|_8$	Taille de x en octets
$\|x\|_b$	Nombre de blocs de b bits constituant x
$x_{pad} = x\|\|\text{pad}(\|x\|)$	Chaîne de bits x à laquelle on ajoute un *padding* fonction de sa taille
$\mathbf{SD}(X_1, X_2)$	Distance statistique entre les variables aléatoires X_1 et X_2
$\text{Col}(X)$	Probabilité de collision de la variable aléatoire X
$\mathbf{H}_2(X)$	Entropie de collision de la variable aléatoire X
$\mathbf{H}_\infty(X)$	Min entropie de la variable aléatoire X
$X \leftarrow \mathcal{D}$	La variable aléatoire X est choisie selon la distribution $\mathcal{D}$
$X \xleftarrow{\$} \mathcal{X}$	La variable aléatoire X est choisie selon la distribution uniforme sur $\mathcal{X}$
$\mathcal{U}_\mathcal{X}$	Distribution uniforme sur l'ensemble $\mathcal{X}$
$U_\mathcal{X}$	Variable uniformément distribuée sur l'ensemble $\mathcal{X}$
$\Pr[\mathcal{D}\colon A]$	Probabilité de l'événement A mesurée selon la distribution $\mathcal{D}$
$\Pr[A \mid B]$	Probabilité de l'événement A sachant l'événement B
$\mathbb{E}(X)$	Espérance de la variable aléatoire X
$\mathsf{lsb}_t(x)$	Les t bits de poids faible de x
$\mathsf{msb}_t(x)$	Les t bits de poids fort de x
$GF(2^n)$	Le corps à 2^n éléments, $n \in \mathbb{N}$
$\mathcal{A} \Rightarrow x$	L'adversaire $\mathcal{A}$ envoie le message x
$\mathcal{A} \Leftarrow x$	L'adversaire $\mathcal{A}$ reçoit le message x
$\mathcal{A}^O$	L'adversaire $\mathcal{A}$ a accès à l'oracle O
$\mathsf{adv}_P^{\mathtt{atk}}(\mathcal{A})$	Avantage de l'adversaire $\mathcal{A}$ dans le jeu atk contre le protocole ou la primitive P
$\mathsf{Succ}_P^{atk}(\mathcal{A})$	Probabilité que l'adversaire $\mathcal{A}$ gagne le jeu atk contre le protocole ou la primitive P
$Fact_\mathbf{z}(\ell)$	Nombre de facteurs de taille ℓ de la suite $\mathbf{z}$
$\mathbb{Z}_p$	Corps fini à p éléments

Première partie

Introduction Générale

1

Introduction

Sommaire

1.1 Brève histoire de la cryptologie

1.1.1 De l'antiquité aux premiers ordinateurs

Les origines de la cryptologie remontent à l'antiquité : le plus vieux document chiffré retrouvé est une tablette d'argile datant du XVI^{e} siècle avant J.-C. Les procédés de chiffrement par substitution sont apparus à cette période et ont été très largement utilisés depuis lors. Au fil des siècles, les techniques cryptographiques se perfectionnent lentement et se complexifient. Jusqu'au XIX^{e} siècle, les motivations sont essentiellement diplomatiques et militaires, l'usage de la cryptographie est pour beaucoup l'apanage des puissants qui font appel à des spécialistes, des dynasties de cryptographes apparaissent (les Rossignol en France, les Da Porta en Italie). Durant toute cette période, la cryptographie se résume essentiellement à un ensemble de savoir-faire et est considérée comme un art et non comme une science.

L'essor du télégraphe dans les années 1840 accroît le besoin de dissimuler le sens des messages et, de ce fait, favorise la généralisation de la cryptographie et la réflexion sur les méthodes cryptographiques. En 1883, Auguste Kerckhoffs publie le livre *La cryptographie militaire* dans lequel il énonce plusieurs propriétés que doit avoir un bon système de chiffrement. Parmi celles-ci, nous citons ici les trois premières qui font preuve d'une étonnante modernité et sont toujours communément admises :

– Le système doit être matériellement, sinon mathématiquement, indéchiffrable.
– Il faut qu'il n'exige pas le secret et qu'il puisse sans inconvénient tomber entre les mains de l'ennemi.

– La clef doit pouvoir (...) être changée ou modifiée au gré des correspondants.

La première propriété signifie que même s'il y a une manière théorique de casser un système cryptographique, ce dernier peut être considéré comme sûr s'il n'existe pas de méthode pour le casser en un temps raisonnable avec les outils dont nous disposons. En d'autres termes, si la meilleure stratégie pour mettre à mal sa sécurité requiert de faire tourner le meilleur ordinateur actuellement disponible pendant 1000 ans, alors on considère que ce système cryptographique peut être utilisé. Actuellement, presque tous les algorithmes utilisés reposent sur des problèmes que l'on pense *calculatoirement difficiles*, c'est-à-dire, qui ont une solution, mais que l'on ne peut pas résoudre rapidement. On arrive à prouver que pour pouvoir porter atteinte à la résistance de ces algorithmes il faut préalablement réussir à casser un de ces problèmes calculatoirement difficiles, ce qui est impossible en temps raisonnable.

Les deuxième et troisième principes mettent en exergue la notion centrale de clef. L'architecture d'un système cryptographique doit pouvoir être connue de tous, même d'un éventuel attaquant, seul le secret de la clef doit être nécessaire pour assurer la sécurité. Cette clef doit être flexible et pouvoir être changée facilement, de telle sorte que l'on en utilise une différente à chaque échange. À la difficulté de concevoir un algorithme résistant aux attaques extérieures, s'ajoutent donc les problèmes de générer une clef sûre puis de modifier régulièrement cette clef. Cette question est d'autant plus importante que toute la sécurité de l'algorithme repose sur la confidentialité de cette clef et la moindre fuite d'information peut compromettre tout le système. Il faut donc être très prudent lors de la mise en accord de clefs. C'est d'ailleurs la difficulté de mettre à jour certaines clefs dans les systèmes cryptographiques allemands qui facilite le travail des cryptanalystes anglais lors de la seconde guerre mondiale.

En effet, pendant cette période les allemands utilisent la plus connue des machines de chiffrement mécaniques : Enigma. Ces machines de chiffrement ont fait leur apparition après la première guerre mondiale et sont intensément utilisées au cours de la seconde. Enigma possède un jeu de rotors dont l'agencement est mis à jour quotidiennement, ce qui revient à changer en partie la clef. Cependant, pour renouveller entièrement la clef, il faudrait que toutes les troupes allemandes modifient régulièrement le jeu de rotors utilisé, et ce de façon synchronisée, ce qui est difficile à effectuer. C'est pourquoi ceci n'est fait que très rarement et l'activité de cryptanalyse des forces alliées s'en trouve allégée. C'est justement pour casser certains algorithmes de chiffrement allemands que les anglais construisent le *Colossus*, considéré comme le premier ordinateur à avoir vu le jour. La cryptographie fait alors connaissance avec l'informatique.

1.1.2 De la naissance d'une science

Après la guerre, entre 1948 et 1950, Shannon fonde la théorie de l'information et initie avec elle, la cryptologie moderne. C'est un tournant important dans l'histoire de la cryptologie, mais il faudra attendre la fin des années 1970 pour que cette science nouvelle prenne vraiment son essor. Ces années sont marquées par la normalisation de l'algorithme de chiffrement symétrique DES (1977), mais surtout par la naissance de la cryptographie asymétrique avec l'article fondateur de Diffie et Hellman [DH76] (1976) puis la création de l'algorithme RSA [RSA78] (1977). Les premières conférences académiques apparaissent au début des années 1980.

La naissance de la cryptographie asymétrique révolutionne la cryptologie et notamment l'échange de clefs. Jusqu'à présent, ce dernier était effectué par des méthodes classiques : une des parties écrivait une liste de clefs à l'avance et la transmettait à l'autre partie, parfois grâce à une tierce personne de confiance. C'est cette méthode qui était employée par les allemands durant la seconde guerre mondiale : ils utilisaient un calendrier secret dans lequel étaient notés les réglages initiaux à utiliser pour chaque jour. Dorénavant, il est possible d'effectuer cette mise en accord

de clefs à distance et au dernier moment par des méthodes cryptographiques. Cependant, un nouveau problème se pose : lors d'un échange de clefs, puisque je suis à distance, il me faut pouvoir vérifier l'identité de l'entité avec laquelle je communique car rien n'empêche *a priori* un adversaire de se faire passer pour la personne avec laquelle je veux discuter. Avant, cette identité ne pouvait être mise en doute : les seules personnes capables de communiquer étaient celles qui possédaient déjà la clef. Maintenant il faut effectuer une *authentification* en même temps que l'échange de clefs. Ces deux notions sont donc liées et ont été très étudiées depuis lors. De nombreuses solutions ont été élaborées pour assurer l'authentification lors d'un échange de clefs.

Grâce aux nouvelles fonctionnalités apportées par la cryptographie asymétrique et au développement de l'informatique, la cryptographie est maintenant très répandue et on l'utilise quotidiennement par le biais d'objets courants (carte bleue, ordinateurs, cartes vitales,...).

1.2 Établissement d'un canal sécurisé

1.2.1 Les trois principes fondamentaux

De nos jours, les outils cryptographiques sont conçus pour remplir de plus en plus de fonctionnalités différentes. Parmi ces nombreuses fonctionnalités, il y en a trois qui sont particulièrement importantes : les protocoles cryptographiques doivent assurer la confidentialité, l'intégrité et l'authentification. Même si nous sommes particulièrement intéressés dans ce mémoire par l'authentification nous allons voir à la sous-section 1.2.3 que notre étude ne peut se faire sans examiner également les deux autres notions.

Éthymologiquement, la cryptologie est « la science du secret », définition qui correspond avec son objectif originel qui était de préserver le secret des messages. C'est la propriété à laquelle tout le monde pense quand on évoque la cryptologie : garantir la confidentialité, ce qui signifie que l'on garde le contenu d'un message secret vis à vis de tout le monde, sauf de ceux qui sont autorisés à le voir.

Cependant, cette fonctionnalité ne suffit pas à garantir la sécurité des communications. Un adversaire peut intercepter un message puis, même s'il n'en connait pas le contenu, modifier ce message de telle sorte que sa signification soit différente. Il ne lui reste plus qu'à faire suivre le message altéré vers son destinataire légitime. Ainsi, même s'il ne connaît pas le contenu du message initial et donc ne sait ni ce qu'il a changé exactement, ni le sens du message obtenu, il peut au moins réussir à désorganiser le camp qu'il attaque et en tirer profit. C'est pourquoi on exige que les protocoles cryptographiques permettent de repérer les messages qui ont été altérés de ceux qui sont restés inchangés : on demande que les protocoles préservent l'intégrité.

Enfin, un adversaire peut se mettre entre les deux parties communiquant. Comme il est habituel en cryptologie, appelons ces deux entités Alice et Bob. Placé entre Alice et Bob, l'adversaire peut intercepter les conversations et essayer de se faire passer pour Bob auprès d'Alice et/ou pour Alice auprès de Bob, et ainsi obtenir des informations. Pour éviter cette attaque, dite attaque par le milieu, il faut qu'Alice puisse vérifier que les messages viennent bien de Bob et réciproquement. En d'autres termes, le récipiendaire d'un message doit pouvoir s'assurer de l'identité de l'entité qui a généré le message. On dit que le message est authentifié et que le protocole assure l'authentification.

En résumé, un des objectifs principaux de la cryptographie est de rendre possible l'établissement de ce que l'on appelle des canaux authentifiés, confidentiels et intègres et une grande attention est portée sur la conception de protocoles qui créent de tels canaux. Dans la suite de ce mémoire, nous parlerons de canal sécurisé pour signifier qu'il est intègre, confidentiel et authentifié.

1.2.2 Modularité des protocoles

Avant de décrire plus précisément le fonctionnement des protocoles qui créent des canaux sécurisés, il est intéressant de remarquer que comme tous les outils cryptographiques, ces protocoles ont une structure modulaire. Puisque cette organisation a beaucoup d'influence sur la manière dont nous menons nos études dans ce mémoire, nous faisons une petite disgression sur celle-ci.

Dans la suite de ce mémoire, nous sommes amenés à étudier des primitives et des protocoles cryptographiques. Les primitives sont des fonctions (au sens mathématiques du terme) qui remplissent une ou plusieurs fonctionnalités précises. Les protocoles sont définis par les entités composant le protocole, par la forme des échanges ayant lieu entre les participants et par les fonctionnalités remplies par le protocole.

Que ce soit pour construire une primitive ou un protocole cryptographique, comme on le verra tout au long de ce mémoire, on favorise toujours des constructions modulaires. Le principe de ces constructions est de commencer par créer des briques élémentaires, remplissant des fonctionnalités aussi simples et précises que possible, puis de trouver des agencements de ces briques élémentaires permettant de remplir des fonctions plus complexes. Les agencements créant une primitive complexe à partir de primitives simples est appelé *mode opératoire*.

Cette méthode de construction modulaire présente deux avantages majeurs. D'une part, cela facilite l'analyse de sécurité, puisqu'elle permet d'étudier chaque bloc du protocole indépendamment des autres. Une fois toutes ces analyses mises bout à bout, cela permet par conséquent de faire reposer toute la sécurité du protocole sur la sécurité des briques élémentaires. Il suffit donc de concentrer son attention sur celles-ci afin d'assurer la résistance de tout l'édifice et donc la confiance que l'on a en lui.

D'autre part, il existe souvent plusieurs outils cryptographiques remplissant la même fonctionnalité et on peut choisir celui que l'on souhaite comme brique élémentaire. Comme ces outils ont des niveaux de sécurité différents et des facteurs d'efficacité différents, cela permet de choisir un compromis entre sécurité et efficacité et surtout de remplacer un outil cryptanalysé par un autre plus solide.

1.2.3 Constitution d'un protocole établissant un canal sécurisé

Les protocoles qui établissent des canaux sécurisés sont une illustration de cette organisation modulaire des outils cryptographiques. En effet, la création d'un canal sécurisé est divisée en deux phases :

- l'échange de clefs authentifié qui est rendu possible grâce aux outils de cryptographie asymétrique puisque ceux-ci ne nécessitent pas de secret partagé préalable,
- la création du canal en lui-même. Grâce aux clefs générées lors de la première étape, on peut utiliser des primitives de cryptographie symétrique, calculatoirement plus efficaces que leurs équivalents asymétriques, pour mettre en place le canal authentifié, confidentiel et intègre.

On ne décrit et n'examine ici que les protocoles hybrides, faisant appel à des primitives symétriques et asymétriques, car ce sont les plus efficaces et les plus utilisés en pratique.

Bien évidemment, les clefs échangées doivent rester parfaitement confidentielles, car si l'adversaire parvenait à récupérer des informations sur la valeur de ces clefs, il pourrait les utiliser pour mettre à mal la sécurité du canal. Mais cette confidentialité n'est pas la seule condition *sine qua non* pour assurer cette sécurité. Comme nous l'avons expliqué à la sous-section 1.2.1, il faut que l'échange de clefs soit authentifié pour éviter les attaques par le milieu. Dans le cas

contraire, un adversaire peut s'interposer entre Alice et Bob, générer une clef avec Alice en se faisant passer pour Bob et générer une clef avec Bob en se faisant passer pour Alice ; dans ce cas de figure tout message à destination de Bob ou Alice transite par l'adversaire qui le déchiffre, le lit et éventuellement le modifie, puis le rechiffre et le fait suivre à son destinataire légitime. L'authentification est donc absolument nécessaire pour être sûr de parler à la bonne personne, et, par conséquent, pour assurer également la confidentialité et l'intégrité des messages qui sont échangés.

Réciproquement, le caractère authentifié du canal dans son ensemble est une conséquence non seulement de l'authentification des clefs, mais aussi du caractère intègre du canal. Puisque les clefs sont authentifiées, on sait exactement qui possède ces clefs et comme le canal est intègre, seule une personne possédant ces clefs peut générer un message valide. En conséquence de quoi, lorsqu'on obtient un message valide, on sait qui l'a généré, le message est donc authentifié. L'authentification est donc une conséquence de l'intégrité et de la confidentialité.

En résumé, pour établir un canal sécurisé, il faut et il suffit donc que l'échange de clefs soit authentifié et génère des clefs parfaitement confidentielles et que le canal établi assure l'intégrité. Comme on le voit, les trois notions d'authentification, d'intégrité et de confidentialité sont indissociables et il nous faut, pour assurer l'authentification tout au long de l'établissement du canal, assurer les trois notions en même temps. Pour analyser les mécanismes cryptographiques permettant l'authentification, il faut donc aussi s'intéresser dans une certaine mesure aux méthodes pour assurer l'intégrité et la confidentialité.

Pour cette raison, nous examinerons dans une première partie de ce mémoire, la mise en place d'un canal sécurisé en présentant d'abord un nouveau protocole d'échange de clefs authentifié, protocole s'appuyant sur différents modes d'authentification, puis nous analyserons la sécurité d'un mode opératoire symétrique de chiffrement authentifié, appelé CCM, qui garantit confidentialité et intégrité.

1.3 Dérivation de clefs

Nous venons de voir que l'échange de clefs authentifié est une étape primordiale dans le processus aboutissant à la mise en place d'un canal sécurisé. Il permet de générer un secret partagé entre les entités participant au protocole. Obtenu grâce à des techniques asymétriques, c'est souvent un élément d'un groupe (c'est le cas par exemple pour le résultat d'un échange Diffie-Hellman) et il possède beaucoup d'entropie (c'est-à-dire qu'un adversaire ne parvient à deviner sa valeur qu'avec très faible probabilité), mais il n'est que rarement une chaîne de bits uniformément distribuée. Cependant, les primitives symétriques utilisées lors de la deuxième phase d'établissement du canal sécurisé requièrent absolument comme clefs des chaînes de n bits qui soient uniformément distribuées dans $\{0,1\}^n$, pour un certain n. On ne peut donc utiliser l'élément secret partagé directement comme clef dans la seconde phase, il faut lui faire subir un traitement que l'on appelle la *dérivation de clefs*. Cette étape charnière entre l'échange de clefs et la création du canal est souvent intégrée en pratique à l'échange de clefs.

Pour assurer l'authentification tout au long du processus, il faut garantir une confidentialité parfaite de la clef générée et pour cela il faut notamment être attentif à la manière dont on effectue la dérivation de clefs afin de ne pas libérer d'information. En pratique, la méthode la plus répandue consiste à hacher l'élément secret avec une fonction de hachage. Si le hachage ne permet pas de générer suffisamment de bits, on ajoute un compteur et on hache plusieurs fois l'élément secret, pour différentes valeurs du compteur et on concatène les résultats. Cette méthode présente l'avantage d'être très simple à mettre en œuvre car les fonctions de hachage

sont présentes dans presque toutes les bibliothèques de cryptographie. Cependant, cette solution n'est *a priori* prouvée sûre que dans le modèle de l'oracle aléatoire, ce qui est peu satisfaisant car c'est un modèle fort.

C'est d'autant moins satisfaisant que l'on dispose d'un schéma théorique et d'outils permettant d'effectuer une dérivation de clefs prouvée sûre sans avoir recours à un oracle aléatoire. La solution consiste à suivre le modèle dit d'*extraction-puis-expansion* (*extract-then-expand* en anglais). Ce modèle se décompose en deux étapes qui sont, comme son nom le laisse présager, une étape d'*extraction* et une étape d'*expansion*.

Lors de l'étape d'*extraction*, l'objectif est de transformer le long secret partagé ayant beaucoup d'entropie en une chaîne de bits plus courte mais uniformément distribuée. L'entropie se trouve diluée au sein de l'élément secret et, quitte à perdre de l'entropie pendant la transformation, on va prendre une partie de l'entropie diluée pour la concentrer dans une petite chaîne de bits. Pour cela, on utilise ce que l'on appelle tout simplement un *extracteur d'entropie*, outil qui a été introduit formellement par Nisan et Zuckerman [NZ96] en 1996 et qui réalise exactement ce dont on a besoin. Même si cet outil n'est pas toujours très connu, il existe de nombreux extracteurs d'entropie dans la littérature et certains sont efficaces, simples à construire et pourraient convenir à une utilisation pratique.

L'étape d'expansion de la clef consiste à transformer la chaîne de bits courte et uniformément distribuée, obtenue à l'étape précédente, en une chaîne de bits longue qui n'est pas vraiment uniformément distribuée mais qui semble l'être aux yeux d'un adversaire calculatoirement limité. Pour cela, on utilise un générateur pseudo-aléatoire ou PRG (pour *pseudo-randomness generator* en anglais). Les générateurs pseudo-aléatoires sont connus depuis longtemps, on sait en construire qui sont tout à la fois efficaces et sûrs.

Comme on le voit, il est possible de mettre en œuvre des outils de dérivation de clefs efficaces et sûr en dehors du modèle de l'oracle aléatoire. Hélas, ces outils ne sont pas utilisés en pratique et ceci pour deux raisons majeures. La première est une raison historique : beaucoup de protocoles utilisés de nos jours sont les descendants de protocoles créés lors des débuts d'Internet. À cette époque, les extracteurs d'entropie n'existaient pas encore formellement, la dérivation de clefs n'avait pas été étudiée de façon théorique, le protocole a donc été conçu avec les connaissances de l'époque. Par la suite, très souvent on a voulu que les versions successives du protocoles soient compatibles entre elles et cette étape de dérivation de clefs n'a pas été améliorée. La seconde explication est la volonté de simplifier la programmation des protocoles : plus le protocole utilise de primitives cryptographiques, plus il faut programmer de primitives. Les fonctions de hachage étant un outil très répandu en cryptographie, elles sont déjà présentes dans les bibliothèques de programme, il n'est donc pas nécessaire d'ajouter une nouvelle primitive.

La dérivation de clefs décrite ci-avant à partir de fonctions de hachage peut facilement s'interpréter en termes de PRG si on considère que la fonction de hachage est une famille de fonctions pseudo-aléatoires : utiliser une fonction pseudo-aléatoire en mode compteur permet de construire un générateur pseudo-aléatoire. C'est donc avant toute chose l'étape d'extraction de clefs qui est "négligée" en pratique et sur laquelle il faut concentrer son attention. C'est ce qui a déjà été remarqué par Krawczyk [Kra] et c'est pour cette raison que celui-ci promeut une nouvelle norme pour la dérivation de clefs, norme qui respecte le modèle extraction-puis-expansion. Pour la même raison, nous nous intéressons exclusivement dans ce mémoire à l'étape d'extraction de clefs. Afin de favoriser l'intégration de cette dernière dans les protocoles utilisés en pratique, nous analysons des outils simples à mettre en œuvre. Nous nous penchons d'abord sur un extracteur d'entropie très simple qui peut être utilisé avec les éléments Diffie-Hellman, puis nous analysons sous quelles conditions nous pouvons utiliser une fonction de hachage pour cette étape d'extraction de clefs.

1.4 Modes opératoires de fonctions de hachage

Les fonctions de hachage sont des primitives très employées en cryptographie. Les plus connues sont les fonctions de la famille de MD4 : MD5, SHA–1, SHA–2. Leur objectif originel est de donner le condensé d'un message, mais elles servent aussi pour de nombreuses autres applications. Pour cette raison, elles sont très présentes tout au long de ce mémoire. Elles remplacent notamment en pratique les oracles aléatoires qui sont des primitives « idéales », c'est pourquoi on les retrouve très régulièrement lorsque l'on examine les fonctions de dérivations de clefs. Nous les considèrons sous cet angle au chapitre 1. Elles sont parfois également considérées comme des fonctions pseudo-aléatoires [BCK96b, Bel06] et c'est d'ailleurs ce que nous faisons aux chapitres 2 et 4 de ce mémoire.

Leur importance pratique, notamment dans le domaine de la dérivation de clefs et de l'authentification, accentue la question de leur sécurité. Or précisément, ces fonctions ont subi plusieurs attaques récentes, attaques qui ont visé les primitives elles-mêmes [WLF+05, WYY05a, WY05, WYY05b, BCJ+05, Kli06, JP07], mais aussi les modes opératoires [Jou04, KS05, KK06] utilisés pour les construire. Certaines de ces attaques sont particulièrement efficaces : MD4 et MD5 ne sont plus résistantes aux collisions, les attaques contre SHA–1 sont encore théoriques mais pourraient être améliorées et devenir pratiques dans un avenir proche. De ce fait, la confiance que l'on accordait aux fonctions actuelles a diminué. L'institut de normes américain, le *National Institute of Standards, and Technology* ou NIST, a lancé un appel pour inciter à la création de nouvelles fonctions de hachage. Ces attaques et cet appel ont également eu pour effet de remettre en cause les modes opératoires utilisés et de favoriser la recherche de nouveaux modes. Puisque ces nouveaux modes seront notamment utilisés pour l'extraction de clefs, il est important d'examiner leur sécurité et de s'assurer qu'ils atteignent les niveaux de sécurité escomptés.

Pour cette raison, nous analysons, dans la dernière partie de ce mémoire, plusieurs nouveaux modes opératoires de fonctions de hachage. Le premier mode observé est une construction à partir d'algorithme de chiffrement par bloc dont les clefs sont fixées une fois pour toute avant le hachage et n'évoluent plus pendant le hachage. Cette construction atteint des niveaux de sécurité théoriques moindres que les niveaux habituels, mais repose sur une hypothèse moins forte que les constructions classiques. Ensuite, nous analysons des modes qui cherchent à atteindre une résistance aux secondes préimages optimale et montrons que certains n'ont pas la sécurité espérée en exhibant une attaque, alors qu'au contraire d'autres ont cette sécurité en donnant une preuve.

2

Introduction aux preuves de sécurité

Sommaire

La méthode d'analyse de sécurité utilisée dans ce mémoire est la même quelle que soit la fonction étudiée. Elle nécessite dans un premier temps une modélisation des outils cryptographiques qui sont utilisés et de la propriété que l'on veut garantir, modélisation effectuée grâce à un jeu faisant intervenir un attaquant. Ce jeu définit de façon précise quels sont les objectifs de cet attaquant, quels sont ses pouvoirs, et quels sont ses moyens. Tout ceci définit le « modèle de sécurité ». C'est ce jeu que nous allons analyser, l'objectif étant d'établir une borne la plus fine possible sur la probabilité de succès de l'attaquant, ce qui donne en définitive une borne supérieure sur la probabilité de casser la propriété qui nous intéresse.

On complète parfois cette analyse de sécurité par la présentation d'une attaque du protocole. L'objectif est alors d'examiner quelle est la meilleure stratégie connue que peut adopter l'adversaire pour casser la propriété. Au contraire de la preuve, cette attaque a pour vocation de donner une borne inférieure sur la probabilité de succès du meilleur attaquant, puisque la meilleure attaque est au moins aussi bonne que celle qui est présentée. Lorsqu'on est capable

d'exhiber une attaque, on connaît alors une borne inférieure qui, combinée à la borne supérieure issue de la preuve, permet une estimation plus fine de la sécurité.

2.1 Machine de Turing, réduction et adversaires

Nous donnons maintenant le cadre formel des preuves et notions de sécurité présentées dans ce mémoire.

2.1.1 Machine de Turing

Le concept des machines de Turing a été introduit en 1936 par Alan Turing pour donner un model abstrait de fonctionnement des ordinateurs et donner un sens mathématique au concept d'algorithme. Une machine de Turing est un automate fini auquel on donne en plus accès à une mémoire, sur laquelle il peut lire et écrire des informations. Cette mémoire est modélisée par un *ruban*, c'est-à-dire une suite de $\mathbb{N}$ de valeurs prises dans un alphabet de taille finie. Ce ruban a donc un début, mais est infini « à droite ». La machine de Turing possède une tête de lecture qui se déplace le long du ruban pour lire et écrire sur ce dernier. Dans la suite on appellera ce ruban, le *ruban de travail*.

Plus formellement, voici la définition d'une machine de Turing.

Définition 2.1 (Machine de Turing)**.** *On appelle* machine de Turing M *le 8-uplet que l'on note* $(\Sigma, B, Q, \sigma, \delta, \Delta, q_0, F)$ *et qui est tel que :*

- Σ *est un ensemble fini de symboles contenant un symbole particulier noté* B *dit* blanc*,*
- Q *un ensemble non vide d'états,*
- $\sigma \colon Q \times \Sigma \to \Sigma$ *est une fonction d'impression,*
- $\delta \colon Q \times \Sigma \to \Sigma$ *est une fonction de transition,*
- $\Delta \colon Q \times \Sigma \to \{G, D\}$ *est une fonction de déplacement,*
- $q_0 \in Q$ *un état dit de départ,*
- F *est un sous-ensemble d'états de* Q *appelé ensemble des états finaux.*

Cette machine de Turing M fonctionne de la manière suivante. Initialement, le ruban possède un nombre fini de symboles de Σ différents du blanc B, M est dans l'état q_0 et sa tête de lecture est placée sur le premier symbole du ruban. À chaque étape, la machine lit le symbole α sous la tête de lecture et en fonction de α et de son état q,

- elle écrit sur le ruban le symbole $\sigma(q, \alpha)$,
- elle passe dans l'état $\delta(q, \alpha)$,
- elle se déplace le long du ruban dans la direction $\Delta(q, \alpha)$.

Elle recommence tant qu'elle n'aboutit pas à un des états finaux et tant que la tête du ruban ne sort pas du ruban par la gauche. Elle pourrait ne jamais s'arrêter, mais dans ce mémoire, nous ne considèrerons que des machines de Turing qui s'arrêtent en arrivant dans un état final. La *complexité* de la machine de Turing est définie par le nombre d'étapes effectuées par celle-ci avant d'arriver dans un état final.

On peut étendre la notion de machine de Turing aux machines de Turing probabilistes. Cette dernière est une machine de Turing classique M à laquelle on donne également accès à un *ruban d'aléa* ω, c'est-à-dire à une suite de $\mathbb{N}$ de charactères choisis chacun de façon uniforme dans Σ. Ce ruban d'aléa est en lecture seule, M ne peut écrire dessus : à chaque itération M y lit un caractère puis se déplace vers la droite. Enfin, aux trois fonctions de déplacement, de transition et d'écriture qui définissent M, on ajoute une troisième entrée, de telle sorte que leur sortie dépend aussi du caractère lu sur la bande d'aléa.

Enfin, dans ce mémoire, nous avons besoin de considérer des machines de Turing qui peuvent échanger des messages. Pour cela, nous utilisons la notion de machine de Turing interactive. Deux machines de Turing M_1 et M_2 sont *interactives* si ce sont des machines de Turing probabilistes ayant accès à deux rubans supplémentaires r_1 et r_2 dits rubans de communication. Le ruban r_1 est en lecture seule pour M_1 et en écriture seule pour M_2, alors que le ruban r_2 est en lecture seule pour M_2 et en écriture seule pour M_1. À chaque itération, chaque machine de Turing lit le symbole présent sur le ruban de communication accessible en lecture, en fonction de ce dernier et des autres données, elle écrit un symbole sur le ruban de communication accessible en écriture et se déplace vers la droite sur les deux rubans de communication. Toutes les fonctions qui définissent les machines de Turing dépendent aussi maintenant du symbole lu sur le ruban de communication. S'il y a plus de deux machines de Turing qui communiquent, alors chaque couple de machine de Turing partage une paire de rubans de communication.

2.1.2 Cadre des définitions de sécurité

Les descriptions faites dans ce chapitre et le suivant sont largement inspirées de celles faites par Shoup dans son papier [Sho04].

Les différentes notions de sécurité qui sont présentées au cours de ce mémoire sont toutes définies comme un *jeu d'attaque* entre un *adversaire* et un maître de jeu, que nous nommerons le *challenger*. Ces derniers sont deux machines de Turing interactives. Chaque notion de sécurité définit précisément les règles du jeu qui lui sont propres, c'est-à-dire les objectifs de l'adversaire, les moyens mis à sa disposition (les oracles avec lesquels l'adversaire peut potentiellement communiquer, les requêtes qu'il est autorisé à faire à ces oracles et au challenger, le format imposé de ces requêtes, leur nombre,...) et le comportement du challenger.

Les oracles évoqués précédemment sont des oracles parfaits qui répondent en temps constant à l'adversaire et ne fournissent pas d'autre information que le résultat qu'ils sont censés fournir. Quand on veut le mettre en évidence, on note parfois par $\mathcal{A}^O$ le fait que l'adversaire $\mathcal{A}$ a accès à l'oracle O lors de son jeu de sécurité, mais quand cette dépendance est claire, nous écrirons simplement $\mathcal{A}$. Par ailleurs, nous noterons par $\mathcal{A}(x_1, \cdots, x_n)$ le fait que l'on transmet à l'adversaire les variables $x_1, \cdots, x_n$ générées pendant le jeu de sécurité : cela met en valeur le fait que le comportement de $\mathcal{A}$ dépend de ces valeurs.

L'adversaire gagne si un certain événement S est vrai. Le plus souvent, la propriété de sécurité est vraie si la probabilité que l'événement S se produise est très proche d'une valeur de référence : soit 0, soit $1/2$ par exemple.

Les règles du jeu de sécurité induisent une distribution de probabilité. Chaque événement est considéré selon cette distribution de probabilité et on ne le précisera pas toujours dans les notations. Notamment, puisque l'adversaire est une machine de Turing probabiliste, la probabilité est prise sur toutes les bandes d'aléa possibles : on dira que la probabilité porte sur les aléas de l'adversaire. Parfois, pour plus de clarté, il sera utile de détailler *en partie* les distributions de probabilité induites par le jeu de sécurité (on insiste bien sur le fait que l'on ne détaillera jamais les distributions de probabilité en entier, on se contentera de mettre en valeur les aspects les plus significatifs de cette distribution). Dans ce cas, on notera par

$$\Pr[\mathcal{D} : T]$$

la probabilité que l'événement T ait lieu en sachant que cet événement est mesuré selon la distribution $\mathcal{D}$. Typiquement $\mathcal{D}$ détaillera certaines étapes du jeu et certains messages générés par l'adversaire. Lors de ces description de probabilité, on utilise la notation $\mathcal{A} \Rightarrow x$ pour signifier que l'adversaire renvoie au challenger (ou à un oracle) le message x et la notation $\mathcal{A} \Leftarrow x$ pour

signifier que l'adversaire reçoit du challenger (ou d'un oracle) le message x. Le lecteur prendra attention à ne pas confondre la notation $\Pr[\mathcal{D}\colon T]$ avec la notation classique $\Pr[A|B]$ qui désigne la probabilité de l'événement A sachant que l'événement B est vrai.

2.1.3 Techniques de preuves

Tout au long de ce mémoire, nous utilisons la plupart du temps deux techniques de preuve pour démontrer les résultats de sécurité : soit une preuve par réduction, soit une preuve par jeux.

Les réductions sont des outils de la théorie de la complexité dont il existe plusieurs définitions. Nous considèrerons ici les réductions au sens de Turing. Supposons que l'on connaisse une machine de Turing $\mathcal{A}$ pour résoudre un problème A et que l'on veuille résoudre un problème B. Dire que l'on réduit le problème B au problème A signifie que l'on construit une machine de Turing $\mathcal{B}$ qui résout le problème B en faisant appel à $\mathcal{A}$ en « boîte noire »(c'est-à-dire qui pourrait faire appel à n'importe quelle autre machine de Turing résolvant le problème A). On attire l'attention du lecteur sur le fait que, comme $\mathcal{A}$ est une sous-routine de $\mathcal{B}$, la complexité de $\mathcal{B}$ est supérieure à celle de $\mathcal{A}$. Dans les preuves qui nous préoccupent, cette réduction doit être aussi fine que possible, c'est-à-dire que la complexité de $\mathcal{B}$ ne doit pas être beaucoup plus grande que celle de $\mathcal{A}$.

On peut utiliser une preuve par réduction lorsque l'on souhaite prouver que le problème A est difficile, sachant que le problème B l'est. En effet, si l'on arrivait à résoudre le problème A avec un algorithme $\mathcal{A}$ efficace, alors la réduction montre que l'on saurait également résoudre le problème B de façon *efficace* grâce à $\mathcal{B}$ (c'est pour cette raison que la réduction doit être fine), ce qui est exclus.

Les preuves par réductions sont parfois délicates à mettre en œuvre et il est alors plus simple et plus clair d'adopter une preuve par jeux. Proposée par Shoup [Sho01a, Sho01b], cette méthode permet de mesurer la probabilité d'un événement S par approches successives en utilisant une suite de $n+1$ jeux $\mathbf{Game}_0,\ldots,\mathbf{Game}_n$. Le jeu $\mathbf{Game}_0$ est celui qui est décrit dans la définition de la notion de sécurité considérée et on désigne par S_0 l'événement S dans l'environnement de probabilité défini par le jeu $\mathbf{Game}_0$. C'est donc la probabilité $\Pr[S_0]$ qui nous intéresse. Pour $1 \leq i \leq n$, le jeu $\mathbf{Game}_i$ est défini précisément (avec des règles différentes de celle de $\mathbf{Game}_0$) et on note par S_i l'événement S dans l'environnement de probabilité du jeu $\mathbf{Game}_i$. L'objectif est d'être capable de majorer les différences de probabilité $|\Pr[S_i] - \Pr[S_{i-1}]|$ et de pouvoir évaluer précisément $\Pr[S_n]$, on est alors capable de trouver une borne supérieure de $\Pr[S_0]$:

$$\Pr[S_0] \leq \sum_{i=1}^{n} |\Pr[S_i] - \Pr[S_{i-1}]| + \Pr[S_n].$$

On appelle *distance entre les jeux* $\boldsymbol{Game}_{i-1}$ *et* $\boldsymbol{Game}_i$, la valeur $|\Pr[S_i] - \Pr[S_{i-1}]|$.

On attire l'attention du lecteur sur le fait que les preuves par réduction et les preuves par jeux ne sont pas à opposer : il arrive que l'on utilise l'une en complément de l'autre. Il n'est pas rare notamment de faire une preuve par réduction au milieu d'une preuve par jeu.

2.2 Modèle de la théorie de l'information

Les propriétés de sécurité que nous établissons ne sont souvent vraies que sous certaines hypothèses : il faut par exemple que l'adversaire soit calculatoirement borné, il faut que tel problème soit difficile pour tout adversaire,... Les hypothèses que nous considérons dans ce mémoire n'ont pas toutes la même valeur : certaines ont plus de chances d'être vraies que d'autres ou sont plus

faciles à assurer en pratique. La valeur accordée au résultat de sécurité est directement fonction de la valeur des hypothèses dont il dépend. On regroupe habituellement les hypothèses dans quatre catégories, appelées modèles : le modèle de la théorie de l'information, le modèle standard, le modèle de l'oracle aléatoire, le modèle du chiffrement idéal. Nous détaillons les définitions de sécurité et les notions dont nous avons besoin dans chacun de ces modèles. Les modèles sont présentés ici par ordre décroissant de force : plus un modèle est fort, plus un résultat de sécurité vrai dans ce modèle a de valeur. Si un résultat est vrai dans un modèle, il est vrai dans tous les modèles moins forts. On attire l'attention du lecteur sur le fait que le modèle du chiffrement idéal et le modèle de l'oracle aléatoire ont été prouvés équivalents récemment [CPS08].

Nous commençons notre présentation par le modèle de la théorie de l'information. Il est le plus fort dans le sens où les résultats vrais dans ce modèle n'ont besoin d'aucune hypothèse dite calculatoire : la complexité des adversaires présentés ici n'est pas restreinte, elle doit juste être finie. Voici certaines notions utiles dans ce modèle.

2.2.1 Distance statistique

Dans toute la suite, si $\mathcal{X}$ est un ensemble fini, on notera $\mathcal{U}_{\mathcal{X}}$ la distribution uniforme sur $\mathcal{X}$, $U_{\mathcal{X}}$ une variable aléatoire uniformément distribuée sur $\mathcal{X}$. Si $\mathcal{X} = \{0,1\}^{\ell}$, on simplifiera cette notation en écrivant $\mathcal{U}_{\ell}$ et U_{ℓ} respectivement. Si $\mathcal{D}_{\mathcal{X}}$ est une distribution sur $\mathcal{X}$ et X est une variable aléatoire, on note $X \leftarrow \mathcal{D}_{\mathcal{X}}$ le fait de choisir X aléatoirement selon $\mathcal{D}_{\mathcal{X}}$. Si X est choisit de façon uniforme sur $\mathcal{X}$, on le notera parfois $X \xleftarrow{\$} \mathcal{X}$ plutôt que $X \leftarrow \mathcal{U}_{\mathcal{X}}$. Enfin, on dira abusivement qu'une variable aléatoire sur $\mathcal{X}$ est choisie *parfaitement au hasard* ou *parfaitement aléatoirement* pour signifier qu'elle est choisie selon la distribution uniforme sur $\mathcal{X}$.

Définition 2.2 (Distance statistique)**.** *Soient $\mathcal{D}_1$ et $\mathcal{D}_2$ deux distributions de probabilité sur le même ensemble $\mathcal{X}$. La* distance statistique *entre les deux distributions est définie par :*

$$\mathbf{SD}\left(\mathcal{D}_1, \mathcal{D}_2\right) \stackrel{def}{=} \frac{1}{2} \sum_{x \in \mathcal{X}} \left| \Pr_{X_1 \leftarrow \mathcal{D}_1}[X_1 = x] - \Pr_{X_2 \leftarrow \mathcal{D}_2}[X_2 = x] \right|.$$

On définit facilement la différence statistique entre deux variables aléatoires X_1 et X_2 sur le même ensemble $\mathcal{X}$. Soient $\mathcal{D}_1$ et $\mathcal{D}_2$ les deux distributions de probabilité de X_1 et X_2 respectivement, la différence statistique entre X_1 et X_2 est égale à $\mathbf{SD}(X_1, X_2) \stackrel{def}{=} \mathbf{SD}(\mathcal{D}_1, \mathcal{D}_2)$. On utilisera indifféremment l'une ou l'autre.

La distance statistique est toujours comprise entre 0 et 1. Elle est aussi parfois appelée *distance* L_1 puisqu'elle dérive de la norme L_1 sur l'ensemble des distributions de probabilité. Elle est très utilisée en cryptographie car elle a une interprétation naturelle que l'on ne retrouve pas pour la distance L_2 par exemple. Cette interprétation est exprimée dans la proposition suivante.

Soient X_1 et X_2 deux variables aléatoires sur le même ensemble $\mathcal{X}$. Considérons le jeu suivant : le challenger choisit parfaitement au hasard une valeur $b \in \{1,2\}$, puis envoie un échantillon de la variable aléatoire X_b à l'adversaire. Cet adversaire, au vue de cet échantillon, essaie de deviner quelle est la valeur de b en renvoyant au challenger un bit b'. Il gagne si $b = b'$. Il existe un lien entre la probabilité de gagner de l'adversaire et la distance statistique entre X_1 et X_2. Ce lien est explicité dans la proposition ci-dessous.

Proposition 2.1. *Soit $\mathcal{A}$ un adversaire (qui peut être calculatoirement non borné) dans le jeu précédent. On note par $\Pr[b = b']$ sa probabilité de gagner.*

(i) On a la relation $\Pr[b = b'] \leq 1/2 + \mathbf{SD}(X_1, X_2)/2$.

(ii) Il existe un adversaire $\mathcal{A}'$ dont la probabilité de gagner est telle que $\Pr[b = b'] = 1/2 + \mathbf{SD}(X_1, X_2)/2$.

Cette proposition est bien connue et simple à montrer, nous donnons une version classique de cette preuve.

Démonstration. Commençons par montrer le point (i).

$$\begin{aligned}
\Pr[b = b'] &= \Pr[b = 1]\Pr[b' = 1 | b = 1] + \Pr[b = 2]\Pr[b' = 2 | b = 2] \\
&= \frac{1}{2}\sum_x \big(\Pr[b' = 1 | \mathcal{A} \Leftarrow x]\Pr[X_1 = x] + \Pr[b' = 2 | \mathcal{A} \Leftarrow x]\Pr[X_2 = x]\big) \\
&= \frac{1}{2}\sum_x \big((1 - \Pr[b' = 2 | \mathcal{A} \Leftarrow x])\Pr[X_1 = x] + \Pr[b' = 2 | \mathcal{A} \Leftarrow x]\Pr[X_2 = x]\big) \\
&= \frac{1}{2}\sum_x \big(\Pr[X_1 = x] + \Pr[b' = 2 | \mathcal{A} \Leftarrow x]\,(\Pr[X_2 = x] - \Pr[X_1 = x])\big) \\
&= \frac{1}{2} + \frac{1}{2}\sum_x \Pr[b' = 2 | \mathcal{A} \Leftarrow x](\Pr[X_2 = x] - \Pr[X_1 = x])
\end{aligned}$$

Notons $\mathcal{X}_1$ l'ensemble des x tels que $\Pr[X_2 = x] \leq \Pr[X_1 = x]$ et $\mathcal{X}_2$ l'ensemble des x tels que $\Pr[X_1 = x] \leq \Pr[X_2 = x]$. On a évidemment que $\sum_{x\in\mathcal{X}_1}(\Pr[X_2 = x] - \Pr[X_1 = x]) + \sum_{x\in\mathcal{X}_2}(\Pr[X_2 = x] - \Pr[X_1 = x])$ est égale à $\sum_x \Pr[X_2 = x] - \sum_x \Pr[X_1 = x] = 0$, donc

$$\sum_{x\in\mathcal{X}_1}(\Pr[X_2 = x] - \Pr[X_1 = x]) = -\sum_{x\in\mathcal{X}_2}(\Pr[X_2 = x] - \Pr[X_1 = x])\,.$$

Par ailleurs, on a choisi $\mathcal{X}_1$ et $\mathcal{X}_2$ tels que

$$\sum_{x\in\mathcal{X}_1}\big(\Pr[X_2 = x] - \Pr[X_1 = x]\big) = -\sum_{x\in\mathcal{X}_1}\big|\Pr[X_2 = x] - \Pr[X_1 = x]\big| \leq 0$$

et

$$\sum_{x\in\mathcal{X}_2}\big(\Pr[X_2 = x] - \Pr[X_1 = x]\big) = \sum_{x\in\mathcal{X}_2}\big|\Pr[X_2 = x] - \Pr[X_1 = x]\big|.$$

Par conséquent,

$$\sum_{x\in\mathcal{X}_1}\big|\Pr[X_2 = x] - \Pr[X_1 = x]\big| = \sum_{x\in\mathcal{X}_2}\big|\Pr[X_2 = x] - \Pr[X_1 = x]\big|.$$

Par ailleurs, comme $\Pr[b' = 2|\mathcal{A} \Leftarrow x]$ est toujours positif, on déduit facilement de ce qui précède que $\sum_x \Pr[b' = 2|\mathcal{A} \Leftarrow x](\Pr[X_2 = x] - \Pr[X_1 = x])$ est inférieur à :

$$\begin{aligned}
\sum_{x\in\mathcal{X}_2}\Pr[b' = 2|\mathcal{A} \Leftarrow x]\big|\Pr[X_2 = x] - \Pr[X_1 = x]\big| &\leq \sum_{x\in\mathcal{X}_2}\big|\Pr[X_2 = x] - \Pr[X_1 = x]\big| \\
&= \frac{1}{2}\sum_{x\in\mathcal{X}}\big|\Pr[X_2 = x] - \Pr[X_1 = x]\big|.
\end{aligned}$$

On déduit finalement que :

$$\begin{aligned}
\Pr[b = b'] &\leq \frac{1}{2} + \frac{1}{2}\cdot\frac{1}{2}\sum_x\big|\Pr[X_2 = x] - \Pr[X_1 = x]\big| \\
&= \frac{1}{2} + \frac{1}{2}\mathbf{SD}(X_1, X_2).
\end{aligned}$$

Ceci prouve le point (i). Cette preuve nous incite à introduire l'adversaire $\mathcal{A}'$ qui renvoie 2 si $\Pr[X_2 = x] \geq \Pr[X_1 = x]$ et 1 sinon. En reprenant la preuve précédente, on voit facilement que pour $\mathcal{A}'$ toutes les inégalités sont des égalités et donc on prouve le point (ii). □

L'adversaire $\mathcal{A}'$ décrit précédemment est optimal, mais déterminer si $\Pr[X_2 = x] \geq \Pr[X_1 = x]$ demande souvent beaucoup de calcul et cet attaquant est alors inefficace et ne peut être utilisé en pratique. On voit que majorer la distance statistique permet de majorer la probabilité de succès de tous les attaquants. C'est en ce sens que le modèle de la théorie de l'information est fort : il permet d'obtenir des résultats de sécurité valables contre tous les attaquants de façon inconditionnelle.

À partir de la distance statistique on dérive la notion de δ-uniformité :

Définition 2.3 (δ-uniformité)**.** *Soient $\mathcal{D}$ une distribution de probabilité sur $\mathcal{X}$ et $\mathcal{U}_\mathcal{X}$ la distribution uniforme sur $\mathcal{X}$. La distribution $\mathcal{D}$ est dite δ-uniforme si :*

$$\mathbf{SD}(\mathcal{U}_\mathcal{X}, \mathcal{D}) \leq \delta.$$

Comme d'habitude, on déduit facilement de cette définition, la notion de δ-uniformité pour une variable aléatoire.

2.2.2 L'entropie et ses mesures

La notion d'entropie d'une distribution de probabilité a été introduite par Shannon en 1949 en même temps qu'il a donné la définition de l'entropie de Shannon. Depuis, cette notion d'entropie a été généralisée, donnant lieu à de nombreuses mesures de l'entropie dont l'entropie de Shannon n'est qu'un cas particulier.

Quelle que soit la mesure que l'on considère, l'entropie d'une distribution de probabilité, ou d'une variable aléatoire, mesure l'aléa présent dans cette distribution. Dit autrement, elle mesure avec quelle « incertitude » on peut deviner la sortie d'une variable aléatoire. La distribution concentrée en un point ne contient pas d'incertitude quant à sa sortie, son entropie vaut toujours 0, quelle que soit la définition d'entropie utilisée. De même parmi toutes les distributions de probabilité sur un ensemble donné, celle ayant le plus d'aléa, et donc celle ayant le plus d'entropie, est toujours la distribution uniforme sur cet ensemble.

L'entropie de Shannon est une, si ce n'est *la* définition de l'entropie la plus utilisée actuellement. Cependant dans ce rapport, nous ferons appel plutôt à deux autres mesures d'entropie, la min-entropie et l'entropie de collision, qui sont souvent plus adaptées aux besoins cryptographiques. Nous ne donnerons pas la définition générale de l'entropie, mais nous nous concentrerons sur ces deux dernières.

Commençons par présenter la min-entropie, également appelée parfois l'entropie d'estimation.

Définition 2.4 (Probabilité d'estimation)**.** *Soit X une variable aléatoire à valeurs dans l'ensemble $\mathcal{X}$ et $\mathcal{D}$ sa distribution de probabilité. Leur probabilité d'estimation est $\gamma(\mathcal{D}) = \gamma(X) \stackrel{def}{=} \max_{x \in \mathcal{X}} \Pr[X = x]$.*

Définition 2.5 (Min-entropie)**.** *Soit X une variable aléatoire à valeurs dans l'ensemble $\mathcal{X}$ et $\mathcal{D}$ sa distribution de probabilité. Leur min-entropie est*

$$\mathbf{H}_\infty(\mathcal{D}) = \mathbf{H}_\infty(X) \stackrel{def}{=} -\log_2(\gamma(X)) = -\log_2(\max_{x \in \mathcal{X}} \Pr[X = x]).$$

La min-entropie mesure la difficulté qu'a un adversaire à prévoir la valeur prise par la variable aléatoire X. Ce dernier maximise ses chances de deviner X en un coup en choisissant la valeur de X la plus probable. Plus la probabilité d'occurrence de cette valeur est faible (et ainsi plus la min-entropie est élevée), plus la probabilité de succès de l'attaquant est faible (et donc plus il est difficile pour lui de deviner la valeur prise par X).

Nous nous intéressons maintenant à l'entropie de collision.

Définition 2.6 (Probabilité de collision)**.** *Soit X une variable aléatoire à valeurs dans l'ensemble $\mathcal{X}$ et $\mathcal{D}$ sa distribution de probabilité. Soit X' une autre variable aléatoire indépendante et de même distribution que X. La probabilité de collision de X et de $\mathcal{D}$ est* $\mathrm{Col}(\mathcal{D}) = \mathrm{Col}(X) \stackrel{def}{=} \Pr[X = X'] = \sum_{x\in\mathcal{X}} \Pr[X = x]^2$.

Comme son nom l'indique, la probabilité de collision mesure la probabilité que, si on effectue deux fois la même expérience, on obtienne deux fois le même résultat : si on tire deux fois de façon indépendante la même variable aléatoire, elle donne la probabilité d'obtenir deux fois la même valeur. La probabilité de collision peut également être interprétée en termes de norme L_2, puisque cette dernière est égale à : $\|X\|_2 = \sqrt{\sum_{x\in\mathcal{X}} \Pr[X = x]^2} = \sqrt{\mathrm{Col}(X)}$.

Définition 2.7 (Entropie de collision)**.** *Soit X une variable aléatoire à valeurs dans l'ensemble $\mathcal{X}$ et $\mathcal{D}$ sa distribution de probabilité. Leur entropie de collision est*

$$\mathbf{H}_2(\mathcal{D}) = \mathbf{H}_2(X) \stackrel{def}{=} -\log_2(\mathrm{Col}(X)) = -\log_2\left(\sum_{x\in\mathcal{X}} \Pr[X = x]^2\right).$$

L'entropie de collision est aussi souvent appelée entropie de Renyi. Il existe une relation simple entre min-entropie et entropie de collision, relation qui s'exprime aussi en termes de probabilité de collision et probabilité d'estimation :

$$\gamma(X)^2 \leq \mathrm{Col}(X) \leq \gamma(X),$$
$$\mathbf{H}_\infty(X) \leq \mathbf{H}_2(X) \leq 2\mathbf{H}_\infty(X).$$

2.2.3 Les extracteurs d'aléa

Informellement, les extracteurs d'aléa, qui ont été introduits pour la première fois par [NZ96], sont des fonctions qui transforment une longue chaîne de bits contenant beaucoup d'aléa en une chaîne de bits plus courte et parfaitement aléatoire, c'est-à-dire uniformément distribuée. Ils peuvent avoir recours à un aléa parfait additionnel qui sert de catalyseur pour extraire l'aléa présent dans la longue chaîne de bits. Cette chaîne de bits uniformément distribuée supplémentaire s'appelle la *graine* (comme dans un générateur pseudo-aléatoire). Sa taille est fixée par l'extracteur, on l'appelle *taille de la graine* de l'extracteur. La définition générale d'un extracteur d'entropie est la suivante.

Définition 2.8 (Extracteur d'aléa)**.** *Une fonction probabiliste calculable en temps polynomial* $\mathsf{Ext}\colon \{0,1\}^d \times \{0,1\}^n \to \{0,1\}^k$ *est un (t,ε)-extracteur si pour toute variable aléatoire W de min-entropie au moins t, on a* $\mathbf{SD}(\mathsf{Ext}(U_d; W), U_k) \leq \varepsilon$ *où* $\mathsf{Ext}(U_d; W)$ *désigne la sortie de* Ext *appliquée à W en utilisant l'aléa U_d.*

Parmi l'ensemble des extracteurs, on distingue deux sous-classes importantes : les extracteurs forts et les extracteurs déterministes. Nous nous intéresserons uniquement à ces deux sous-classes dans ce mémoire.

Les extracteurs forts ont la particularité d'inclure leur graine dans leur sortie. Cela signifie que même si cette graine est connue de tous, la sortie de l'extracteur reste uniformément distribuée aux yeux de l'attaquant. Cela a une conséquence très importante en pratique : la graine peut être rendue publique, ce qui facilite son échange entre les participants d'un protocole.

Définition 2.9 (Extracteur fort d'aléa)**.** *Une fonction probabiliste calculable en temps polynomial* $\mathsf{Ext}\colon \{0,1\}^d \times \{0,1\}^n \to \{0,1\}^k$ *est un* (t,ε)*-extracteur fort si pour toute variable aléatoire* W *de min-entropie au moins* t*, on a* $\mathbf{SD}\big(\,(\mathsf{Ext}(U_d;W),U_d)\,,(U_k,U_d)\,\big) \leq \varepsilon$ *où* $\mathsf{Ext}(U_d;W)$ *désigne la sortie de* Ext *appliquée à* W *en utilisant l'aléa* U_d*.*

La différence $t-k$ est habituellement appelée *la perte d'entropie* de l'extracteur (il s'agit de la différence entre la min entropie de l'entrée et la min entropie de la sortie). Dans la définition précédente, la variable U_d est la graine et d la taille de la graine de l'extracteur.

Comme prouvé par [RTS97], les extracteurs fort d'aléa peuvent extraire jusqu'à $k = t - 2\log_2(1/\varepsilon) + \mathcal{O}(1)$ bits. Plusieurs constructions atteignant cette borne [Sha02], la tendance actuelle est de chercher à minimiser la taille de la graine. Pour ce mémoire, la taille de la graine sera peu importante c'est pourquoi on se contente de s'intéresser aux fonctions de hachage deux à deux indépendantes, ou *pairwise independent hash functions*, qui permettent de construire un extracteur fort optimal. Nous allons maintenant donner la construction de cet extracteur. Pour cela nous introduisons les définitions suivantes.

Définition 2.10 (Indépendance deux à deux, XOR universalité, famille de fonctions de hachage δ-presque universelle)**.** *Soit* $\mathcal{H} = \{h_i\}_{i\in\mathcal{I}}$ *un ensemble de fonctions de* $\{0,1\}^n$ *dans* $\{0,1\}^k$ *calculables efficacement, indicé par un ensemble* $\mathcal{I}$*. Soit* h_I *une fonction choisie uniformément dans* $\mathcal{H}$*.*

- *La famille de fonctions de hachage* $\mathcal{H}$ *est dite* deux à deux indépendantes *si pour tout couple* (x,y) *d'éléments distincts de* $\{0,1\}^n$*, les variables aléatoires* $h_I(x)$ *et* $h_I(y)$ *sont indépendantes et uniformément distribuées dans* $\{0,1\}^k$*, c'est-à-dire :*

$$\forall\, x,y \in \{0,1\}^n, x \neq y, \forall a,b \in \{0,1\}^k : \Pr_{I\leftarrow\mathcal{U}_\mathcal{I}}[h_I(x) = a \wedge h_I(y) = b] = \frac{1}{2^{2k}}.$$

- *La famille de fonctions de hachage* $\mathcal{H}$ *est dite* XOR universelle *si pour tout couple* (x,y) *d'éléments distincts de* $\{0,1\}^n$*, la différence* $h_I(x)\oplus h_I(y)$ *est uniformément distribuée dans* $\{0,1\}^k$*, c'est-à-dire :*

$$\forall\, x,y \in \{0,1\}^n, x \neq y, \forall a \in \{0,1\}^k : \Pr_{I\leftarrow\mathcal{U}_\mathcal{I}}[h_I(x) \oplus h_I(y) = a] = \frac{1}{2^k}.$$

- *La famille de fonctions de hachage* $\mathcal{H}$ *est dite* δ-presque universelle *si pour tout* (x,y) *d'éléments distincts de* $\{0,1\}^n$*, la probabilité que les sorties* $h_I(x)$ *et* $h_I(y)$ *collisionnent est faible, c'est-à-dire :*

$$\forall\, x,y \in \{0,1\}^n, x \neq y : \Pr_{I\leftarrow\mathcal{U}_\mathcal{I}}[h_I(x) = h_I(y)] \leq \frac{1}{2^k} + \delta.$$

  Si $\delta = 0$ *la famille de fonctions est dite parfaitement* universelle*.*

Précisons que la notation $\oplus$ désigne l'addition bit par bit modulo 2 et que dans tout ce mémoire, pour des questions de simplicité évidente, nous parlerons simplement de « a xor b » pour désigner l'expression $a \oplus b$. Pour les mêmes raisons, nous inventons également le verbe « xorer » pour signifie « additionner bit par bit modulo 2 ».

On remarque que l'indépendance deux à deux implique la XOR universalité et qu'une famille XOR universelle est aussi une famille parfaitement universelle.

Un exemple de famille de fonctions XOR universelle est l'ensemble des fonctions linéaires de $\{0,1\}^n$ vers $\{0,1\}^k$ (la linéarité est considérée sur $(\mathbb{Z}_2, \oplus)$). On a alors $\mathcal{I} = \{0,1\}^{n \cdot k}$. Si $k = n$ la famille de fonctions $h_a \colon x \mapsto a \cdot x$, $a \in \{0,1\}^n$, où la multiplication est faite dans $\mathbb{F}_{2^n}$, est aussi XOR universelle et a l'avantage d'avoir un ensemble $\mathcal{I} = \{0,1\}^n$ plus petit que la précédente. On peut généraliser cette construction aux cas où $k \neq n$. Si k est plus grand que n alors on travaille sur $\mathbb{F}_{2^k}$: on considère l'ensemble des fonctions $h_a \colon x \mapsto a \cdot x$, $a \in \{0,1\}^k$, où la multiplication est faite dans $\mathbb{F}_{2^k}$. Si k est plus petit que n alors on tronque la sortie en ne conservant que les k bits de poids faible, ce qui donne la famille des fonctions $h_a \colon x \mapsto (a \cdot x) \mod 2^k$, $a \in \{0,1\}^n$, où la multiplication est faite dans $\mathbb{F}_{2^n}$[1].

Ces familles de fonctions permettent de construire facilement des extracteurs forts d'entropie grâce au résultat suivant, appelé le « leftover hash lemma ». Il s'agit historiquement du premier extracteur présenté dans la littérature [ILL89] et probablement également du plus simple. Il dit qu'une famille de fonctions δ-presque universelle est un bon extracteur d'entropie si δ est suffisamment proche de 0 (et *a fortiori* si $\delta = 0$, c'est-à-dire si la famille de fonctions est parfaitement universelle).

Lemme 2.1 (*Leftover hash lemma*)**.** *Soit $\mathcal{H} = \{h_i\}_{i \in \mathcal{I}}$ une famille de fonctions de hachage δ-presque universelle de n bits vers k bits. Si $t \geq k + 2e$ et si $\delta \leq 2^{-2e-k}$, alors $\mathcal{H}$ est un $(t, 2^{-e})$-extracteur fort.*

Ce lemme nous est très utile par la suite, notamment car nous en présentons une version calculatoire dont la preuve se calque sur la preuve de la version classique. Pour cette raison, nous allons en donner la démonstration. Mais avant de prouver ce lemme, nous allons présenter quelques résultats intermédiaires utiles.

Le lemme suivant s'interprète en terme de normes pour les suites sur des ensembles finis : il précise la constante telle que la norme L_1 est inférieure à la norme L_2 multipliée par cette constante.

Lemme 2.2. *Soient $\mathcal{X}$ un ensemble fini et $(\alpha_x)_{x \in \mathcal{X}}$ une suite dans l'ensemble des réels. On a :*

$$\frac{\left(\sum_{x \in \mathcal{X}} |\alpha_x|\right)^2}{|\mathcal{X}|} \leq \sum_{x \in \mathcal{X}} \alpha_x^2 \tag{2.1}$$

Démonstration. Cette inégalité est une conséquence directe de l'inégalité de Cauchy-Schwarz :

$$\sum_{x \in \mathcal{X}} |\alpha_x| = \sum_{x \in \mathcal{X}} |\alpha_x| \cdot 1 \leq \sqrt{\sum_{x \in \mathcal{X}} \alpha_x^2} \cdot \sqrt{\sum_{x \in \mathcal{X}} 1^2} \leq \sqrt{|\mathcal{X}|} \sqrt{\sum_{x \in \mathcal{X}} \alpha_x^2}.$$

Le résultat s'en déduit facilement. □

Dans le cas des probabilités, si X est une variable aléatoire à valeurs dans $\mathcal{X}$, en posant $\alpha_x = \Pr[X = x]$, comme la somme des probabilités est égale à 1 et comme $\mathrm{Col}(X) = \sum_{x \in \mathcal{X}} \Pr[X = x]^2$, on obtient :

$$\frac{1}{|\mathcal{X}|} \leq \mathrm{Col}(X). \tag{2.2}$$

Ce résultat permet également de mettre en relation la probabilité de collision de la variable X et la distance statistique entre X et la variable uniformément distribuée sur $\mathcal{X}$.

[1] Cette famille est XOR universelle car $a \cdot x \oplus a \cdot y = b \mod 2^k$ si et seulement si il existe $c \in \{0,1\}^{n-k}$ tel que $a = (2^k \cdot c \oplus b) \cdot (x \oplus y)^{-1}$, ce qui signifie que pour tout $x \neq y$, il existe exactement 2^{n-k} valeurs de a telles que $h_a(x) \oplus h_a(y) = b$ parmi les 2^n valeurs possibles.

Lemme 2.3. *Soient X une variable aléatoire à valeurs dans un ensemble $\mathcal{X}$ et $\varepsilon = SD(X, U_{\mathcal{X}})$ la distance statistique entre X et la variable uniformément distribuée sur $\mathcal{X}$. On a :*

$$\mathrm{Col}(X) \geq \frac{1 + 4\varepsilon^2}{|\mathcal{X}|}. \tag{2.3}$$

Démonstration. On suppose que ε est différent de 0, sinon le résultat est évident d'après l'équation (2.2). On introduit $q_x = |\Pr[X = x] - 1/|\mathcal{X}||/(2\varepsilon)$, on a donc $\sum_x q_x = 1$. D'après l'équation (2.1), on a :

$$\begin{aligned}\frac{1}{|\mathcal{X}|} &\leq \sum_{x\in\mathcal{X}} q_x^2 = \frac{1}{4\varepsilon^2}\sum_{x\in\mathcal{X}} (\Pr[X = x] - 1/|\mathcal{X}|)^2 = \frac{1}{4\varepsilon^2}\left(\sum_{x\in\mathcal{X}} \Pr[X = x]^2 - 1/|\mathcal{X}|\right) \\ &\leq \frac{1}{4\varepsilon^2}\left(\mathrm{Col}(X) - 1/|\mathcal{X}|\right).\end{aligned}$$

On en déduit directement le résultat attendu. □

Nous nous attaquons maintenant à la preuve du *leftover hash lemma.*

Démonstration. La preuve que nous allons exposer provient du livre de Shoup [Sho05].

Soient I et I' deux variables indépendantes uniformément distribuées dans $\mathcal{I}$ et X et X' deux variables indépendantes et identiquement distribuées dans $\mathcal{X}$.

$$\begin{aligned}\mathrm{Col}(I, h_I(X)) &= \Pr[I = I' \wedge h_I(X) = h_{I'}(X')] = \Pr[I = I']\Pr[h_I(X) = h_I(X')] \\ &= \frac{1}{|\mathcal{I}|}\Big(\Pr[h_I(X) = h_I(X')|X = X']\Pr[X = X'] \\ &\quad + \Pr[h_I(X) = h_I(X')|X \neq X']\Pr[X \neq X']\Big) \\ &\leq \frac{1}{|\mathcal{I}|}\left(\mathrm{Col}(X) + \Pr[h_I(X) = h_I(X')|X \neq X']\right) \\ &\leq \frac{1}{|\mathcal{I}|}\left(\mathrm{Col}(X) + \frac{1}{2^k} + \delta\right).\end{aligned}$$

Or le résultat (2.3) appliqué à $(I, h_I(X))$ nous dit que $(1 + 4\varepsilon^2)/(2^k|\mathcal{I}|) \leq \mathrm{Col}(I, h_I(X))$, où ε est la distance statistique entre $(I, h_I(X))$ et la distribution uniforme. On peut donc majorer ε par $\sqrt{2^k(\mathrm{Col}(X) + \delta)}/2$. Comme pour tout a et b positifs, $\sqrt{a+b} \leq \sqrt{a} + \sqrt{b}$, ε est inférieur à $\left(\sqrt{2^k\mathrm{Col}(X)} + \sqrt{2^k\delta}\right)/2$. Enfin, puisque $\mathrm{Col}(X) \leq \gamma(X) = 2^{-t}$, si $2^{(k-t)/2} \leq 2^{-e}$ et $\sqrt{2^k\delta} \leq 2^{-e}$ alors $\varepsilon \leq 2^{-e}$. □

On remarque que comme une famille de fonctions deux à deux indépendantes est parfaitement universelle, le lemme s'applique avec $\delta = 0$ et la dernière construction présentée ci-dessus est donc un extracteur fort dont la taille de la graine est $\log_2|\mathcal{I}| = n$ et la perte d'entropie est $2e$.

Les extracteurs forts comme ceux issus du *Leftover Hash Lemma* peuvent être utilisées avec en entrée n'importe quelle variable aléatoire, pour peu qu'elle ait suffisamment de min-entropie. Cependant, il nécessite un aléa supplémentaire dont on peut vouloir se passer. Or, il a été prouvé qu'une fonction qui n'utiliserait pas cet aléa supplémentaire ne peut pas être un extracteur pour n'importe quelle variable aléatoire[2]. Par contre, ce résultat n'interdit pas de créer une fonction

[2]Voici comment Dodis le prouve dans sa thèse [Dod00]. Si $f\colon \{0,1\}^n \mapsto \{0,1\}^k$ est un extracteur, considérant la variable aléatoire X dont toute la masse est concentrée de façon uniforme sur les préimages des 2^{t+k-n} points les plus fréquents de l'image de f. Il y a au moins 2^t telles préimages donc X a une min entropie supérieure à t. Cependant, $f(X)$ ne peut être statistiquement proche de l'uniforme que si il contient strictement plus que $k-1$ bits d'entropie, donc si $m + k - n > k - 1$, ce qui signifie que $m > n - 1$: X est déjà presque uniformément distribuée et il n'est pas nécessaire d'extraire d'entropie !

qui n'utilise pas d'aléa supplémentaire mais qui soit un bon extracteur pour une classe restreinte de variables aléatoires. Un tel extracteur est ce que l'on appelle un extracteur déterministe et a été introduit dans [TV00].

Définition 2.11 (Extracteur Déterministe)**.** *Soit un ensemble $\mathcal{W}$ de variables aléatoires. La fonction calculable en temps polynomial $\mathsf{Ext}\colon \{0,1\}^n \to \{0,1\}^k$ est un $(\mathcal{W},\varepsilon)$-extracteur déterministe si pour toute variable aléatoire $W \in \mathcal{W}$ de min-entropie au moins t, on a $\mathbf{SD}(\mathsf{Ext}(W), U_k) \leq \varepsilon$.*

Des exemples d'extracteurs déterministes sont présentés dans le chapitre 5 de la partie III.

2.3 Modèle standard

Dans le modèle standard, les attaquants peuvent être calculatoirement limité : on dira qu'un attaquant est *calculatoirement limité*, s'il existe une borne T telle que, dans tous les cas, la complexité de l'attaquant doit être inférieure à T. On parlera parfois de la complexité *temporelle* de l'attaquant, pour insister sur le fait que cet attaquant est calculatoirement limité. C'est le modèle qui est considéré le plus proche de la réalité, car les algorithmes que l'on utilise en pratique doivent être efficaces, pour qu'on puisse en voir le bout.

Le modèle standard est moins fort que le modèle de la théorie de l'information. Cela se comprend aisément puisque les résultats vrais en théorie de l'information s'appliquent inconditionnellement à tous les attaquants, en particulier aux attaquants calculatoirement limités, et sont donc aussi vrais dans le modèle standard. Cela est illustré par la proposition 2.1 qui met en lien la distance statistique entre deux distributions avec la probabilité qu'un attaquant parvienne à distinguer ces deux distributions : cette proposition s'applique en particulier aux adversaires calculatoirement limités.

Au détriment d'hypothèses plus fortes, dites hypothèses calculatoires, le modèle standard permet d'obtenir plus de résultats que le modèle de la théorie de l'information. Une hypothèse calculatoire suppose qu'aucun attaquant calculatoirement limité ne peut casser un certain problème. En effet, certains problèmes qui sont faciles pour un adversaire tout puissant peuvent être très difficiles pour un adversaire calculatoirement limité. Il existe certaines hypothèses calculatoires, introduites dans la littérature il y a parfois plus de 30 ans, qui n'ont jamais été cassées de façon efficaces. Pour cette raison, elles sont considérées maintenant comme réalistes. De même, il existe maintenant plusieurs primitives cryptographiques dont on maîtrise bien la conception et contre lesquelles des adversaires calculatoirement limités ne peuvent souvent rien faire. Ces hypothèses et ces primitives sont très utilisées pour construire des protocoles. Nous présentons dans cette section les hypothèses et les primitives qui sont utiles dans ce mémoire.

2.3.1 Hypothèses calculatoires

Voici les deux hypothèses calculatoires dont nous avons besoin dans ce mémoire.

Problème Diffie-Hellman calculatoire. Soit $\mathbb{G}$ un groupe multiplicatif cyclique d'ordre q. Soit g un générateur de $\mathbb{G}$, soit (x,y) deux entiers choisis aléatoirement de façon uniforme dans $\mathbb{Z}_q$. Le problème calculatoire Diffie-Hellman consiste à calculer, étant donné (g^x, g^y), l'élément $g^{xy} = \mathsf{CDH}_{\mathbb{G},g}(g^x, g^y)$.

Soit $\mathcal{A}$ un adversaire-CDH avec une complexité temporelle inférieure à T. Nous notons $\mathsf{Succ}^{\mathsf{cdh}}_{\mathbb{G},g}(\mathcal{A})$ la probabilité que l'adversaire $\mathcal{A}$ réussisse à calculer g^{xy} à partir de (g^x, g^y) et notons $\mathsf{Succ}^{\mathsf{cdh}}_{\mathbb{G},g}(T) = \max_{\mathcal{A}}\{\mathsf{Succ}^{\mathsf{cdh}}_{\mathbb{G},g}(\mathcal{A})\}$ où le maximum est pris sur tous les adversaires avec complexité temporelle au plus T.

Dans le cas où $\mathbb{G}$ est un sous groupe de $\mathbb{Z}_p^\star$ avec p premier, alors on utilisera plutôt les notations $\mathsf{CDH}_{p,g}(g^x, g^y)$, $\mathsf{Succ}_{p,g}^{\mathsf{cdh}}(\mathcal{A})$ et $\mathsf{Succ}_{p,g}^{\mathsf{cdh}}(T)$.

On dira que l'on fait l'hypothèse Diffie-Hellman calculatoire ou hypothèse CDH, si $\mathsf{Succ}_{\mathbb{G},g}^{\mathsf{cdh}}(T)$ est considérée comme négligeable.

Problème Diffie-Hellman décisionnel. Soit $\mathbb{G}$ un groupe multiplicatif cyclique d'ordre q. Soit g un générateur de $\mathbb{G}$. Le problème Diffie-Hellman décisionnel consiste à distinguer les distributions $\{(x,y) \leftarrow \mathbb{Z}_q : (g^x, g^y, g^{xy})\}$ et $\{(x,y,z) \leftarrow \mathbb{Z}_q : (g^x, g^y, g^z)\}$.

Plus précisément, dans le jeu DDH, un challenger choisit au hasard au début du jeu trois éléments (x,y,z) de façon uniforme dans $\mathbb{Z}_q$ et un bit b de façon uniforme dans $\{0,1\}$. Si $b=1$ alors il envoie (g^x, g^y, g^{xy}), sinon il envoie (g^x, g^y, g^z). Si l'adversaire-DDH $\mathcal{A}$ a une complexité temporelle inférieure à T, alors au bout du temps T l'adversaire $\mathcal{A}$ renvoie un bit b' et gagne si $b = b'$.

Nous notons $\mathsf{Succ}_{\mathbb{G},g}^{\mathsf{ddh}}(\mathcal{A})$ la probabilité que $\mathcal{A}$ réussisse à distinguer ces deux distributions, c'est-à-dire la probabilité que $b = b'$, et notons $\mathsf{Succ}_{\mathbb{G},g}^{\mathsf{ddh}}(T) = \max_{\mathcal{A}}\{\mathsf{Succ}_{\mathbb{G},g}^{\mathsf{ddh}}(\mathcal{A})\}$ où le maximum est pris sur tous les adversaires avec complexité temporelle au plus T.

On dira que l'on fait l'hypothèse Diffie-Hellman décisionnelle, appelée également hypothèse DDH, si $\mathsf{Succ}_{\mathbb{G},g}^{\mathsf{ddh}}(T)$ est considérée comme négligeable. Le groupe $\mathbb{G}$ est alors appelé un groupe DDH.

2.3.2 Primitives cryptographiques

Nous présentons maintenant deux des primitives dont nous avons besoin dans ce mémoire.

Permutations pseudo-aléatoires

Nous notons par $\mathfrak{S} = \mathfrak{S}_{\mathcal{D}om}$ l'ensemble des permutations de $\mathcal{D}om$. Soit $\Pi : \mathcal{K}eys \times \mathcal{D}om \to \mathcal{D}om$ une famille de permutations, on dit que c'est une famille de permutations pseudo-aléatoires si l'avantage d'un adversaire dans le jeu suivant est faible. Avant le début du jeu, le challenger choisit un bit b au hasard et, si $b = 1$ il choisit π au hasard dans $\mathfrak{S}$, sinon il choisit une clef K au hasard et affecte $\pi = \Pi(K, \cdot)$. Ce que l'on appelle l'adversaire PRP $\mathcal{A}$ (pour *Pseudo-Random Permutation*) peut alors interagir avec π en posant au plus q requêtes x_i dans $\mathcal{D}om$, $1 \leq i \leq q$, et reçoit en échange $\pi(x_i)$. Enfin l'adversaire renvoie un bit b' et gagne si $b = b'$. L'avantage en PRP de cet attaquant, noté $\mathsf{adv}_{\Pi}^{\mathtt{prp}}(\mathcal{A})$, vaut :

$$\mathsf{adv}_{\Pi}^{\mathtt{prp}}(\mathcal{A}) = \left|\Pr\left[K \xleftarrow{\$} \mathcal{K}eys : \mathcal{A}^{\Pi(K,\cdot)} \Rightarrow 1\right] - \Pr\left[\pi \xleftarrow{\$} \mathfrak{S} : \mathcal{A}^{\pi} \Rightarrow 1\right]\right|.$$

Ensemble et distribution sans préfixe

Avant de présenter la deuxième primitive cryptographique, nous donnons d'abord quelques définitions utiles.

Soient x et x' deux chaînes de bits. On dit que x est un *préfixe* de x' s'il existe une chaine de bits x'', éventuellement nulle, telle que $x' = x\|x''$ où $\|$ désigne la concaténation. On notera par $x \sqsubset x'$ le fait que x soit un préfixe de x'.

Soit $\mathcal{S}$ un ensemble de chaînes de bits. On dit que $\mathcal{S}$ est sans préfixe si pour tout couple $(x,x') \in S^2$, $x \sqsubset x'$ implique que $x = x'$.

Une distribution $\mathcal{D}_2$ sur $\mathcal{S} \times \mathcal{S}$ est dite sans préfixe si pour tous les couples $(x,x') \in \mathcal{S}^2$ tels que $\Pr\left[(X,X') \xleftarrow{\mathcal{D}_2} \mathcal{S} \times \mathcal{S} : (X,X') = (x,x')\right] > 0$, $x \sqsubset x'$ implique que $x = x'$.

Une distribution $\mathcal{D}$ sur $\mathcal{S}$ est dite sans préfixe si la distribution produit $\mathcal{D}_\pi = \mathcal{D} \times \mathcal{D}$ sur $\mathcal{S} \times \mathcal{S}$ est sans préfixe. Ceci peut également être reformulé de la façon suivante : la distribution $\mathcal{D}$ sur $\mathcal{S}$ est dite sans préfixe si pour tous les couples $(x, x') \in \mathcal{S}^2$ tels que $\Pr\left[X \xleftarrow{\mathcal{D}} \mathcal{S} \colon X = x\right] > 0$ et $\Pr\left[X \xleftarrow{\mathcal{D}} \mathcal{S} \colon X = x'\right] > 0$, $x \sqsubset x'$ implique que $x = x'$.

Enfin un adversaire $\mathcal{A}$ est dit sans préfixe si l'ensemble formé par toutes ses requêtes forme un ensemble sans préfixe et s'il envoie uniquement des distributions sans préfixe.

Fonctions pseudo-aléatoires

Nous notons par $\mathcal{F} = \mathcal{F}_{(\mathcal{D}om,\mathcal{R}ng)}$ l'ensemble des fonctions de $\mathcal{D}om$ dans $\mathcal{R}ng$. Soit $F : \mathcal{K}eys \times \mathcal{D}om \to \mathcal{R}ng$ une famille de fonctions, on dit que c'est une famille de fonctions pseudo-aléatoires si l'avantage d'un adversaire dans le jeu suivant est faible. Avant le début du jeu, le challenger choisit un bit b au hasard et, si $b = 1$ il choisit f au hasard dans $\mathcal{F}$, sinon il choisit une clef K au hasard et affecte $f = F(K, \cdot)$). Ce que l'on appelle l'adversaire PRF $\mathcal{A}$ (pour *Pseudo-Random Function*) peut alors interagir avec f en posant au plus q requêtes x_i dans $\mathcal{D}om$, $1 \leq i \leq q$, et reçoit en échange $f(x_i)$. Enfin l'adversaire renvoie un bit b' et gagne si $b = b'$. L'avantage en PRF de cet attaquant, noté $\mathsf{adv}_F^{\mathtt{prf}}(\mathcal{A})$, vaut :

$$\mathsf{adv}_F^{\mathtt{prf}}(\mathcal{A}) = \left|\Pr\left[K \xleftarrow{\$} \mathcal{K}eys \colon \mathcal{A}^{F(K,\cdot)} \Rightarrow 1\right] - \Pr\left[f \xleftarrow{\$} \mathcal{F} \colon \mathcal{A}^{f} \Rightarrow 1\right]\right|.$$

Dans le cas où l'adversaire $\mathcal{A}$ est contraint à être sans préfixe (c'est-à-dire que l'ensemble des requêtes formulées par $\mathcal{A}$ est un ensemble sans préfixe) alors on dit que $\mathcal{A}$ est un adversaire PRF sans préfixe ou adversaire *pf-PRF* et on note $\mathsf{adv}_F^{\mathtt{pf-prf}}(\mathcal{A})$ l'avantage de $\mathcal{A}$ dans ce cas.

2.4 Modèles de l'oracle aléatoire et du chiffrement idéal

Les deux modèles que nous présentons maintenant sont considérées moins réalistes car ils font appels à des oracles qui n'existent pas en pratique. Ces oracles modélisent certaines primitives cryptographiques, mais cette modélisation est très idéale. On lui reproche parfois d'être trop loin de la réalité. Cependant cette modélisation donne parfois une bonne approximation de la raison pour laquelle, en pratique, un adversaire ne pourra pas casser la propriété de sécurité. Une preuve dans ces modèles donne donc principalement des arguments pour avoir confiance en un protocole. Il faut considérer ces modèles avec certaines précautions et restrictions d'usage, mais, même si une preuve dans ces modèles n'a pas la même portée qu'une preuve dans le modèle standard, il vaut mieux un protocole qui a une preuve dans un de ces modèles qu'un protocole qui n'a pas de preuve du tout. Une preuve dans ces modèles est toujours une bonne chose.

2.4.1 Modèle du chiffrement idéal

Pour modéliser les algorithmes de chiffrement par bloc, on utilise souvent le modèle du chiffrement idéal, modèle qui a été introduit par Shannon. Dans ce modèle, l'adversaire n'est pas calculatoirement limité et l'algorithme de chiffrement par bloc est vu comme une famille de fonctions $E \colon \{0,1\}^\kappa \times \{0,1\}^n \to \{0,1\}^n$ telle que pour tout k, $E(k, \cdot)$ est une permutation sur $\{0,1\}^n$. Pour toute clef k, $E(k, \cdot)$ est choisie aléatoirement, en suivant la distribution uniforme, dans l'ensemble des permutations sur n bits. Cela implique, pour tout adversaire et pour toute clef k, $k \in \{0,1\}^\kappa$, $E(k, \cdot)$ est une permutation aléatoire et indépendante des autres permutations de la famille.

Dans ce modèle, l'adversaire a accès à deux oracles E et E^{-1}, ce que l'on note $\mathcal{A}^{E,E^{-1}}$, et il peut poser au plus Q requêtes aux oracles. À chacune des requêtes (K_i, X_i) qu'il fait à E il reçoit en réponse $Y_i = E(K_i, X_i)$ et à chacune des requêtes (K_i, Y_i) qu'il fait à E^{-1} il reçoit en réponse $X_i = E^{-1}(K_i, Y_i)$.

2.4.2 Modèle de l'oracle aléatoire

Le modèle de l'oracle aléatoire a été introduit par Bellare et Rogaway [BR93b] en 1993 pour modéliser des fonctions de hachage. Dans ce modèle, le challenger et l'adversaire peuvent faire des requêtes à un *oracle aléatoire* noté O, lequel répond en choisissant une valeur aléatoire, avec pour seule contrainte que si on lui pose deux fois la même requête, alors il doit faire deux fois la même réponse. Plus précisément, le modèle fixe un entier n et un domaine d'entrée $\mathcal{D}om$ pour l'oracle (dans ce mémoire le plus souvent $\mathcal{D}om = \{0,1\}^*$ ou bien $\mathcal{D}om = \{0,1\}^n$). Au début du jeu de sécurité, l'oracle choisit au hasard une fonction f de $\mathcal{D}om$ dans $\{0,1\}^n$ et à toutes les requêtes $x \in \mathcal{D}om$ faites par une partie, il répond par $f(x)$.

2.5 Fonctions de hachage

Avant de finir cette partie introductive sur les notions utiles dans les preuves de sécurité, nous présentons une dernière primitive cryptographique très utilisée en pratique : les fonctions de hachage. Le but premier des fonctions de hachage est de pouvoir donner pour n'importe quel message de taille arbitraire un condensé de taille fixe. Elles remplissent des fonctions très diverses dans les protocoles et nous sommes amenés à les considérer à plusieurs reprises dans ce mémoire. Pour cette raison, nous en donnons une description commune dans ce chapitre.

En cryptographie, on exige d'une bonne fonction de hachage qu'elle assure trois propriétés : la résistance aux collisions, la résistance en seconde préimage et la résistance en préimage. Nous allons donner la définition de ces trois propriétés mais d'abord fixons quelques notations.

Dans cette section, nous désignons par $O_1, \ldots, O_n$ n oracles. Dans ce mémoire, selon que l'on se place dans le modèle du chiffrement idéal ou dans le modèle de l'oracle aléatoire, ces oracles choisirons ou bien une permutations aléatoire ou bien une fonction aléatoire : nous préciserons au cas par cas la définition exacte de ces oracles. Nous notons par $H^{O_1,\ldots,O_n} : \{0,1\}^* \to \{0,1\}^n$ une fonction de hachage paramétrée par ces oracles : cela signifie que l'évaluation de la fonction de hachage est dépendante des fonctions ou permutations aléatoires choisies par les oracles. Puisque les oracles choisissent de nouvelles fonctions ou permutations aléatoires au début de chaque jeu de sécurité, un adversaire contre la fonction de hachage ne peut connaître à l'avance les valeurs prises par H en certains points. Il est donc contraint d'interroger les oracles au cours du jeu pour déterminer ces valeurs. Nous autorisons l'adversaire à faire au plus Q requêtes à l'ensemble de ces oracles.

Enfin, puisque la définition des oracles dont dépend H sera toujours très claire, nous simplifions les notations en écrivant H au lieu de $H^{O_1,\ldots,O_n}$.

2.5.1 Résistance aux collisions

Le but d'un adversaire en collision contre la fonction de hachage H est de trouver deux messages différents M et M' tels que $H(M) = H(M')$. La probabilité que l'adversaire réussisse à casser la résistance aux collisions de H est notée $\mathsf{adv}_H^{\mathsf{Coll}}(\mathcal{A})$ et est égale à :

$$\mathsf{adv}_H^{\mathsf{Coll}}(\mathcal{A}) \stackrel{def}{=} \Pr\left[\mathcal{A}^{O_1,\ldots,O_n} \Rightarrow (M, M') \colon H(M) = H(M') \wedge M \neq M'\right].$$

Cette probabilité est prise sur l'ensemble des aléas de l'adversaire $\mathcal{A}$ et sur celui des oracles. Nous utiliserons la notation $\mathsf{adv}_H^{\mathsf{Coll}}(Q)$ pour désigner le maximum des probabilités de succès $\mathsf{adv}_H^{\mathsf{Coll}}(\mathcal{A})$, où ce maximum est pris sur tous les adversaires $\mathcal{A}$ qui font au plus Q requêtes.

Il existe une attaque générique contre la résistance en collision d'une fonction de hachage. On appelle cette attaque l'attaque en paradoxe des anniversaires : choisir $(M_1, \ldots, M_{2^{n/2}})$ $2^{n/2}$ messages deux à deux distincts et calculer les $2^{n/2}$ hachés de ces messages en faisant les requêtes correspondantes aux oracles. Le paradoxe des anniversaires dit que dans ce cas, avec grande probabilité, il existe $i \neq j$ tels que $H(M_i) = H(M_j)$. On peut donc trouver une collision en $2^{n/2}$ calculs de la fonction de hachage. Comme cette attaque est générique, la seule chose que l'on peut espérer, c'est qu'il n'existe pas d'attaque plus efficace et c'est le plus souvent ce que l'on cherche à éviter lorsque l'on construit une fonction de hachage. La meilleure attaque possible doit être celle par paradoxe des anniversaires.

2.5.2 Résistance en seconde préimage

Pour cette notion, il est nécessaire de restreindre l'ensemble de définition de H, afin de pouvoir définir facilement une distribution de probabilité uniforme sur l'ensemble des messages. Soit r un entier naturel, on considère donc la fonction de hachage $H\colon \{0,1\}^r \to \{0,1\}^n$. L'objectif d'un adversaire en seconde préimage contre la fonction de hachage H est, étant donné un message M, de trouver un message M', différent de M, tel que $H(M) = H(M')$. Le message M est choisi par le challenger selon la distribution uniforme sur $\{0,1\}^r$. La probabilité que l'adversaire réussisse à casser la résistance en seconde préimage de H est notée $\mathsf{adv}_H^{\mathsf{sec}}(\mathcal{A})$ et est égale à :

$$\mathsf{adv}_H^{\mathsf{sec}}(\mathcal{A}) \stackrel{def}{=} \Pr\left[M \xleftarrow{\$} \{0,1\}^r, \mathcal{A}^{O_1,\ldots,O_n}(M) \Rightarrow M' \colon H(M) = H(M') \wedge M \neq M'\right].$$

Cette probabilité est prise sur l'ensemble des aléas de l'adversaire $\mathcal{A}$, sur celui des oracles et sur le choix du message M. Nous utiliserons la notation $\mathsf{adv}_H^{\mathsf{sec}}(Q)$ pour désigner le maximum des probabilités de succès $\mathsf{adv}_H^{\mathsf{sec}}(\mathcal{A})$, où ce maximum est pris sur tous les adversaires $\mathcal{A}$ qui font au plus Q requêtes.

Il est facile d'utiliser un adversaire en seconde préimage pour construire un adversaire contre la résistance aux collisions : on choisit M au hasard, on lance l'adversaire en seconde préimage, on lui donne M, il nous renvoie M' et il suffit alors de renvoyer (M, M') qui, avec forte probabilité, induit bien une collision. La résistance aux collisions implique donc la résistance en seconde préimage. Cependant, on ne cherche pas le même niveau de sécurité pour les deux propriétés : on souhaite le plus souvent qu'il soit beaucoup plus difficile de trouver une seconde préimage que de trouver une collision. Un adversaire ne doit pas trouver de collision tant qu'il n'a pas fait plus de $2^{n/2}$ calculs de la fonction de hachage et il ne doit pas trouver de seconde préimage tant qu'il n'a pas fait plus de 2^n calculs de la fonction de hachage.

2.5.3 Résistance en préimage

Pour cette notion encore, il est nécessaire de restreindre l'ensemble de définition de H, afin de pouvoir définir facilement une distribution de probabilité uniforme sur l'ensemble des messages. Soit r un entier naturel, on considère donc la fonction de hachage $H\colon \{0,1\}^r \to \{0,1\}^n$. L'objectif d'un adversaire en préimage contre la fonction de hachage H est, étant donné une valeur $Y \in \{0,1\}^n$, de trouver un message M tel que $H(M) = Y$. La valeur Y est choisie par le challenger de la manière suivante : il choisit un message M' parfaitement au hasard dans $\{0,1\}^r$ puis définit $Y = H(M')$. La probabilité que l'adversaire réussisse à casser la résistance en préimage de H est

notée $\mathsf{adv}_H^{\mathsf{pre}}(\mathcal{A})$ et est égale à :

$$\mathsf{adv}_H^{\mathsf{pre}}(\mathcal{A}) \stackrel{def}{=} \Pr\left[M' \stackrel{\$}{\leftarrow} \{0,1\}^r, Y = H(M'), \mathcal{A}^{O_1,\ldots,O_n}(Y) \Rightarrow M : H(M) = Y\right].$$

Cette probabilité est prise sur l'ensemble des aléas de l'adversaire $\mathcal{A}$, sur celui des oracles et sur le choix de M'. Nous utiliserons la notation $\mathsf{adv}_H^{\mathsf{pre}}(Q)$ pour désigner le maximum des probabilités de succès $\mathsf{adv}_H^{\mathsf{pre}}(\mathcal{A})$, où ce maximum est pris sur tous les adversaires $\mathcal{A}$ qui font au plus Q requêtes.

2.5.4 Mode cascade

Le mode cascade a été introduit indépendamment par Merkle [Mer89] et Damgård [Dam89] en 1989, aussi on l'appelle souvent *construction de Merkle-Damgård* . On le trouve aussi parfois dans la littérature sous le nom de mode chaîné ou de mode itéré. Il est le mode opératoire le plus répandu pour construire des fonctions de hachage : MD4, MD5, SHA–1 et SHA–2, pour ne citer que les plus connues, sont basées sur ce mode. Son importance pratique en fait un mode incontournable et nous l'étudions ou l'utilisons à plusieurs reprises au cours de ce mémoire, c'est pourquoi nous en donnons ici la description.

C'est un mode opératoire qui permet de construire une fonction de hachage acceptant des messages de taille arbitraire à partir d'une fonction de hachage acceptant uniquement des messages de taille fixe. Pour cela, on itére la fonction de hachage de taille fixe, aussi appelée *fonction de compression*, de telle sorte qu'à chaque appel de cette fonction, on traite une partie du long message donné en entrée. Pour cette raison, la fonction de hachage résultante est également nommée *fonction de hachage itérée*.

En voici la description précise. Soient $h : \{0,1\}^n \times \{0,1\}^b \to \{0,1\}^n$ la fonction de compression et IV une chaîne de bits de $\{0,1\}^n$ que l'on désignera aussi sous le nom de *valeur initiale*. La construction en cascade de h est la fonction $h^* : \{0,1\}^n \times \left(\{0,1\}^b\right)^* \to \{0,1\}^n$, définie par :

$$y_0 = IV,\ y_i = h(y_{i-1}, x_i) \text{ et } h^*(IV, x) = y_m$$

où $x = (x_1, \ldots, x_m)$ est une chaîne de bits de taille $m \cdot b$. Les valeurs y_i, $0 \leq i \leq m$, sont appelées les *valeurs de chaînage intermédiaires*.

La fonction h^* n'accepte que des messages dont la taille est un multiple de b. Afin de pouvoir traiter un message x de taille totalement arbitraire, on définit une manière de compléter ce message, ou *padding*, pour obtenir un message dont la taille est un multiple exact de b. En pratique cette manière de compléter les messages est une fonction de la taille $|x|$ en bits du message. Nous noterons dans ce mémoire $\mathrm{pad}(|x|)$ la fonction induite par ce *padding* et nous utiliserons la notation $x_{pad} = x \| \mathrm{pad}(|x|)$.

On définit donc la fonction de hachage $H : \{0,1\}^n \times \{0,1\}^* \to \{0,1\}^n$ par

$$H(IV, x) = h^*\left(IV, x_{pad}\right).$$

Si l'on souhaite mettre en avant la valeur initiale utilisée, on écrira $H(IV, x)$, sinon on écrira $H(x)$ et on considèrera H comme une fonction de $\{0,1\}^*$ dans $\{0,1\}^n$.

Le *padding* classique consiste à ajouter un unique '1' suivi par autant de '0' que nécessaire pour obtenir un message dont la taille en bits est un multiple la taille du bloc, b. Merkle [Mer89] et Damgård [Dam89] ont prouvé indépendamment en 1989 qu'intégrer la taille du message dans le *padding* augmente la sécurité de la construction : avec ce qui est maintenant connu comme le *renforcement* de Merkle-Damgård, ils ont prouvé que le schéma permet de conserver la résistance

aux collisions de la fonction de compression, dans le sens où une collision sur la fonction de hachage H implique une collision sur la fonction de compression h. En d'autres termes, il est au moins aussi difficile de trouver une collision sur la fonction de hachage que sur la fonction de compression. Précisément, la fonction de *padding* induite est :

$$\mathrm{pad}(|x|) = 10^s \| \, |x| \text{ avec } s = b - 1 - t \mod b,$$

où t désigne le nombre de bits sur lequel on encode la taille de x. Ce nombre doit être précisé dans les spécifications de la fonction de hachage. À cause de ce nombre t, la taille maximale possible des messages qui peuvent être hachés est limitée à $2^t - 1$ bits. Dans la plupart des fonctions de hachage utilisées en pratique, la taille maximale possible des message est de 2^{64} bits, soit 2^{55} blocs de 512 bits.

Deuxième partie

Création d'un canal sécurisé et authentifié

3

Authentification multi-facteurs et génération de clefs

Sommaire

3.1 Introduction

3.1.1 Facteurs d'authentification

Comme expliqué en introduction dans la section 1.2, lorsque que l'on veut établir un canal qui assure l'authentification des messages échangés, la notion d'authentification est indissociable des deux notions d'intégrité et de confidentialité. Aussi, pour nous intéresser à certains mécanismes cryptographiques assurant l'authentification, nous devons nous intéresser aussi à certains moyens de garantir l'intégrité et la confidentialité. Pour cette raison, nous étudions les mécanismes conduisant à l'établissement d'un canal authentifié, confidentiel et intègre, aussi appelé simplement canal sécurisé.

Cependant avant d'entamer cette étude, faisons remarquer qu'en toute rigueur, nous n'aurions besoin de ne nous intéresser qu'à la confidentialité des clefs utilisées par les algorithmes cryptographiques et non à la confidentialité du canal en entier. Cependant, en pratique les canaux que l'on crée ne sont pas uniquement authentifiés et intègres, il sont aussi tous confidentiels. Aussi, même si notre étude s'intéresse à l'authentification, elle conservera une portée générale et examinera la création d'un canal authentifié, confidentiel et intègre.

On rappelle que la création d'un canal sécurisé se décompose en deux étapes essentielles : tout d'abord l'échange de clefs authentifié puis, grâce à ces clefs, la création du canal proprement dit par des méthodes symétriques. C'est à la première étape que nous allons nous intéresser dans ce chapitre.

Puisqu'aucun canal sécurisé n'existe à l'origine, l'échange de clefs authentifié doit se faire sur un canal public supposé non sûr. Pour mener à bien cet échange de clefs, il faut réussir à vérifier l'identité de l'entité avec laquelle on communique et ce grâce à une caractéristique qui lui est propre, c'est-à-dire une preuve d'identité que seule cette entité peut fournir : c'est ce que l'on appelle un *moyen d'authentification*. Il existe plusieurs moyens d'authentification qui définissent une personne de façon unique et notamment : une information secrète connue d'elle seule, un objet que seule cette personne possède ou bien une mesure d'un trait biométrique de cette personne. Ces exemples représentent les trois classes de facteurs d'authentification humaine admises généralement, à savoir :

- quelque chose que vous *savez* (tel qu'un mot de passe par exemple),
- quelque chose que vous *possédez* (tel qu'un dispositif sûr et inclonable sur lequel est écrit une clef privée),
- ce que vous *êtes* (tel qu'une mesure biométrique par exemple).

Brainard *et al.* [BJR+06] ont récemment proposé un quatrième moyen d'authentification : *quelqu'un que vous connaissez*, aussi appelé le réseau social. Cependant par la suite nous ne nous focaliserons que sur les trois facteurs d'authentification classiques décrits ci-dessus.

Aucun de ces facteurs d'authentification n'est parfait et tous présentent des inconvénients et des faiblesses. Ainsi, ils prêtent tous trois le flan à diverses attaques, notamment des attaques qui ne peuvent être empêchées par des méthodes cryptographiques et qui nécessitent des protections extérieures :

- le mot de passe peut être récupéré par de « l'ingénierie sociale » (par exemple par de l'hameçonnage ou *phishing* [JM06] ou par l'utilisation d'un logiciel malveillant), les utilisateurs doivent donc être prudents quand ils entrent leur mot de passe et ne doivent pas le transmettre ;
- le dispositif sûr peut être volé, ouvert ou cloné, c'est pourquoi il doit notamment être protégé afin de le rendre inviolable et inclonable ;
- des données biométriques peuvent être récupérées et copiées, le capteur doit donc être capable de détecter si une donnée biométrique est une vraie donnée biométrique, correspondant effectivement à la personne qui est en train d'être contrôlée, ou s'il s'agit d'une imitation.

Ces attaques rendent possibles des usurpations d'identités qui ne peuvent pas être détectées par des moyens cryptographiques. Aussi, afin de rendre plus difficile la tâche d'un attaquant, mais également afin de limiter le risque d'erreur, on peut combiner deux ou trois facteurs lors de la même authentification. En effet, on peut espérer que combiner les trois facteurs d'authentification dans le même protocole force l'adversaire à devoir casser les trois facteurs en même temps pour réussir à s'authentifier. Cependant, toute combinaison des trois facteurs n'atteint pas nécessairement l'objectif fixé, on peut imaginer des protocoles dans lesquels il suffit de casser un seul facteur d'authentification pour casser l'ensemble du protocole. Aussi, la conception d'un

protocole doit être validée par une analyse de sécurité rigoureuse. Pour cela, nous développons dans ce chapitre un modèle de sécurité qui prend en compte les spécificités de chacun des facteurs d'authentification. Notons que ce chapitre a donné lieu à une publication [PZ08].

3.1.2 Clefs asymétriques

Sur un dispositif, que l'on espère inviolable et inclonable, il est possible de mettre un ensemble composé d'une clef privée et d'une clef publique. Les spécificités des échanges de clefs à base d'un couple clef publique/clef privée sont bien connues [BR93a, CK01, CK02]. Ce sont les facteurs d'authentification les plus simples à utiliser car on peut utiliser des signatures [GMR88] pour authentifier les échanges ou bien d'autres primitives asymétriques [GM84, RS92].

La clef privée $\mathsf{sk}_{\mathcal{C}}$ est choisie uniformément au hasard dans un ensemble de clefs privées $\mathcal{K}eys$, où $\mathcal{K}eys$ est supposé être très grand (c'est-à-dire que la clef a une grande entropie), de telle sorte que $1/|\mathcal{K}eys|$ est négligeable. Elle est enregistrée sur un dispositif sécurisé. Grâce au protocole et à la clef publique correspondant à cette clef privée, on pourra vérifier que le client possède bien cette clef privée ce qui garantira que le client *possède* bien ce dispositif.

3.1.3 Mot de passe

Au contraire du couple clef publique/clef privée, le mot de passe $\mathsf{pwd}_{\mathcal{C}}$ est choisi dans un dictionnaire à faible entropie $Dict \subset \mathbb{Z}_p^\star$, selon la distribution de probabilité $\mathcal{D}_{\mathsf{pwd}}$. Nous noterons par $\mathcal{D}_{\mathsf{pwd}}(q)$ la somme des probabilités des q éléments les plus probables selon la distribution $\mathcal{D}_{\mathsf{pwd}}$. Grâce au protocole on pourra vérifier que le client possède bien le mot de passe, ce qui garantira que le client le *connaît*.

Puisque le mot de passe doit être mémorisable, il possède une faible entropie et la recherche exhaustive, qui est prohibitivement coûteuse pour une clef asymétrique car elle a une forte min-entropie, devient réaliste pour le mot de passe. Il est notamment possible de mener ce qu'on appelle des *attaques par dictionnaire en ligne*. Celles-ci consistent à essayer d'usurper l'identité de quelqu'un en tentant de s'authentifier auprès du serveur grâce au mot de passe qui paraît le plus probable. Suite à cette tentative, ou bien l'adversaire réussit à s'authentifier, ou bien il déduit de son échec que le mot de passe qu'il a essayé n'est pas le bon et peut donc l'enlever de la liste des mots de passe possibles. S'il avait échoué, l'adversaire recommence avec un nouveau mot de passe et ce, jusqu'à ce qu'il réussisse à s'authentifier. Cette attaque est générique, indépendante du protocole et pour cette raison, inévitable. Aussi, lorsque l'on crée un protocole d'authentification par mot de passe, on souhaite que cette attaque par dictionnaire en ligne soit la meilleure attaque existante. On veut notamment que le protocole soit résistant aux *attaques par dictionnaire hors ligne*. Dans une attaque par dictionnaire hors ligne, l'attaquant dans un premier temps accumule un peu d'information par le biais d'attaques actives ou passives, puis dans un second temps est capable de façon efficace de restreindre très fortement l'ensemble des mots de passe possibles. Pour résumé, on veut qu'un adversaire passif n'apprenne rien du mot de passe utilisé et on veut qu'un adversaire actif, à chaque tentative d'authentification, ne soit capable que d'oter un et un seul mot de passe à la liste des mots de passe possibles. Au bout de q tentatives d'authentification, la probabilité de succès de l'attaquant sera donc plus au plus de $\mathcal{D}_{\mathsf{pwd}}(q)$.

3.1.4 Traits biométriques

L'utilisation de l'authentification à base de traits biométriques est plus récente et soulève des problèmes différents et moins bien connus, c'est pourquoi nous la détaillons plus. Elle soulèvent notamment trois problèmes majeurs.

Empreintes Biométriques. Tout d'abord, les traits biométriques posent un défi technique au cryptographe car deux mesures du même trait biométrique d'un individu ne sont pas exactement identiques, même si elles se ressemblent fortement. De ce fait, pour déterminer si deux mesures d'un trait biométrique proviennent du même individu ou de deux personnes différentes, il faut mettre en place un mécanisme dit « de correspondance », c'est-à-dire un mécanisme qui les compare et apprécie à quel point ces deux mesures sont similaires. Le mécanisme de correspondance peut par exemple faire appel à une fonction de distance et décréter que si la distance entre les deux mesures biométriques est plus petite qu'un certain seuil, alors ces deux mesures proviennent du même individu. C'est ce qui est fait dans le cas de l'iris [Dau02], avec comme caractéristique supplémentaire que la distance considérée pour l'iris est la distance de Hamming. Cependant, aucun mécanisme de correspondance n'est infaillible, ils sont tous susceptibles de commettre deux erreurs possibles qui sont la *fausse acceptation* (quand le système accepte un individu alors qu'il ne devrait pas) et le *faux rejet* (quand le système ne reconnaît pas quelqu'un qu'il devrait reconnaître). C'est pourquoi un protocole d'échange de clefs authentifié basé sur la biométrie doit tenir compte des erreurs de mesure et rendre l'exécution de ce mécanisme de correspondance possible, mais ne doit pas augmenter les taux de faux rejet et de fausse acceptance de façon significative.

Afin de modéliser cet aléa dans la mesure du trait biométrique d'un individu on introduit des distributions de probabilités sur l'ensemble des mesures possibles. Ainsi, pour chaque client $\mathcal{C}$, $\mathcal{D}_{\mathcal{C}}$ définit la distribution de probabilité du trait biométrique (empreinte digitale, forme du visage, iris,*etc.*) pour ce client. Par la suite, les variables aléatoires $W_{\mathcal{C}}$ et $W'_{\mathcal{C}}$ désigneront deux mesures du trait biométrique du client $\mathcal{C}$. Ces variables aléatoires sont indépendantes et suivent la distribution $\mathcal{D}_{\mathcal{C}}$. Afin que la modélisation de la biométrie soit pertinente pour l'authentification, nous devons faire quelques hypothèses sur le mécanisme de correspondance, et plus précisément à propos de la distance de Hamming, puisque nous allons utiliser cette distance pour décider si deux mesures biométriques viennent du même individu :

- d'une part, la distance séparant deux mesures $W_{\mathcal{C}}$ et $W'_{\mathcal{C}}$ du trait biométrique du même client est faible avec bonne probabilité. Plus précisément, il existe un seuil t, tel que pour tout client $\mathcal{C}$,
$$\Pr\left[W_{\mathcal{C}} \leftarrow \mathcal{D}_{\mathcal{C}}, W'_{\mathcal{C}} \leftarrow \mathcal{D}_{\mathcal{C}} : d_H(W_{\mathcal{C}}, W'_{\mathcal{C}}) \leq t\right] \geq 1 - \varepsilon_{\mathsf{fr}}.$$
L'indice fr indique les « faux rejets ».
- d'autre part, pour toute paire de clients distincts $\mathcal{C} \neq \mathcal{C}'$, la distance entre $W_{\mathcal{C}}$ et $W_{\mathcal{C}'}$ est grande avec bonne probabilité. Plus précisément, il existe un seuil $\tau \geq t$, tel que pour tout $\mathcal{C} \neq \mathcal{C}'$,
$$\Pr\left[W_{\mathcal{C}} \leftarrow \mathcal{D}_{\mathcal{C}}, W_{\mathcal{C}'} \leftarrow \mathcal{D}_{\mathcal{C}'} : d_H(W_{\mathcal{C}}, W_{\mathcal{C}'}) > \tau\right] \geq 1 - \varepsilon_{\mathsf{fa}}.$$
L'indice fa indique les « fausses acceptations ».

Hypothèse de présence. Une caractéristique principale des données biométriques est qu'elles ne peuvent être supposées secrètes que dans des contextes où on ne cherche pas à assurer un haut niveau de sécurité et c'est ce qui constitue le deuxième problème majeur des données biométriques. En effet, on peut très facilement récupérer une empreinte digitale sur un objet (comme un verre par exemple) que quelqu'un a touché ; de même il est possible d'obtenir l'image d'un iris en prenant une photo de bonne qualité. Il n'est donc pas raisonnable de considérer la donnée biométrique comme une information parfaitement confidentielle et la traiter comme s'il s'agissait d'une clef secrète, même si cela a déjà été fait dans la littérature [JW99, DRS04, Boy04, BDK+05, DKRS06]. Ainsi les célèbres *secure sketches* et *fuzzy extractors* introduits par Dodis *et*

al. [DRS04] ne sont d'aucune utilité ici car ils nécessitent justement que la donnée biométrique reste secrète.

Mais alors, si les données biométriques sont publiques, comment peut-on empêcher un adversaire d'usurper l'identité d'un utilisateur honnête ? La seule méthode pour utiliser les données biométriques pour l'authentification est de garantir que la mesure est faite sur un être humain réel, présent en chaire et en os lors du contrôle, et non d'une contrefaçon. Pour cela plusieurs solutions techniques ont été élaborées : utilisation de canaux authentifiés entre le capteur et le serveur d'authentification, vérification de plusieurs données biométriques en même temps, surveillance de l'authentification et contrôle de l'intégrité du capteur par un être humain de confiance, *etc.* Toutes ces solutions permettent de supposer comme vraie l'hypothèse selon laquelle la donnée biométrique mesurée provient d'un être humain physiquement présent et selon laquelle on peut faire confiance au capteur (il n'a pas été altéré par un adversaire). Cette dernière hypothèse est appelée « hypothèse de présence » [Val02, Dau04]. Cette hypothèse est une hypothèse forte notamment car elle implique que la donnée biométrique est fraîche (et non issue d'un rejeu), qu'elle vient d'une personne vivante (et non de la contrefaçon d'un trait biométrique), et que les calculs effectués à partir de cette donnée biométrique l'ont été honnêtement. Cependant même si cette hypothèse est forte, les propriétés de la biométrie la rendent absolument indispensable pour assurer la sécurité de l'authentification. Sans elle, aucune authentification par biométrie n'est possible.

Pour modéliser cette hypothèse, on utilise un oracle de calcul $\mathsf{Compute}(\Pi^i_{\mathcal{C}}, W', aux)$, où $\mathcal{C}$ désigne un client (désormais les entités intervenant dans le protocole seront ou bien des clients ou bien des serveurs) : selon l'état du client, à partir d'entrées additionnelles aux, comme par exemple d'autres facteurs d'authentification, et d'une nouvelle acquisition W' de la donnée biométrique, l'oracle calcule honnêtement le message qui aurait été généré par $\mathcal{C}$. Puisque ceci modélise la tentative d'un attaquant pour s'authentifier en utilisant sa propre donnée biométrique, W' doit être choisie selon une (mauvaise) distribution $\mathcal{D}$ pour laquelle on a $\Pr[W' \leftarrow \mathcal{D}, W_{\mathcal{C}} \leftarrow \mathcal{D}_{\mathcal{C}} : d_H(W', W_{\mathcal{C}}) > \tau] \geq 1 - \varepsilon_{\mathsf{fa}}$. Avec l'hypothèse de présence pour le client $\mathcal{C}$, on considère que tous les messages faisant intervenir la donnée biométrique et venant prétendument de $\mathcal{C}$, ont été générés par l'oracle de calcul.

Grâce à l'hypothèse de présence, même si nous supposons que pour tout client $\mathcal{C}$, la distribution de données biométriques $\mathcal{D}_{\mathcal{C}}$ est publique, l'acceptation d'une empreinte biométrique garantit tout de même que le client semble *être* le client attendu.

Anonymat des bases de données. Enfin, en pratique les mesures biométriques sont conservées dans des bases de données dans lesquelles elles sont souvent liées à d'autres informations personnelles concernant l'utilisateur. Puisque par ailleurs, une mesure biométrique peut être utilisée pour identitifier un individu de façon univoque, les mesures biométriques présentes dans la base de données sont particulièrement sensibles. Or les bases de données sont souvent vulnérables à des attaques menées par des adversaires externes ou internes, aussi pour préserver l'anonymat des personnes présentes on exige souvent que la mesure biométrique soit chiffrée.

3.2 Modèle de sécurité d'un échange de clefs authentifié classique

Dans cette partie, nous décrivons le modèle de sécurité pour l'échange de clefs authentifié (noté AKE dans la suite, pour *Authenticated Key Exchange*). Ce modèle est construit à partir du modèle de sécurité classique pour l'échange de clefs basé sur le mot de passe [BR93a, BPR00], dans le modèle de l'indistinguabilité réel-ou-aléatoire [BDJR97, AFP05].

3.2.1 Notions de sécurité

Sécurité sémantique. La sécurité sémantique de la clef est modélisée en utilisant le modèle réel-ou-aléatoire [BDJR97, AFP05], désigné dans toute la suite par RoR, pour *Real or Random*. Au début du jeu, le challenger choisit uniformément un bit b, bit qui détermine les réponses que le challenger apportera à ce qu'on appelle les requêtes-Test lors du jeu : si $b = 1$ il répondra à l'adversaire toujours par les clefs réelles, si $b = 0$ il répondra à l'adversaire toujours par des clefs aléatoires uniformes (voir la description des requêtes-Test à la sous-section suivante pour plus de détails). L'adversaire peut interagir avec les instances du protocole à travers plusieurs oracles décrits ci-dessous et à la fin du jeu il doit renvoyer un bit b'. Si $b = b'$ alors l'adversaire est déclaré gagnant, dans le cas contraire il est déclaré perdant.

On note par Succ l'événement $b = b'$. L'**avantage-ake** $\mathsf{adv}_P^{\mathtt{ake}}(\mathcal{A})$ et la fonction avantage du protocole P sont respectivement :

$$\mathsf{adv}_P^{\mathtt{ake}}(\mathcal{A}) = 2 \cdot \Pr[\mathsf{Succ}] - 1, \quad \mathsf{adv}_P^{\mathtt{ake}}(t, Q) = \max_{\mathcal{A}} \left\{ \mathsf{adv}_P^{\mathtt{ake}}(\mathcal{A}) \right\},$$

où le maximum est pris sur tous les attaquants ayant une complexité temporelle plus petite que t et un nombre de requêtes au plus égal à Q.

Sécurité dans le temps. Le protocole est sûr dans le temps si on a la propriété suivante : si la clef de session est sémantique sûre au moment de sa génération alors elle le restera toujours, même si la clef à long terme est corrompue par la suite. Afin de garantir cette notion de sécurité, notre modèle doit autoriser l'adversaire à effectuer des requêtes-Test, même quand une requête-$\mathsf{CorruptKey}$ à été faite, tant que la session a été complétée avant la corruption du client.

Authentification. Le protocole garantit l'authentification si la propriété suivante est vraie : une partie n'accepte que si le protocole a été effectué avec une vraie partie et non une instance de l'adversaire. Un protocole peut garantir une authentification mutuelle (à la fin du protocole, un client est sûr que c'est avec le serveur qu'il interagit et réciproquement le serveur est sûr que c'est avec le client qu'il interagit) ou une authentification unilatérale (par exemple le serveur est sûr que c'est avec le client qu'il interagit, par contre le client ne pourra jamais être sûr que c'est bien avec le serveur qu'il interagit). Dans notre protocole nous prouvons uniquement *l'authentification unilatérale du client*. On modélise habituellement les attaques contre l'authentification unilatérale du client auprès du serveur en considérant les sessions où le serveur accepte, mais sans aucun client partenaire. On note par Succ l'événement suivant : une instance du serveur accepte alors qu'il n'a aucune instance d'un client comme partenaire (c'est-à-dire notamment aucune instance d'un client qui aurait la même transcription partielle de la communication).

Le **succès-auth** $\mathsf{Succ}_P^{\mathtt{auth}}(\mathcal{A})$ et la fonction de succès du protocole P sont respectivement :

$$\mathsf{Succ}_P^{\mathtt{auth}}(\mathcal{A}) = \Pr[\mathsf{Succ}], \quad \mathsf{Succ}_P^{\mathtt{auth}}(t, Q) = \max_{\mathcal{A}} \left\{ \mathsf{Succ}_P^{\mathtt{auth}}(\mathcal{A}) \right\},$$

où le maximum est pris sur tous les attaquants avec une complexité temporelle d'au plus t et pouvant faire au plus Q requêtes.

3.2.2 Modélisation et pouvoir de l'attaquant

Participants, Sessions et Partenariat. Dans le modèle d'AKE que nous considérons ici, les participants sont ou bien des clients $\mathcal{C}$ ou bien un unique serveur d'authentification S. Le serveur et chaque client peuvent initier et participer à plusieurs sessions en parallèle. Une instance i de l'entité U, où U est soit le serveur soit un client, est notée par Π_U^i. Cette instance inclut trois variables, initialisées à *null* :

- pid_U^i : l'*identificateur de partenaire*, qui est l'instance avec qui Π_U^i pense interagir,
- sid_U^i : l'*identificateur de session*, en pratique cela peut être la transcription des messages reçus et envoyés par Π_U^i, excepté le dernier message échangé,
- acc_U^i : une variable booléenne qui est déterminée à la fin de la session et désigne si l'instance Π_U^i passe à l'état *accepte* ou pas.

Les deux instances Π_U^i et $\Pi_{U'}^j$ sont dites *partenaires* si les conditions suivantes sont remplies : (1) $\mathsf{pid}_U^i = \Pi_{U'}^j$, et $\mathsf{pid}_{U'}^j = \Pi_U^i$; (2) $\mathsf{sid}_U^i = \mathsf{sid}_{U'}^j \neq null$; (3) $\mathsf{acc}_U^i = \mathsf{acc}_{U'}^j = 1$.

Requêtes autorisées. Les requêtes que peut effectuer l'adversaire sont les suivantes :

- $\mathsf{Send}(m, \Pi_U^i)$: cette requête permet à l'adversaire de jouer avec les instances, en interceptant, modifiant ou en créant des messages. La réponse à cette requête est la réponse générée par l'instance Π_U^i au message m.
- $\mathsf{Reveal}(\Pi_U^i)$: cette requête modélise la fuite d'information sur la clef de session établie par les parties. C'est le cas notamment si elle est utilisée par la suite dans un protocole symétrique faible. Si aucune clef de session n'est définie pour cette instance ou si cette instance (ou son partenaire) a déjà été testée, alors la sortie est $\perp$. Dans le cas contraire, l'oracle renvoie la clef de session calculée par l'instance Π_U^i.
- $\mathsf{CorruptKey}(\mathcal{C})$: cette requête modélise la capacité de corruption de l'adversaire. Il peut en effet voler le facteur d'authentification d'un ou de plusieurs clients. À cette requête, l'oracle répond par la clef à long terme du client $\mathcal{C}$.

Afin de modéliser formellement la sécurité sémantique concernant l'authentification du client, on autorise également l'adversaire à faire des requêtes-Test, mais au serveur uniquement : nous sommes intéressés par la confidentialité de la clef uniquement lorsqu'elle est établie avec un vrai serveur. Bien sûr, pour parvenir à ses fins, l'adversaire peut essayer de se faire passer pour le serveur auprès d'un client afin d'apprendre certaines informations à propos des clefs à long terme du client. Mais seules les usurpations de l'identité du client seront considérées comme des attaques menées avec succès. Les requêtes-Test sont modélisées de la façon suivante :

- $\mathsf{Test}(\Pi_S^i)$: Si Π_S^i n'est pas *fraîche* (voir ci-dessous pour la définition), alors $\perp$ est renvoyé, sinon l'oracle envoie
  - la clef de session de l'instance Π_S^i (c'est-à-dire $\mathsf{Reveal}(\Pi_S^i)$), si $b = 1$ – le cas réel ;
  - un clef aléatoire uniforme prise dans le même domaine, si $b = 0$ – le cas aléatoire.

Fraîcheur. La notion de fraîcheur définit essentiellement les clefs de session qui ne sont pas connues trivialement de l'adversaire. Puisque nous allons nous concentrer seulement sur la fraîcheur du serveur, nous disons qu'une instance Π_S^i est fraîche si :

- avant l'acceptation, la clef à long terme de $\mathcal{C}$ (le partenaire de Π_S^i) n'était pas corrompue. Cela signifie qu'aucune requête-CorruptKey n'a été faite au client $\mathcal{C}$;
- aucune requête-Reveal n'a été envoyée à Π_S^i ou à son partenaire.

3.3 Modèle de sécurité d'un échange de clefs multi-facteurs

Dans cette partie, nous décrivons les particularités du modèle de sécurité pour l'échange de clefs multi-facteurs authentifié (noté MAKE dans la suite, pour *Multy-factors Authenticated Key Exchange*). Ce modèle est construit à partir du modèle précédent en tenant compte des particularités de chacun des facteurs.

3.3.1 Notions de sécurité

Les définitions des notions de sécurité dans le cas du MAKE sont presque identiques à celles du cas de l'AKE classique. Pour la sécurité sémantique, la définition est la même, seules changent les notations de l'**avantage-make** $\mathsf{adv}_P^{\mathtt{make}}(\mathcal{A})$ et de la fonction avantage du protocole P qui sont respectivement :

$$\mathsf{adv}_P^{\mathtt{make}}(\mathcal{A}) = 2 \cdot \Pr[\mathsf{Succ}] - 1, \quad \mathsf{adv}_P^{\mathtt{make}}(t, Q) = \max_{\mathcal{A}} \left\{ \mathsf{adv}_P^{\mathtt{make}}(\mathcal{A}) \right\},$$

où le maximum est pris sur tous les attaquants ayant une complexité temporelle plus petite que t et un nombre de requêtes au plus égal à Q.

La notion d'authentification unilatérale est exactement identique à celle de l'AKE classique.

Seule la définition de la sécurité dans le temps change légèrement pour tenir compte de l'augmentation du nombre de facteurs. Comme expliqué ci-dessous, l'adversaire est autorisé à corrompre une ou plusieurs données du client $\mathcal{C}$: il peut apprendre son mot de passe $\mathsf{pwd}_\mathcal{C}$ (par ingéniérie sociale par exemple), récupérer sa clef privée $\mathsf{sk}_\mathcal{C}$ (grâce à une attaque par canaux auxiliaires sur la carte à puce par exemple), ou en cassant l'hypothèse de présence (en attaquant ou en dupant le capteur).

Afin de prendre en compte la sécurité dans le temps, on autorise l'adversaire à effectuer des requêtes-Test, même quand les 3 requêtes-$\mathsf{CorruptKey}$ ont été faites, tant que la session a été complétée avant la corruption totale du client. Pour modéliser ceci, on modifie la notion de fraîcheur : on considére qu'avant l'acceptation, une session est fraîche si moins de 3 requêtes-$\mathsf{CorruptKey}$ ont été effectuées (voir la sous-section suivante pour plus de formalisme).

3.3.2 Modélisation et pouvoirs de l'attaquant

Pour décrire le modèle de sécurité du MAKE, nous nous appuyons sur le modèle de sécurité de l'AKE classique, décrit à la section précédente, en insistant uniquement sur les différences entre les deux modèles.

Participants, Sessions et Partenariat. Dans le MAKE, la modélisation des participants, des sessions et du partenariat est en tout point identique au cas de l'AKE classique.

Clefs à long terme. Chaque client $\mathcal{C}$ possède un triplet $t_\mathcal{C} = (\mathcal{D}_\mathcal{C}, \mathsf{sk}_\mathcal{C}, \mathsf{pwd}_\mathcal{C})$, où $\mathcal{D}_\mathcal{C}$ est une distribution de probabilité pour la donnée biométrique, alors que $\mathsf{sk}_\mathcal{C}$ et $\mathsf{pwd}_\mathcal{C}$ sont respectivement une clef privée à forte min-entropie et un mot de passe à faible min-entropie. Le serveur possède une liste de triplets $t_S = \langle t_S[\mathcal{C}] \rangle$, où $t_S[\mathcal{C}]$ est une transformée de $t_\mathcal{C}$. Plus précisément, quand le client $\mathcal{C}$ s'inscrit dans le système, il génère une mesure biométrique $W_\mathcal{C}$ dite « de référence », selon la distribution $\mathcal{D}_\mathcal{C}$, ainsi que deux données privées $\mathsf{sk}_\mathcal{C}$ et $\mathsf{pwd}_\mathcal{C}$. Le triplet $t_S[\mathcal{C}]$ est une transformation injective de $(W_\mathcal{C}, \mathsf{sk}_\mathcal{C}, \mathsf{pwd}_\mathcal{C})$.

Requêtes autorisées. Les requêtes que peut effectuer l'adversaire sont largement similaires à celles du modèle AKE classique. Nous ne détaillons que celles qui changent par rapport à celui-ci ainsi que les requêtes nouvelles :

- $\mathsf{Compute}(\Pi^i_\mathcal{C}, W', \mathsf{sk}, \mathsf{pwd})$: il a déjà été décrit dans la sous-section 3.1.4, mais on en rappelle succintement la définition. Selon l'état du client, à partir des secrets sk, pwd et d'une nouvelle acquisition W' de la donnée biométrique, il calcule honnêtement le message que génèrerait $\mathcal{C}$. La variable aléatoire W' doit être choisie selon une distribution de probabilité $\mathcal{D}$ telle que $\Pr[W' \leftarrow \mathcal{D}, W_\mathcal{C} \leftarrow \mathcal{D}_\mathcal{C} : d_H(W', W_\mathcal{C}) > \tau] \geq 1 - \varepsilon_{\mathsf{fa}}$.

- $\mathsf{Send}(m, \Pi_U^i)$: de même que précédemment, la réponse à cette requête est la réponse générée par l'instance Π_U^i au message m. Cependant, comme expliqué ci-dessous, dans le cas du MAKE, si $\mathsf{pid}_S^i = \Pi_{\mathcal{C}}^j$ est une instance du client avec laquelle le serveur croit parler, si l'hypothèse de présence n'a pas été cassée pour le client $\mathcal{C}$ et si le calcul de m fait intervenir la donnée biométrique, alors m doit avoir été généré précédemment par une requête $\mathsf{Compute}(\Pi_{\mathcal{C}}^j, W', \mathsf{sk}, \mathsf{pwd})$ à l'oracle de calcul.
- $\mathsf{CorruptKey}(\mathcal{C}, a)$: cette requête modélise toujours la capacité de corruption de l'adversaire. Il peut cette fois voler/casser *un ou plusieurs* facteurs d'authentification des clients.
  - Si $a = 1$, l'oracle renvoie le mot de passe $\mathsf{pwd}_{\mathcal{C}}$ de $\mathcal{C}$;
  - si $a = 2$, l'oracle renvoie la clef privée $\mathsf{sk}_{\mathcal{C}}$ de $\mathcal{C}$;
  - si $a = 3$, l'attaquant est à partir de maintenant autorisé à soumettre via une requête-Send *n'importe quel* message, sans avoir à le générer préalablement grâce à l'oracle de calcul. Cela modélise l'attaque contre l'hypothèse de présence, l'attaquant devient maître du capteur et peut faire tous les calculs qu'il souhaite sans restriction aucune.

  On insiste sur le fait que dans ce qui suit, nous nous limitons à des corruptions non-adaptatives : aucune corruption d'un facteur d'authentification d'un client ne peut être effectuée alors qu'une ou plusieurs sessions faisant intervenir ce client sont en cours, l'adversaire doit attendre que toutes ces sessions se terminent pour corrompre le ou les facteurs, après quoi il pourra de nouveau initier des sessions avec ce client.
- $\mathsf{Test}(\Pi_S^i)$: ces requêtes sont définies de la même manière que dans le cas de l'AKE classique, même si la notion de fraîcheur, elle, change (voir ci-dessous).
- $\mathsf{Reveal}(\Pi_U^i)$: ce sont les mêmes requêtes que dans le cas de l'AKE classique.

Fraîcheur. La notion de fraîcheur évolue légèrement, nous disons maintenant qu'une instance Π_S^i du serveur est fraîche si :

- avant l'acceptation, $\mathcal{C}$ (le partenaire de Π_S^i) n'était pas *totalement corrompu*. Cela signifie que strictement moins de 3 requêtes-CorruptKey ont été faites au client $\mathcal{C}$;
- aucune requête-Reveal n'a été envoyée à Π_S^i ou à son partenaire.

3.4 Schéma proposé

3.4.1 Présentation du schéma

La description complète de notre protocole est fournie dans la figure 3.1. Elle suppose une initialisation commune, avec des paramètres (u, v, p, g, q), où p et q sont des nombres premiers tels que q divise $p-1$, g est un élément d'ordre q de $\mathbb{Z}_p^\star$ et engendre le sous-groupe $\mathbb{G}$. Ensuite, u et v sont des éléments choisis aléatoirement selon la distribution uniforme dans $\mathbb{G}$. Nous modélisons également H comme un oracle aléatoire [BR93b].

Le serveur enregistre toutes les données correspondant à l'utilisateur $\mathcal{C}$ et fournies par ce dernier durant la phase d'inscription :

- la clef publique $h = g^{x_{\mathcal{C}}}$, reliée au secret à forte entropie $x_{\mathcal{C}}$;
- un chiffrement Elgamal [Elg84] de la mesure biométrique de référence $W_{\mathcal{C}} = (W_i)_{i \leq N}$ —où W_i est le i-e bit de $W_{\mathcal{C}}$ et N le nombre de bits— avec la clef publique $h = g^{x_{\mathcal{C}}}$, c'est-à-dire un N-uplet de couples $(g^{r_i}, h^{r_i} g^{W_i})_i$;
- le mot de passe $\mathsf{pwd}_{\mathcal{C}} \in Dict \subset \mathbb{Z}_q^\star$.

On peut remarquer que le serveur ne connaît pas les données biométriques des clients puisqu'elles sont chiffrées avec une clef choisie par les clients.

$\mathcal{C} : (W'_{\mathcal{C}} = (W'_i)_i, \mathsf{sk}_{\mathcal{C}} = x_{\mathcal{C}}, \mathsf{pwd}_{\mathcal{C}})$		$\mathcal{S} : \left((g^{r_i}, h^{r_i} g^{W_i})_i, h = g^{x_{\mathcal{C}}}, \mathsf{pwd}_{\mathcal{C}}, \mathcal{C}\right)_{\mathcal{C}}$
$b \xleftarrow{\$} \mathbb{Z}_q$ et $B = g^b$, $B^* = B \cdot v^{\mathsf{pwd}_{\mathcal{C}}}$	$\xrightarrow{\mathcal{C},\ B^*}$	Pour $1 \leq i \leq N, r'_i \xleftarrow{\$} \mathbb{Z}_q$ et calculer $g^{s_i} = g^{r'_i} \cdot g^{r_i}, h^{s_i} g^{W_i} = h^{r'_i} \cdot h^{r_i} g^{W_i}$
	$\xleftarrow{\mathcal{S}, (g^{s_i})_i, A^*}$	$a \xleftarrow{\$} \mathbb{Z}_q$, $A = g^a$, $A^* = A \cdot u^{\mathsf{pwd}_{\mathcal{C}}}$
Pour $1 \leq i \leq N$, calculer $H(K'_i) = \alpha'_i \Vert \beta'_i \Vert k'_i$ avec : $K_{\mathcal{C}} = \left(\frac{A^*}{u^{\mathsf{pwd}_{\mathcal{C}}}}\right)^b, K'_{\mathcal{C},i} = (g^{s_i})^{x_{\mathcal{C}}} \cdot g^{W'_i}$ $K'_i = \mathcal{S} \Vert \mathcal{C} \Vert (g^{s_i})_i \Vert A^* \Vert B^* \Vert K'_{\mathcal{C},i} \Vert K_{\mathcal{C}} \Vert \mathsf{pwd}_{\mathcal{C}} \Vert i$	$\xrightarrow{(\alpha'_i)_i}$	Pour $1 \leq i \leq N$, Calculer $H(K_i) = \alpha_i \Vert \beta_i \Vert k_i$ avec : $K_{\mathcal{S}} = \left(\frac{B^*}{v^{\mathsf{pwd}_{\mathcal{C}}}}\right)^a, K'_{\mathcal{S},i} = h^{s_i} \cdot g^{W_i}$ $K_i = \mathcal{S} \Vert \mathcal{C} \Vert (g^{s_i})_i \Vert A^* \Vert B^* \Vert K'_{\mathcal{S},i} \Vert K_{\mathcal{S}} \Vert \mathsf{pwd}_{\mathcal{C}} \Vert i$ Si $\#\{i \colon \alpha_i \neq \alpha'_i\} \leq t$ Alors $\mathsf{acc} = \mathsf{true}$, $K = \mathsf{lsb}_{\mathsf{k}}\left(\Vert_{i \colon \alpha_i = \alpha'_i} k_i\right)$
Si $\#\{i \colon \beta_i \neq \beta'_i\} \leq t$ Alors $\mathsf{acc} = \mathsf{true}$, $K' = \mathsf{lsb}_{\mathsf{k}}\left(\Vert_{i \colon \alpha_i = \alpha'_i} k_i\right)$ Sinon $\mathsf{acc} = \mathsf{false}$, $K' \xleftarrow{\$} \{0,1\}^{\mathsf{k}}$	$\xleftarrow{(\beta_i)_i}$	Sinon $\mathsf{acc} = \mathsf{false}$, $K \xleftarrow{\$} \{0,1\}^{\mathsf{k}}, \beta_i \xleftarrow{\$} \{0,1\}^{\ell}$

FIG. 3.1 – Notre protocole d'échange de clefs authentifié multi-facteurs.

Pour s'authentifier, le client $\mathcal{C}$ possède une mesure biométrique éphémère $W'_{\mathcal{C}} = (W'_i)_i$, une clef privée à long terme $x_{\mathcal{C}}$ et un mot de passe $\mathsf{pwd}_{\mathcal{C}} \in \mathit{Dict} \subset \mathbb{Z}_q^\star$.

Le protocole garantit que, si la mesure biométrique $W'_{\mathcal{C}}$ est suffisamment proche de la donnée biométrique de référence $W_{\mathcal{C}}$ (qui vous êtes), si la clef privée $x_{\mathcal{C}}$ correspond à la clef publique h (ce que vous possédez), et si les mots de passe sont identiques (ce que vous connaissez), alors le serveur accepte le client, et ils génèrent une clef secrète éphémère commune $K' = K$.

Plus précisément, nous prouvons qu'aucun adversaire ne peut se faire passer pour un client auprès du serveur, sauf si les trois facteurs d'authentification sont corrompus. Toutes les clefs, sur lesquelles le client et le serveur se sont mis d'accord, sont sémantiquement sûres, même après 3 corruptions (c'est la sécurité dans le temps).

Succinctement, (B^*, A^*) correspond à la partie authentification par mot de passe (comme dans EKE [BM92, AP05]) ; pour chaque i, (h, g^{s_i}) est un échange de clefs Diffie-Hellman qui génère la clef $g^{x_{\mathcal{C}} \cdot s_i}$, avec $x_{\mathcal{C}}$ utilisée pour l'authentification. On remarque qu'à chaque fois on génère de nouveaux s_i parfaitement aléatoires et donc que l'échange de clefs Diffie-Hellman n'est pas statique. Le bit W_i (ou W'_i pour le client) de la donnée biométrique est utilisé comme masque, on obtient donc finalement $g^{x_{\mathcal{C}} \cdot s_i} \cdot g^{W_i}$ ($g^{x_{\mathcal{C}} \cdot s_i} \cdot g^{W'_i}$ pour le client). Si pour la plupart des i les masques W_i et W'_i sont égaux, alors pour la plupart des i les authentificateurs α'_i et les vérificateurs α_i seront égaux également, de même que les β_i et β'_i, et les k_i et k'_i.

Pour compléter la définition de partenariat dans notre protocole, nous devons préciser que le sid est égal aux deux premiers échanges $((\mathcal{C}, B^*), (\mathcal{S}, (g^{s_i})_i, A^*))$.

Nous allons montrer maintenant que ce protocole est sûr sous l'hypothèse calculatoire Diffie-Hellman [DH76].

3.4.2 Preuve de sécurité

Avant d'établir les résultats de sécurité, nous rappelons que la définition du problème Diffie-Hellman calculatoire, problème sur lequel repose la sécurité du protocole, est donnée dans la sous-section 2.3.1.

Faux rejet et fausse acceptance. Nous rappelons également que pour tout client $\mathcal{C}$ et pour tout adversaire qui utilise une vraie donnée biométrique (différente de celle du client), on a $\Pr[d_H(W', W_{\mathcal{C}}) > \tau] \geq 1 - \varepsilon_{\mathsf{fa}}$ où τ est un entier plus grand que t et $\varepsilon_{\mathsf{fa}}$ désigne la probabilité de fausse acceptance. Le protocole n'augmente pas la probabilité de faux rejet mais il augmente la probabilité de fausse acceptance, à cause des tests supplémentaires sur les égalités $\alpha_i = \alpha'_i$. L'augmentation de probabilité est majorée par

$$\Pr\left[\#\{i : \alpha'_i \neq \alpha_i\} \leq t \,|\, d_H(W', W_{\mathcal{C}}) > \tau\right] \leq \frac{\binom{\tau}{\tau - t}}{2^{\ell(\tau - t)}}.$$

Correction du protocole. Une session de protocole entre deux entités honnêtes est correcte si pour tout i, $K_i = K'_i$ est équivalent à $\alpha_i = \alpha'_i$ ou $\beta_i = \beta'_i$. Elle échoue s'il existe un indice i tel que $K_i \neq K'_i$ et $\alpha_i = \alpha'_i$ ou $\beta_i = \beta'_i$. Puisqu'il y a au plus t indices i tel que $K_i \neq K'_i$, la probabilité qu'un protocole honnête ne soit pas correct est majorée par $2t \Pr[\alpha_i = \alpha'_i | K_i \neq K'_i]$ qui est égal à $2t/2^{\ell}$.

Sécurité du protocole. Nous présentons maintenant le théorème principal de cette partie, théorème qui précise la sécurité du protocole.

Théorème 3.1. *Considérons le protocole P décrit ci-dessus sur un groupe d'ordre premier q, où le dictionnaire des mots de passe $Dict \subset \mathbb{Z}_q^\star$ est muni de la distribution $\mathcal{D}_{\mathsf{pwd}}$. Soit $\mathcal{A}$ un adversaire contre la sécurité sémantique avec une complexité en temps d'au plus T, faisant moins de q_{session} requêtes-Send et demandant moins de q_h requêtes à l'oracle aléatoire. Alors on a :*

$$\begin{aligned}
\mathsf{adv}_P^{\mathtt{make}}(\mathcal{A}) &\leq 2\sum_{\mathcal{C}} \mathcal{D}_{\mathsf{pwd}}(q_{\mathcal{C}}) + 4q_h^2 \cdot \mathsf{Succ}_{p,g}^{\mathsf{cdh}}(T + 4\tau_e) + \frac{q_{\mathsf{session}}^2}{q} + \frac{2q_h}{q} \\
\mathsf{Succ}_P^{\mathtt{auth}}(\mathcal{A}) &\leq \sum_{\mathcal{C}} \mathcal{D}_{\mathsf{pwd}}(q_{\mathcal{C}}) + 2q_h^2 \cdot \mathsf{Succ}_{p,g}^{\mathsf{cdh}}(T + 4\tau_e) + \frac{q_{\mathsf{session}}^2}{2q} + \frac{q_h}{q} \\
&\quad + q_{\mathsf{session}} \left(\varepsilon_{\mathsf{fa}} + \frac{\binom{\tau}{\tau - t}}{2^{\ell(\tau - t)}} + \frac{N^t \cdot (2^\ell - 1)^t}{2^{\ell N}(t-1)!} \right)
\end{aligned}$$

où τ_e désigne le temps de calcul pour une exponentiation et $q_{\mathcal{C}}$ le nombre de sessions actives que l'adversaire établit contre le client $\mathcal{C}$.

Démonstration. La preuve consiste en une suite de jeux :

Game 0. Ce jeu décrit la vraie d'attaque contre le protocole. Nous sommes intéressés par les deux événements suivant :

- $\mathbf{S}_0$ (pour la sécurité sémantique) qui est vrai si l'adversaire devine correctement le bit b choisi au début du jeu.
- $\mathbf{A}_0$ (pour l'authentification du client), qui a lieu si une instance du serveur accepte sans instance client partenaire (c'est-à-dire sans instance d'un client avec qui il partage la même transcription des messages échangés).

Dans le jeu $\mathbf{Game}_n$, nous étudions l'événement $\mathbf{A}_n$ et l'événement $\mathbf{S}_n$. On fait remarquer que

$$\mathsf{adv}_P^{\mathtt{make}}(\mathcal{A}) = 2\Pr[\mathbf{S}_0] - 1, \quad \mathsf{Succ}_P^{\mathtt{auth}}(\mathcal{A}) = \Pr[\mathbf{A}_0].$$

Par conséquent on a

$$\begin{aligned} \mathsf{adv}_P^{\mathtt{make}}(\mathcal{A}) = 2\Pr[\mathbf{S}_n] - 1 + 2(\Pr[\mathbf{S}_0] - \Pr[\mathbf{S}_n]) &\leq 2\Pr[\mathbf{S}_n] - 1 + 2\sum_{i=0}^{n-1}\Delta_i \\ \mathsf{Succ}_P^{\mathtt{auth}}(\mathcal{A}) = \Pr[\mathbf{A}_n] + (\Pr[\mathbf{A}_0] - \Pr[\mathbf{A}_n]) &\leq \Pr[\mathbf{A}_n] + \sum_{i=0}^{n-1}\Delta_i, \end{aligned}$$

où Δ_i désigne la distance entre les jeux $\mathbf{Game}_i$ et $\mathbf{Game}_{i+1}$.

Game 1. Dans ce jeu, nous simulons les oracles aléatoires (H, mais aussi une fonction supplémentaire H' qui va apparaître dans le jeu $\mathbf{Game}_3$) de façon classique, en maintenant deux listes Λ_H et $\Lambda_{H'}$ contenant les requêtes faites à H et H' respectivement et les réponses associées. Nous simulons également tous les oracles en répondant aux requêtes-Send, requêtes-Reveal et requêtes-Test comme les vrais joueurs le feraient. À partir de cette simulation, nous voyons que ce jeu est indistinguable de la vraie attaque : $\Delta_0 = 0$.

Comme les distributions de probabilité des données biométriques sont publiques, pour chaque client nous pouvons générer aléatoirement une donnée biométrique utilisée/connue comme référence par le serveur. Quand une simulation du client le nécessite, nous pouvons générer une donnée biométrique aléatoirement selon la distribution de probabilité (publique) de la donnée biométrique du client.

Game 2. Afin de garantir l'indépendance des sessions, nous annulons les jeux dans lesquels les transcriptions des communications $((\mathcal{C}, B^*), (\mathcal{S}, A^*, (g^{s_i})_i))$ de deux sessions entrent en collision. Puisque les communications font intervenir toujours au moins une partie honnête, soit $(A^*, (g^{s_i})_i)$, soit B^* est choisi selon une distribution uniforme. Par conséquent, la probabilité de collision est majorée par $q_{\mathsf{session}}^2/2q$, où q_{session} est le nombre de sessions. On a donc : $\Delta_1 \leq q_{\mathsf{session}}^2/2q$.

Game 3. Pour la génération des authentificateurs et des clefs de session, nous remplaçons maintenant l'oracle H par un oracle privé H' dans les cas suivants : pour les sessions qui sont fraîches (ce qui peut être facilement vérifié : ce sont les sessions qui font intervenir le serveur et pour lesquelles le client correspondant supposé n'est pas totalement corrompu), et pour les sessions qui font intervenir un client (mais pas le serveur) dont le mot de passe et la clef privée sont inconnus de l'adversaire (aucune des requêtes 1-CorruptKey et des requêtes 2-CorruptKey n'ont été faites à ce client). Dans ces cas là, on calcule K_i et K_i' de la façon suivante :

$$K_i = \mathcal{S}\Big\|\mathcal{C}\Big\|(g^{s_i})_i\Big\|A^*\Big\|B^*\Big\|g^{W_i}\Big\|i, \quad K_i' = \mathcal{S}\Big\|\mathcal{C}\Big\|(g^{s_i})_i\Big\|A^*\Big\|B^*\Big\|g^{W_i'}\Big\|i,$$

puis on calcule

$$H'(K_i) = \alpha_i\|\beta_i\|k_i, \quad H'(K_i') = \alpha_i'\|\beta_i'\|k_i'.$$

Comme nous l'avons déjà expliqué, nous avons choisi une donnée biométrique de référence au hasard pour chaque utilisateur et, quand cela est nécessaire, nous pouvons générer une nouvelle donnée biométrique aléatoire selon la distribution de probabilité (publique) du trait biométrique du client. Par conséquent, nous pouvons inclure g^{W_i} et/ou $g^{W_i'}$ dans les calculs publics précédents, afin de rendre K_i et K_i' éventuellement différents, même pour des données biométriques compatibles.

Nous n'utilisons plus $K_{\mathcal{C}}$, $K'_{\mathcal{C},i}$, $K_{\mathcal{S}}$ et $K'_{\mathcal{S},i}$, par conséquent nous pouvons omettre de les calculer. Par ailleurs, nous n'utilisons plus A et B, nous pouvons donc changer la manière de

calculer A^* et B^* en faisant ainsi : $a^* \xleftarrow{\$} \mathbb{Z}_q$, $A^* = g^{a^*}$ et $b^* \xleftarrow{\$} \mathbb{Z}_q$, $B^* = g^{b^*}$. Enfin, puisque nous n'utilisons plus ni le mot de passe ni la clef privée, nous pouvons les choisir au dernier moment, c'est-à-dire au moment d'une requête de corruption du mot de passe (1-$\mathsf{CorruptKey}$) ou d'une requête de corruption de la clef privée (2-$\mathsf{CorruptKey}$) ou à la toute fin du jeu (quand l'adversaire donne sa réponse b').

Cependant, quand un client est totalement corrompu (et que l'adversaire joue contre le serveur le rôle de ce client), ou quand l'adversaire joue contre un client dont il connaît le mot de passe et la clef privée, les clefs K_i et K'_i sont calculées normalement et nous utilisons l'oracle public H.

Toutes les corruptions sont non-adaptatives, donc, quand une session est initiée, nous connaissons le degré de corruption du client. Par conséquent, les requêtes à l'oracle-$\mathsf{Compute}$ vont se concentrer sur les sessions pour lesquelles nous connaissons les degrés de corruption, puisque la biométrie n'intervient que dans le troisième échange. Cet oracle doit en effet savoir comment effectuer la simulation des K'_i, en utilisant H ou H'.

Les jeux $\mathbf{Game}_2$ et $\mathbf{Game}_3$ sont indistinguables, à moins de demander certaines requêtes spécifiques (pour une session initiée avant de demander la dernière requête-$\mathsf{CorruptKey}$) : si l'adversaire demande

$$\mathcal{S}\left\|\mathcal{C}\right\|(g^{s_i})_i\left\|A^*\right\|B^*\left\|K'_{\mathcal{C},i}\right\|K_{\mathcal{C}}\left\|\mathsf{pwd}_{\mathcal{C}}\right\|i \text{ ou } \mathcal{S}\left\|\mathcal{C}\right\|(g^{s_i})_i\left\|A^*\right\|B^*\left\|K'_{\mathcal{S},i}\right\|K_{\mathcal{S}}\left\|\mathsf{pwd}_{\mathcal{C}}\right\|i,$$

pour une certaine transcription des communications $((\mathcal{C}, B^*), (\mathcal{S}, A^*, (g^{s_i})_i))$, et pour un certain index i, à la fonction H, alors que la fonction H' a été utilisée par le simulateur. Nous désignons par AskH_3 cet événement. On notera que l'on peut décider si cet événement est vrai ou pas uniquement quand la clef et le mot de passe ont été choisis.

Dans ce jeu, pour tous les clients, les $(\alpha_i)_i$, les $(\beta_i)_i$ et la clef sont calculés à partir de l'oracle aléatoire privé. Par conséquent, quelque soit la valeur du bit b intervenant dans les requêtes-Test, la réponse est aléatoire, indépendante pour toutes les sessions, à moins qu'une transcription des communications $((\mathcal{C}, B^*), (\mathcal{S}, A^*, (g^{s_i})_i))$ n'apparaisse deux fois, ce qui a déjà été exclu dans le jeu $\mathbf{Game}_2$. Par conséquent, nous avons :

$$\Delta_2 \leq \Pr[\mathsf{AskH}_3] \qquad \text{et} \qquad \Pr[\mathbf{S}_3] = \frac{1}{2}.$$

De même, la seule possibilité pour l'adversaire de s'authentifier contre une instance du vrai serveur est de deviner les α_i au hasard ou d'utiliser l'oracle-$\mathsf{Compute}$, à moins que la transcription $((\mathcal{C}, B^*), (\mathcal{S}, A^*, (g^{s_i})_i))$ n'apparaisse deux fois.

Si l'adversaire essaie de deviner les α_i au hasard, puisque $|\alpha_i| = \ell$, sa probabilité de succès est majorée par :

$$\frac{1}{2^{N\ell}} \cdot \sum_{k=0}^{t} \binom{N}{k} (2^\ell - 1)^k \leq \frac{N^t \cdot (2^\ell - 1)^t}{2^{\ell N}(t-1)!}.$$

S'il utilise l'oracle-$\mathsf{Compute}$, tous les α'_i et les β_i sont générés par un oracle de calcul de confiance et puisque, dans ce cas là, l'adversaire utilise sa propre donnée biométrique W', qui est, avec grande probabilité, assez différente de la donnée biométrique du client $W_{\mathcal{C}}$, sa probabilité de succès est exactement égale à la probabilité de fausse acceptance calculée un peu plus haut.

Par conséquent,

$$\Pr[\mathbf{A}_3] \leq q_{\mathsf{session}} \left(\varepsilon_{\mathsf{fa}} + \frac{\binom{\tau}{\tau-t}}{2^{\ell(\tau-t)}} + \frac{N^t \cdot (2^\ell - 1)^t}{2^{\ell N}(t-1)!} \right).$$

Game 4. Notre but grâce, à ce jeu, est de majorer la probabilité de l'événement AskH_3. Nous désignons par AskH_4 le même événement dans ce jeu et rappelons que $\mathsf{AskH}_3 \leq \mathsf{AskH}_4 + \Delta_3$. Nous allons donc majorer successivement Δ_3 puis AskH_4.

Voici comment nous définissons le jeu $\mathbf{Game}_4$ par rapport au jeu $\mathbf{Game}_3$. Nous ne générons plus u et v nous même au hasard. Dans ce jeu, nous recevons un couple Diffie-Hellman $(X = g^x, Y = g^y)$ d'une entité extérieure et fixons $u = X$ et $v = Y$. Cela ne change pas la distribution de probabilité de u et v, par contre, nous utilisons l'adversaire pour résoudre le problème Diffie-Hellman lié à (X, Y) : nous montrons que la probabilité de l'événement AskH est liée à la probabilité de réussir à calculer la valeur du résultat Diffie-Hellman de (X, Y). Par ailleurs, nous annulons maintenant les jeux pour lesquels, pour une transcription des échanges $((\mathcal{C}, B^*), (\mathcal{S}, A^*, (g^{s_i})_i))$ qui à la fois

- a été générée avant qu'une requête de corruption de mot de passe n'ait été faite pour le client $\mathcal{C}$, et
- provient d'une exécution faisant intervenir l'adversaire, contre une instance du serveur ou une instance du client $\mathcal{C}$,

il y a deux quadruplets $(A^*, B^*, \mathsf{CDH}_{p,g}(A^*/u^{\mathsf{pwd}_k}, B^*/v^{\mathsf{pwd}_k}), i_k)$, avec deux mots de passe pwd_0 et pwd_1 différents et deux indices i_0 et i_1, éventuellement différents, tels que

$$\mathcal{S}\left\|\mathcal{C}\right\|(g^{s_i})_i\left\|A^*\right\|B^*\left\|K'_{\mathcal{S},i_k}\right\|\mathsf{CDH}_{p,g}(A^*/u^{\mathsf{pwd}_k}, B/v^{\mathsf{pwd}_k})\left\|\mathsf{pwd}_k\right\|i_k$$

soit dans Λ_H.

Distance. Nous prouvons d'abord que $\Delta_3 \leq q_h^2 \cdot \mathsf{Succ}^{\mathsf{cdh}}_{p,g}(T + 3\tau_e)$. Par définition, les jeux $\mathbf{Game}_3$ et $\mathbf{Game}_4$ ne peuvent être distingués que s'il existe deux quadruplets tels que décrit ci-dessus. Nous allons montrer que si ces quadruplets existaient alors on saurait résoudre le problème Diffie-Hellman calculatoire.

Si ces quadruplets existent alors, pour $k \in \{0, 1\}$, $\mathsf{CDH}_{p,g}(A^*/u^{\mathsf{pwd}_k}, B^*/v^{\mathsf{pwd}_k})$ est égal à

$$\frac{\mathsf{CDH}_{p,g}(A^*, B^*) \cdot \mathsf{CDH}_{p,g}(u^{-1}, B^*)^{\mathsf{pwd}_k} \cdot \mathsf{CDH}_{p,g}(A^*, v^{-1})^{\mathsf{pwd}_k}}{\mathsf{CDH}_{p,g}(u, v)^{\mathsf{pwd}_k^2}}.$$

Puisque nous simulons soit A^* ou B^*, nous connaissons au moins un logarithme discret et nous pouvons extraire $\mathsf{CDH}_{p,g}(X, Y)$. Montrons le quand nous connaissons le logarithme discret $B^* = g^{b^*}$, la preuve est similaire quand on connaît le logarithme discret de $A^* = g^{a^*}$. Puisque les deux mots de passe sont différents l'un de l'autre et tous deux différents de zéro,

$$\mathsf{CDH}_{p,g}(X, Y) = \frac{\mathsf{CDH}_{p,g}(A^*/u^{\mathsf{pwd}_0}, B^*/v^{\mathsf{pwd}_0})^{1/\mathsf{pwd}_0(\mathsf{pwd}_1 - \mathsf{pwd}_0)}}{\mathsf{CDH}_{p,g}(A^*/u^{\mathsf{pwd}_1}, B^*/v^{\mathsf{pwd}_1})^{1/\mathsf{pwd}_1(\mathsf{pwd}_1 - \mathsf{pwd}_0)}} \cdot (A^*)^{-b^*/\mathsf{pwd}_0\mathsf{pwd}_1}.$$

Conclusion. Afin de conclure avec le calcul de $\Pr[\mathsf{AskH}_4]$, nous distinguons les cas où la transcription des échanges $((\mathcal{C}, B^*), (\mathcal{S}, A^*, (g^{s_i})_i))$ vient d'une exécution entre :

- deux instances de $\mathcal{C}$ et $\mathcal{S}$, ou une instance de $\mathcal{C}$ ou $\mathcal{S}$ et l'adversaire mais les échanges sont générés par un oracle, événement désigné par $\mathsf{AskH\text{-}Passive}_4$;
- une instance de $\mathcal{C}$ et l'adversaire, et où au moins un échange n'est pas généré par un oracle, événement désigné par $\mathsf{AskH\text{-}with}\mathsf{C}_4$;
- une instance de $\mathcal{S}$ et l'adversaire, et où au moins un échange n'est pas généré par l'oracle-$\mathsf{Compute}$, cet événement désigné par $\mathsf{AskH\text{-}with}\mathsf{S}_4$.

Supposons qu'il y existe un triplet $(A^*, B^*, D = \mathsf{CDH}_{p,g}(A^*/u^{\mathsf{pwd}}, B^*/v^{\mathsf{pwd}}))$ tel que

$$\mathcal{S}\Big\|\mathcal{C}\Big\|(g^{s_i})_i\Big\|A^*\Big\|B^*\Big\|K'_{\mathcal{C},i}\Big\|D\Big\|\mathsf{pwd}\Big\|i \text{ ou } \mathcal{S}\Big\|\mathcal{C}\Big\|(g^{s_i})_i\Big\|A^*\Big\|B^*\Big\|K'_{\mathcal{S},i}\Big\|D\Big\|\mathsf{pwd}\Big\|i$$

soit dans Λ_H, pour un certain mot de passe pwd choisi par l'adversaire.

Si la transcription des échanges correspondante $((\mathcal{C}, B^*), (\mathcal{S}, A^*, (g^{s_i})_i))$ vient d'une exécution entre deux instances de $\mathcal{C}$ et $\mathcal{S}$, cela signifie que et A^* et B^* ont été simulés (et l'adversaire était seulement passif). Dans ce cas, nous connaissons les logarithmes discrets a^* et b^*, et

$$\mathsf{CDH}_{p,g}(A^*/u^{\mathsf{pwd}}, B^*/v^{\mathsf{pwd}}) = \frac{g^{a^*b^*} \cdot (v^{a^*}u^{b^*})^{\mathsf{pwd}}}{\mathsf{CDH}_{p,g}(v,u)^{\mathsf{pwd}^2}}.$$

Puisque pwd est non nul dans $\mathbb{Z}_q$, il peut être inversé modulo q et alors

$$\mathsf{CDH}_{p,g}(X,Y) = \left(\frac{g^{a^*b^*} \cdot v^{a^* \cdot \mathsf{pwd}} \cdot u^{b^* \cdot \mathsf{pwd}}}{\mathsf{CDH}_{p,g}(A^*/u^{\mathsf{pwd}}, B^*/v^{\mathsf{pwd}})}\right)^{1/\mathsf{pwd}^2}.$$

Par conséquent $\Pr[\mathsf{AskH\text{-}Passive}_4] \leq q_h \times \mathsf{Succ}^{\mathsf{cdh}}_{p,g}\ (T + 4\tau_e)$.

Si la transcription des échanges correspondante $((\mathcal{C}, B^*), (\mathcal{S}, A^*, (g^{s_i})_i))$ vient d'une exécution entre une instance de $\mathcal{C}$ et l'adversaire, où au moins un message échangé n'est pas généré par un oracle, cela signifie que B^* a été simulé et que A^* a été généré par l'adversaire. Nous savons que soit la requête de corruption de la clef privée soit la requête de corruption du mot de passe n'a pas été faite (dans le cas contraire la simulation aurait été faite en utilisant H dans le jeu $\mathbf{Game}_3$).

- Supposons que la requête de corruption de la clef privée n'a pas été faite avant l'initiation de cette session, auquel cas $x_{\mathcal{C}}$ et h sont inconnus de l'adversaire. Il lui est alors difficile de calculer $h^{s_i} = (g^{s_i})^{x_{\mathcal{C}}}$ (puisqu'il n'a aucune information du tout) : la probabilité que l'adversaire y parvienne est majorée par q_h/q.
- Si la requête de corruption de la clef privée a été faite, cela implique que la requête de corruption du mot de passe n'a pas été faite. Par définition du jeu $\mathbf{Game}_4$, nous avons annulé certaines parties de sorte qu'il existe au plus un mot de passe pwd tel qu'il existe un indice i, $1 \leq i \leq N$, tel que :

  $$\mathcal{S}\Big\|\mathcal{C}\Big\|(g^{s_i})_i\Big\|A^*\Big\|B^*\Big\|K_{\mathcal{S}}\Big\|K'_{\mathcal{S},i}\Big\|\mathsf{pwd}\Big\|i$$

  est dans Λ_H. En d'autres termes, pour toute transcription des échanges, il existe seulement un mot de passe qui peut être testé par l'adversaire et donc la probabilité de succès de l'adversaire est majorée par $\sum_{\mathcal{C}} \mathcal{D}_{\mathsf{pwd}}(q_{\mathcal{C}})$.

Si la transcription des échanges correspondante $((\mathcal{C}, B^*), (\mathcal{S}, A^*, (g^{s_i})_i))$ vient d'une exécution entre une instance de $\mathcal{S}$ et l'adversaire, où au moins un message échangé n'est pas généré par un oracle-$\mathsf{Compute}$, cela signifie que $(A^*, (g^{s_i})_i)$ a été simulé et que B^* a été généré par l'adversaire. Puisque le serveur accepte un message non généré par l'oracle-$\mathsf{Compute}$, cela signifie que la requête de corruption de la biométrie a été faite pour le client $\mathcal{C}$ correspondant. Par conséquent, la même analyse que ci-dessus peut être faite, en distinguant les cas où la requête de corruption de la clef privée a été faite du cas où celle du mot de passe a été faite.

Nous pouvons en déduire que

$$\Pr[\mathsf{AskH}_4] \leq \sum_{\mathcal{C}} \mathcal{D}_{\mathsf{pwd}}(q_{\mathcal{C}}) + \frac{q_h}{q} + q_h \cdot \mathsf{Succ}^{\mathsf{cdh}}_{p,g}\ (T + 4\tau_e),$$

ce qui permet de conclure. □

3.5 Discussion

3.5.1 Optimalité de la preuve

La majoration de la probabilité d'authentification présentée dans le théorème 3.1 a deux termes principaux qui sont

$$q_S \left(\varepsilon_{\mathsf{fa}} + \frac{\binom{\tau}{\tau-t}}{2^{\ell(\tau-t)}} + \frac{N^t \cdot (2^\ell - 1)^t}{2^{\ell N}(t-1)!} \right) \qquad \text{et} \qquad \sum_{\mathcal{C}} \mathcal{D}_{\mathsf{pwd}}(q_{\mathcal{C}}).$$

Les deux derniers termes dans la parenthèse peuvent être rendus négligeables en augmentant la valeur du paramètre ℓ, ce qui n'est pas le cas des deux autres termes : $q_S \cdot \varepsilon_{\mathsf{fa}}$ et $\sum_{\mathcal{C}} \mathcal{D}_{\mathsf{pwd}}(q_{\mathcal{C}})$. Nous affirmons que notre schéma est optimal et que le résultat de sécurité est fin car ces deux termes ne peuvent pas être évités, même avec un protocole meilleur que le nôtre, alors que les autres termes peuvent être rendus négligeables.

Considérons l'adversaire suivant : $\mathcal{A}$ fait une requête de corruption du mot de passe et une requête de corruption de la clef privée et ensuite essaie de s'authentifier en utilisant son propre trait biométrique. À chaque fois qu'il essaie de s'authentifier, sa probabilité de succès est égale à la probabilité de fausse acceptance. Par conséquent, sa probabilité globale de succès est approximativement égale à $q_S \cdot \varepsilon_{\mathsf{fa}}$. Cette attaque est générique, elle ne dépend pas du protocole d'authentification implémenté, ce qui prouve que le premier terme de la majoration ne peut être évité par aucun moyen cryptographique.

Considérons maintenant l'adversaire qui fait toutes les requêtes de corruptions de la clef privée et toutes les requêtes de corruption de la biométrie, pour tous les adversaires. Le système n'est plus protégé que par les mots de passe. Pour chaque client $\mathcal{C}$, l'adversaire essaie à présent $q_{\mathcal{C}}$ fois de se faire passer pour ce client auprès du serveur en utilisant les $q_{\mathcal{C}}$ mots de passe les plus probables. Pour chaque client $\mathcal{C}$, la probabilité de succès de l'adversaire est majorée par $\mathcal{D}_{\mathsf{pwd}}(q_{\mathcal{C}})$, par conséquent la probabilité globale de succès est approximativement égale à $\sum_{\mathcal{C}} \mathcal{D}_{\mathsf{pwd}}(q_{\mathcal{C}})$. Cela prouve que la meilleure attaque consiste à essayer les mots de passe les plus probables contre autant de clients que possible. Encore une fois, cette attaque est générique et indépendante du protocole employé. Par conséquent, cette borne ne peut pas non plus être évitée.

Les autres termes sont négligeables, notre majoration globale contre l'authentification est donc fine et notre protocole optimal. De la même manière, on peut montrer l'optimalité et la finesse de la borne pour la sécurité sémantique.

3.5.2 Paramètres pratiques

Nous allons maintenant examiner quelles sont les valeurs prises en pratique par les différents termes. Une empreinte de l'iris est habituellement encodée sur $N = 1024$ bits et $t = 300$ est considéré comme un bon seuil pour la distance de Hamming entre deux mesures du même trait biométrique. Avec de tels paramètres, le taux de fausse acceptance est estimé à 2^{-14}. Pour obtenir un taux de faux rejet similaire, on peut raisonnablement prendre un seuil $\tau = 400$. Dans ce cas, si $\ell \geq 4$ alors

$$\frac{\binom{\tau}{\tau-t}}{2^{\ell(\tau-t)}} + \frac{N^t \cdot (2^\ell - 1)^t}{2^{\ell N}(t-1)!} \leq \frac{\binom{400}{100}}{2^{400}} + \frac{2^{3000}}{2^{2896}(299)!} \leq \frac{2^{321}}{2^{400}} + \frac{2^{104}}{2^{2033}} \leq 2^{-78}. \tag{3.1}$$

On rappelle que ℓ est la longueur des étiquettes d'authentification. Plus elles sont courtes, plus le protocole est efficace, du point de vue de la complexité de la communication. Pouvons nous réduire ℓ ? Considérons un adversaire qui a corrompu à la fois le mot de passe et la clef privée.

Avec forte probabilité (plus forte que $\varepsilon_{\mathsf{fa}}$), la distance de Hamming entre une mesure du trait biométrique de l'adversaire et la donnée biométrique de référence du client est approximativement égale à 512, par conséquent il y a environ 512 indices i tels que $\alpha_i = \alpha'_i$. Si $\ell = 1$, il y a environ 512/2 autres indices tels que par hasard α_i soit égal à α'_i, c'est-à-dire près de 768 indices en tout. L'adversaire est donc capable de se faire passer pour le client avec forte probabilité. Par conséquent, si $\ell = 1$, un adversaire peut se faire passer pour un client avec une probabilité plus grande que la probabilité de fausse acceptance.

On en déduit que ℓ doit être plus grand que 2, et la borne (3.1) montre que $\ell = 4$ est un bon choix pour garantir un bon niveau de résistance aux attaques par authentification. Cependant, pour garantir la correction d'une exécution honnête (c'est-à-dire, pour tout indice i, on a $\alpha_i = \alpha'_i$ ou $\beta_i = \beta'_i$ si et seulement si $W_i = W'_i$) une solution est de choisir un ℓ plus grand. Si $\ell = 24$, une exécution honnête se termine avec succès avec une probabilité de $2^{-14} \approx \varepsilon_{\mathsf{fa}}$.

Une autre solution pour garantir la correction d'une session honnête est d'ajouter une étape de distillation [BBCM94] après le protocole. La distillation permet à deux entités, possédant deux clefs éventuellement distinctes mais séparées par une faible distance de Hamming, de se mettre d'accord sur une clef secrète commune, moyennant le fait de révéler quelques uns des bits des clefs secrètes originelles. L'étape de distillation doit se faire sur $\|_i k_i$ et $\|_i k'_i$ en entier et on peut alors choisir $\ell = 4$. En effet, quand $\ell = 4$, il est probable qu'il existe i tel que $\alpha_i = \alpha'_i$ et $\beta_i = \beta'_i$ alors que $W_i \neq W'_i$. Or si $W_i \neq W'_i$ il est probable également que $k_i \neq k'_i$, la distillation permettra de corriger ce genre d'erreurs. Comme la distillation implique de révéler certains bits secrets originels, la clef secrète commune obtenue finalement est alors plus petite que les clefs initiales, mais ceci n'est pas un problème dans notre cas, puisque les clefs d'origines sont relativement grandes.

Une étape de distillation permet également d'éviter certaines attaques par déni de service, attaques au cours desquelles l'adversaire changerait un petit nombre des α'_i (ce qui n'est possible que si l'hypothèse de présence a été cassée) ou des β_i. Si, pour ces indices i, $K_i = K'_i$, alors avec grande probabilité les deux entités génèrent deux clefs secrètes différentes, pensent à tort qu'elles partagent le même secret et acceptent (modifier un petit nombre de α'_i et de β_i ne va très probablement pas modifier la décision finale). Cette attaque peut être détectée et les sessions affectées peuvent être identifiées en rajoutant une étape de confirmation de clef. Cependant, l'avantage de la distillation est qu'elle permet de corriger les erreurs introduites par l'adversaire ou par le hasard, et donc d'éviter d'avoir à rejouer le protocole une fois de plus.

4

Analyse d'un schéma de chiffrement authentifié symétrique

Sommaire

4.1 Introduction

Après avoir effectué l'échange de clefs authentifié, les parties sont en possession d'une (longue) chaîne de bits de taille fixe, qui, aux yeux d'un adversaire, semble uniformément distribuée parmi l'ensemble des chaînes de même taille. Elles vont scinder la chaîne de bits en autant de clefs que nécessaire, clefs qui vont servir à établir un canal intègre, confidentiel et authentifié.

Comme on l'a vu en introduction à la sous-section 1.2.3, le caractère authentifié du canal est une conséquence à la fois du caractère intègre du canal et du fait que l'échange de clefs est authentifié. Pour établir un canal sécurisé, il suffit donc qu'il assure la confidentialité et l'intégrité.

Selon l'algorithme de chiffrement et l'algorithme d'authentification de message choisis et suivant le mode opératoire utilisé, la sécurité en intégrité et en confidentialité du protocole varie. Les constructions doivent donc faire l'objet d'une étude pour évaluer leur sécurité et c'est ce que nous allons faire maintenant. Pour être plus précis, nous allons nous intéresser à un mode de chiffrement authentifié en particulier, appelé CCM. Celui-ci permet non seulement de chiffrer et authentifier des messages, mais également d'adjoindre, lors d'un échange, des données qui ne seront pas chiffrées mais uniquement authentifiées et que l'on appellera les « données associées ». Il a été proposé par Doug Whiting, Russ Housley et Niels Ferguson en 2002. Les initiales CCM veulent dire Compteur et Cbc-Mac, ce qui signifie que le mode de chiffrement utilisé est le mode compteur et que l'algorithme d'authentification utilisé est le CBC-MAC. CCM est intéressant car ces deux algorithmes de cryptographie symétrique sont très populaires et sont implémentés dans un grand nombre de produits. CCM peut donc être construit en utilisant des fonctions préexistantes. Il est utilisé dans beaucoup de standards de réseau sans fil, tels que IEEE 802.11 [WHF02] (WiFi), IEEE 802.15.40 (Wireless Personal Area Network/ZigBee), dans des standards de l'internet comme la RFC 3610 et la RFC 4309 et enfin conseillé par le NIST SP 800-38C [Dwo02].

La sécurité de CCM s'appuie sur un format particulier des entrées. Assurer ainsi la sécurité a déjà été fait par le passé. Nous pouvons citer à titre d'exemple la transformation de Merkle-Damgård ou le CBC-MAC. La transformation de Merkle-Damgård consiste à ajouter la taille du message à la fin de ce dernier et permet de transformer une fonction de compression résistante aux collisions en une fonction de hachage résistantes aux collisions. Pour le CBC-MAC on ajoute la longueur du message dans le premier bloc, ce qui permet de transformer l'ensemble des messages en un ensemble sans préfixe et donc d'éviter les attaques classiques contre le CBC-MAC.

Jonsson a prouvé [Jon03] que CCM est sûr en confidentialité et en intégrité tant que moins de $2^{n/2}$ requêtes de chiffrement ont été faites. Selon Jonsson, la confidentialité de CCM ne peut être prouvée sûre au-delà du paradoxe des anniversaires. Par contre, il est possible qu'il soit sûr en authentification au-delà. Jonsson explique que si la sécurité est prouvée jusqu'à $2^{n/2}$ requêtes, il n'existe aucune attaque connue qui atteigne cette borne et il conjecture une sécurité en $\mathcal{O}(2^n)$ requêtes de chiffrement, conjecture qui n'a pas été démontrée jusqu'à présent.

Dans la même voie de recherche, un autre schéma de chiffrement authentifié avec des données authentifiées associées a été proposé par Bellare, Rogaway et Wagner en 2004 : AEX [BRW04]. Il a été construit en réaction à CCM, jugé trop complexes par les auteurs d'AEX. Ce dernier a de nombreux points communs avec CCM mais évite certains de ses défauts. Comme CCM, il est prouvé sûr [BRW04] jusqu'à $2^{n/2}$ requêtes. Cette borne est classique et un schéma de chiffrement dont la sécurité l'atteint est habituellement considéré comme sûr.

CCM a été critiqué par Rogaway et Wagner [RW03] qui ont mis en avant plusieurs raisons pour lesquelles l'efficacité n'est pas optimale. Entre autres, ils reprochent que CCM ne puisse être exécuté en ligne et qu'il casse l'alignement des mots. Le problème principal de CCM est qu'il ne peut pas être exécuté en ligne, puisque l'émetteur doit connaître la longueur du message et la longueur des données associées avant de commencer à le traiter. La seconde critique concerne uniquement les données associées. Enfin, une dernière remarque peut être faite sur la manière de choisir les *nonces*. Un *nonce* (nous utilisons ici le mot anglais) est une chaîne de bits choisie et utilisée par l'algorithme de chiffrement authentifié puis rendue publique. Pour que la preuve de sécurité de CCM soit valide, il faut que l'on n'emploie pas deux fois le même *nonce* pour chiffrer deux messages différents. Il faut donc faire attention à la manière dont on les choisit.

En pratique, cette restriction peut être facilement respectée lorsque l'échange ne fait intervenir que deux parties, par contre il est nettement plus difficile de s'y conformer lorsque plus de trois parties participent à la communication. Pour améliorer l'efficacité du protocole ou pour faciliter son implémentation, il peut donc être tentant de s'affranchir de certaines des restrictions imposées par CCM, comme par exemple choisir les *nonces* au hasard, sans s'inquiéter de savoir s'ils ont déjà été utilisés pour chiffrer ou pas. Par conséquent, nous avons examiné si on pouvait modifier le protocole d'origine sans réduire la sécurité conjecturée de CCM, ou si au contraire toute modification créait des faiblesses dans le protocole.

Nous allons donc essayer de justifier pourquoi on ne pouvait pas autoriser des *nonces* qui puissent se répéter et pourquoi il est important de mettre la longueur des données associées et celle du message au début de ce dernier. Pour cela nous allons montrer que, si ceci n'est pas fait, il existe alors des attaques contre CCM.

Par ailleurs, il est difficile de prouver la conjecture de Jonsson selon laquelle la sécurité en authentification de CCM est supérieure à 2^n. Même si la borne de $2^{n/2}$ est habituellement considérée suffisante, il peut être intéressant de transformer CCM si cela permet d'obtenir une sécurité *prouvée* en 2^n, tant que le changement opéré reste léger. C'est dans cette optique que nous allons proposer quelques modifications simples, aussi peu que possible, permettant d'obtenir une sécurité bien meilleure que celle de Jonsson. Ces attaques et les modifications proposées ont donné lieu à une publication [FMVZ08].

4.2 Description du modèle de sécurité

Nous commençons par donner les définitions précises des notions d'intégrité et de confidentialité pour un algorithme symétrique.

4.2.1 Intégrité

La notion de sécurité que nous utiliserons pour analyser la sécurité en authentification d'un schéma de chiffrement authentifié symétrique P est l'intégrité du texte chiffré (désignée par INT-CTXT). Formellement, dans un jeu d'intégrité, l'adversaire $\mathcal{A}$ a accès à un oracle de chiffrement $\mathcal{E}(\cdot)$ et à un oracle de vérification $\mathcal{VO}(\cdot)$ auxquels il peut faire les requêtes de son choix. L'oracle de chiffrement prend en entrée des textes clairs et renvoie un chiffré correspondant. L'adversaire envoie à l'oracle de vérification des messages chiffrés, aussi appelés tentatives de contrefaçon, et l'oracle répond par 1 si le message chiffré est valide, par 0 dans le cas contraire. Le but de l'adversaire est de générer un chiffré valide (c'est-à-dire qui est accepté par l'oracle de vérification) et qui est différent de tous les chiffrés qui ont été générés précédemment par l'oracle de chiffrement. On remarquera que l'adversaire peut envoyer plusieurs requêtes à l'oracle de vérification. La probabilité de succès de $\mathcal{A}$ est :

$$\mathsf{Succ}_P^{\texttt{int-ctxt}}(\mathcal{A}) = \Pr[\mathcal{A}^{\mathcal{E}(\cdot),\mathcal{VO}(\cdot)} \Rightarrow C \colon \mathcal{VO}(C) \Rightarrow 1],$$

la probabilité portant sur les aléas de l'adversaire, sur les clefs utilisées pour l'algorithme de chiffrement authentifié et éventuellement sur les aléas utilisés pour répondre aux requêtes de chiffrement.

4.2.2 Confidentialité

La notion de sécurité que nous utiliserons pour analyser la sécurité en confidentialité d'un schéma de chiffrement authentifié symétrique P est l'indistinguabilité dans le cadre d'une attaque

à messages clairs choisis (désignée par IND-CPA). Formellement, dans le jeu de la confidentialité, un adversaire $\mathcal{A}$ a accès à un oracle de chiffrement $\mathcal{E}(\cdot)$ auquel il peut faire des requêtes de la forme (M_0, M_1) où M_0 et M_1 sont deux messages de son choix. Au début du jeu, l'oracle choisit un bit b et chiffre toujours le message M_b. Le but de l'adversaire est de deviner le bit b. L'indistinguabilité est ici définie dans le modèle « droite ou gauche » ou LoR pour *Left or Right*. Ce dernier a été introduit et prouvé comme étant équivalent au modèle le plus fort dans [BDJR97]. L'avantage de $\mathcal{A}$ vaut :

$$\mathsf{adv}_P^{\texttt{ind-cpa}}(\mathcal{A}) = \left|\Pr[\mathcal{A}^{\mathcal{E}(\cdot)} \Rightarrow 1 | b = 1] - \Pr[\mathcal{A}^{\mathcal{E}(\cdot)} \Rightarrow 1 | b = 0]\right|,$$

les probabilités portant sur les aléas de l'adversaire, sur les clefs utilisées pour l'algorithme de chiffrement authentifié et éventuellement sur les aléas utilisés pour répondre aux requêtes de chiffrement.

4.3 CBC-MAC et mode compteur

Avant de nous intéresser à CCM lui-même nous donnons succinctement la description de l'algorithme de messages CBC-MAC employé dans CCM pour l'authentification et du mode compteur utilisé pour le chiffrement. Nous présentons également quelques résultats de sécurité concernant ces deux algorithmes qui seront utiles pour l'analyse de CCM.

4.3.1 CBC-MAC

Soient E un algorithme de chiffrement par bloc de $\{0,1\}^n$ et K une clef pour E. Soit $M = (M_1, \ldots, M_m)$ un message constitué de m blocs de n bits. Le CBC-MAC de M est $CBC(M) = T_{m+1}$ où $T_1 = 0^n$ et $T_{i+1} = E(K, M_i \oplus T_i)$ pour tout $1 \leq i \leq m$.

Nous présentons maintenant un résultat prouvé dans [BPR05] qui majore l'avantage d'un attaquant contre un adversaire pf-PRF contre CBC-MAC (on rappelle que le pf dans pf-PRF signifie que les requêtes générées par l'adversaire forment un ensemble sans préfixe).

Théorème 4.1 (BPR)**.** *Soit $\mathcal{A}$ un adversaire PRF sans préfixe contre CBC-MAC avec des blocs de n bits, $\mathcal{A}$ est autorisé à faire au plus $q \geq 2$ requêtes d'au plus s blocs et a un temps de calcul d'au plus T. On a alors :*

$$\mathsf{adv}_{CBC\text{-}MAC}^{\texttt{pf-prf}}(\mathcal{A}) \leq \frac{sq^2}{2^n}\left(12 + \frac{64s^3}{2^n}\right).$$

4.3.2 Mode compteur

Soient E un algorithme de chiffrement par blocs de $\{0,1\}^n$ et K une clef pour E. Soient $M = (M_0, \ldots, M_m)$ un message constitué de $m+1$ blocs de n bits et $A_0, \ldots, A_m$ une suite de $m+1$ blocs de n bits tels que $A_i = A_0 \oplus i$. Cette suite doit être différente à chaque nouveau chiffrement, et plus précisément, deux suites utilisées pour chiffrer deux messages différents ne doivent pas avoir ne serait ce qu'un bloc en commun. On obtient le chiffré $C = (C_0, \ldots, C_m)$ de M par le mode compteur en calculant $C_i = E(K, A_i) \oplus M_i$ pour tout $0 \leq i \leq m$.

Nous présentons maintenant un résultat évaluant la confidentialité du mode compteur en utilisant une fonction pseudo-aléatoire. Ce résultat de sécurité a été établi dans [BDJR97] :

Théorème 4.2 (BDJR)**.** *Soit F une famille de PRF de $\{0,1\}^n$ dans $\{0,1\}^n$. Alors, pour tout adversaire $\mathcal{A}$ contre la confidentialité du mode CTR, avec un temps de calcul T, et qui peut faire*

au plus q_e requêtes de chiffrement d'au plus s blocs chacune, il existe un adversaire $\mathcal{A}'$ contre F, avec un temps de calcul d'au plus T et qui fait au plus sq_e requêtes de chiffrement, tel que :

$$\mathsf{adv}^{\mathtt{ind\text{-}cpa}}_{CTR}(\mathcal{A}) \leq 2\mathsf{adv}^{\mathtt{prf}}_{F}(\mathcal{A}').$$

4.4 Description de CCM

Nous allons maintenant décrire le cadre général des chiffrements authentifiant puis le mode de chiffrement authentifié appelé CCM et le format de ses diverses entrées. Dans [SPC04], le NIST donne des recommandations concernant différents choix que l'on peut avoir à faire pour implémenter CCM : utiliser une clef unique pour à la fois la chaîne CBC et le mode compteur, des *nonces* qui ne doivent pas se répéter... Certaines de ces restrictions peuvent être ignorées sans problème pour la sécurité, alors que d'autres sont nécessaires. À la fin de ce chapitre nous discutons ces choix.

4.4.1 Chiffrement authentifiant

Pour réaliser un algorithme de chiffrement authentifiant, la méthode la plus intuitive est d'utiliser un algorithme de chiffrement combiné à un algorithme d'authentification (aussi appelé MAC pour *Message Authentication Code*). Il reste ensuite à les articuler entre eux, c'est-à-dire à choisir un mode opératoire. On distingue trois modes opératoires : le mode authentifier-puis-chiffrer, le mode authentifier-et-chiffrer, le mode chiffrer-puis-authentifier. Tous trois ont été utilisés en pratique. Le premier consiste à calculer ce que l'on appelle le *mac* du message (c'est-à-dire la sortie de l'algorithme d'authentification lorsque le message est donné en entrée), à le concaténer au message et à chiffrer le tout. Le second consiste à calculer le mac du message, à chiffrer le message et à renvoyer les deux résultats. Enfin, le dernier mode consiste à chiffrer le message, à calculer le mac du chiffré puis de renvoyer chiffré et mac. Krawczyk [Kra01] a étudié en détail les trois modes opératoires. Il a montré que sous des hypothèses raisonnables sur le MAC et sur le chiffrement, le mode chiffrer-puis-authentifier est sûr. Au contraire, il a exhibé des contre-exemples qui prouvent que les deux autres modes opératoires ne sont pas sûrs de manière générique. Cela ne signifie pas nécessairement que l'on ne peut pas les employer pour concevoir des algorithmes qui reposent sur eux, ni que les algorithmes déjà existants ne sont pas sûrs. Par contre, cela implique qu'il faut étudier en particulier chacune des constructions basée sur un de ces deux modes, pour vérifier et prouver sa solidité.

L'avantage du schéma chiffrer-puis-authentifier est qu'il invite à la modularité : je peux utiliser n'importe quel algorithme de chiffrement ou d'authentification sûr. Il laisse donc toute liberté au concepteur. Il possède deux inconvénients, le premier étant justement qu'il laisse toute liberté à son concepteur. De nombreuses erreurs de conception ont été commises par le passé et les organes de normalisation ont senti la nécessité de donner la description précise d'un algorithme et prouvé sûr. Son second inconvénient est qu'il nécessite deux passes : une pour calculer le chiffré, une autre pour calculer le mac. Pour cette raison, des algorithmes spécifiques de chiffrement authentifiant à une passe ont été proposés [GD01]. Cependant ils ont tous été brevetés, ce qui entrave leur large utilisation. C'est en partie pour cette raison que les organes de normalisation ont favorisé la conception d'un algorithme en deux passes comme CCM : on avait besoin d'un mode contraint non breveté. Cependant CCM est un mode authentifier-puis-chiffrer, c'est pourquoi il a fallu adapter une preuve de sécurité.

Enfin, avec l'usage on s'est rendu compte que certaines données, comme les headers par exemple, étaient connues de l'attaquant et n'avaient pas besoin d'être chiffrées. Aussi, cela a

amené à se pencher vers des modes permettant d'associer des données qui sont uniquement authentifiées. C'est ce qu'autorise CCM.

4.4.2 Notations

Dans ce chapitre, pour décrire les détails de la construction de CCM, nous allons utiliser les notations suivantes :

- pour toute chaîne de bits ou pour tout entier x, $|x|$ désigne sa taille en bits, $|x|_8 = \left\lceil \frac{|x|}{8} \right\rceil$ sa taille en octets et $[x]_s$ désigne la représentation binaire de x sur s bits ;
- E est un algorithme de chiffrement par blocs pour des blocs de $n = 128$ bits et avec une clef de k bits ;
- M est un texte clair, constitué de blocs de n bits désignés par $M_1, \ldots, M_{m-1}$ et un dernier bloc M_m d'au plus n bits ;
- les données associées (ce sont des données qui sont authentifiées mais pas chiffrées) sont désignées par $D_1, \ldots, D_a$ et sont constituées de $a-1$ blocs de n bits et d'un bloc d'au plus n bits ;
- le texte chiffré C est constitué de $m+1$ blocs $C_0, C_1, \ldots, C_m$ où C_i fait n bits de long pour $0 \leq i \leq m-1$ et C_m fait au plus n bits ;
- $B = B_0, B_1, \ldots, B_r$ est une entrée constituée de r blocs de n bits de long et utilisée pour le calcul du CBC-MAC, B_0 étant appelée la valeur pré-initiale ;
- $A_0, A_1, \ldots, A_m$ sont des entrées pour le mode compteur ;
- le *nonce*, utilisé pour générer la valeur pré-initiale B_0 et toutes les valeurs de compteur $A_0, A_1, \ldots, A_m$, est noté N. Ce *nonce* fait ℓ bits de long, avec $7 \leq \ell/8 \leq 13$ (ℓ doit être divisible par 8) ;
- q est un entier tel que $2 \leq q \leq 8$ et $q + \ell/8 = 15$, Q est une chaîne de q octets qui désigne la représentation de la taille en octets du message M, représentation faite sur q octets, c'est-à-dire que $Q = \left[|M|_8\right]_{8q}$;
- t désigne la longueur en bits du mac, longueur qui doit être divisible par 16, et $4 \leq t/8 \leq 16$.

4.4.3 Description du mode opératoire

Le mode CCM est un mode authentifier-puis-chiffrer où l'authentification est faire par un CBC-MAC et le chiffrement par un mode compteur. Ce mode utilise le même algorithme de chiffrement par blocs E dans la chaîne CBC et dans le mode compteur. La longueur du bloc est égale à $n = 128$ bits. Nous désignons par K la clef utilisée pour le CBC-MAC et par K' la clef utilisée dans le mode compteur. Le choix de K et K' est discuté dans la sous-section 4.4.5.

Soit $M = M_1 \| M_2 \| \ldots \| M_m$ un message clair et soit $D = D_1 \| D_2 \| \ldots \| D_a$ les données associées que l'on va uniquement authentifier et non chiffrer.

Tout d'abord, l'algorithme de chiffrement choisit un *nonce* N sur ℓ bits. Le *nonce* va être utilisé pour dériver la valeur pré-initiale pour le CBC-MAC et les blocs compteur.

Dans une première étape, les données associées et les blocs du texte clair sont authentifiés en déterminant leur valeur CBC-MAC. Cependant ce calcul est relativement différent de l'évaluation classique d'un CBC-MAC : il authentifie une entrée formatée $B = B_0 \| \ldots \| B_r$ générée à partir de N, M et D. Le format précis de B est décrit dans la sous-section 4.4.4. La valeur pré-initiale B_0 qui fait un bloc de long est traitée comme le premier bloc de message dans la chaîne CBC. Sa propriété principale est de contenir la longueur du mac, la longueur du texte clair et la longueur du *nonce*. Le CBC-MAC est une chaîne CBC simple. Sa sortie, notée T, est longue de t bits, avec $32 \leq t \leq n$.

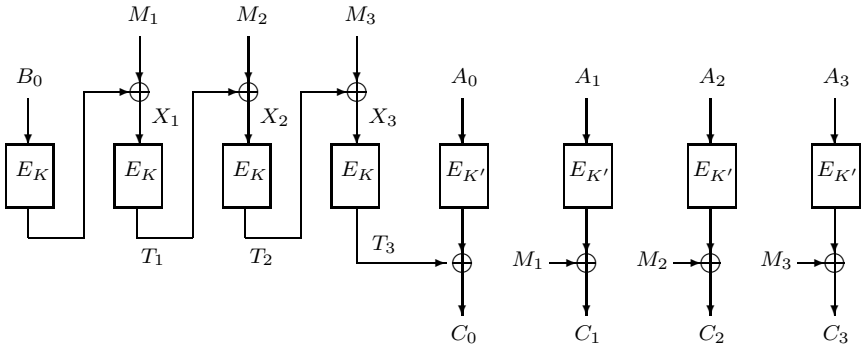


FIG. 4.1 – Le mode de chiffrement authentifié CCM.

Dans un second temps, la valeur T du mac et le message clair M sont concaténés et chiffrés avec un mode compteur. Les entrées pour le mode compteur sont des blocs de n bits, notés $A_0 \| A_1 \| \ldots \| A_m$. Leur format est décrit dans la sous-section 4.4.4. Brièvement, chacun contient un marqueur, la valeur du *nonce* et l'indice du bloc du texte clair. Le mac T est chiffré en calculant $C_0 = \mathsf{lsb}_t\left(E_{K'}(A_0)\right) \oplus T$ et tous les blocs du texte clair M_i, $1 \leq i \leq m$, sont chiffrés en calculant $C_i = E_{K'}(A_i) \oplus M_i$.

Le chiffré $C = C_0 \| C_1 \| \ldots \| C_m$ et les données associées D sont alors transmis. La valeur du *nonce* est transmise si nécessaire (dans le cas où elle ne serait pas synchronisée).

La figure 4.1 décrit le mode CCM appliqué à un message long de 3 blocs et auquel aucune donnée n'est associée.

Le processus de déchiffrement consiste tout d'abord à déchiffrer C avec le mode compteur et ensuite à calculer le mac T' à partir d'une entrée formatée B, générée grâce au *nonce* N, aux données associées et au message clair que l'on vient d'obtenir. Si le mac est valide, le message est renvoyé. Dans le cas contraire, un message d'erreur est envoyé. Pour une description des algorithmes de chiffrement et de déchiffrement en pseudo-code, voir les algorithmes 1 et 2.

4.4.4 Format des entrées

Les spécifications [SPC04] décrivent précisément le format pour les différentes entrées.

La chaîne CBC-MAC utilise une entrée formatée $B = B_0, \ldots, B_r$. La valeur pré-initiale B_0 sur n bits est déterminée par le *nonce* sur ℓ bits et par diverses autres informations. Le premier octet est un marqueur contenant un bit pour des extensions futures du mode, un bit indiquant si des données associées sont présentes ou pas, trois bits indiquant la taille en octets de la valeur de mac (ces trois bits doivent nécessairement être différents de 000) et trois bits pour la taille en octets de la représentation binaire de la taille en octets du message M. Les octets restant contiennent la valeur du *nonce* suivie par la valeur Q, la représentation en bits de la taille en octets de M. La figure 4.2 présente le format de B_0.

On remarquera qu'une collision sur B_0 a lieu si et seulement si les trois événements suivants sont simultanément vrais : une collision entre les *nonces* a lieu, si des données sont associées dans les deux cas ou non associées dans les deux cas et si les messages sont de la même taille.

S'il y a des données associées D à authentifier alors $B_1 \| \ldots \| B_u$ est la concaténation d'un encodage particulier de la taille de D (voir [SPC04] pour plus de détails), de D elles-mêmes et d'autant de '0' que nécessaire pour obtenir une chaîne de bits qui peut être divisée en blocs de

Algorithme 1 Chiffrement de CCM(M, D)

Choisir N,
fonction AUTHENTIFICATION(N, D, M)
    Générer $B_0, \ldots, B_r$ à partir de N, M et D.
    $X_0 \leftarrow E(B_0)$
    pour $i \leftarrow 1,r$ **faire**
        $X_i \leftarrow E(B_i \oplus X_{i-1})$
    fin de pour
    $T \leftarrow \mathsf{lsb}_t(X_r)$
    renvoyer T
fin de fonction
fonction CHIFFREMENT(N, D, T, M)
    Générer $A_0, \ldots, A_m$ à partir de N.
    $c_0 \leftarrow \mathsf{lsb}_t(E(A_0)) \oplus T$
    pour $i \leftarrow 1,m$ **faire**
        $C_i \leftarrow E(A_i) \oplus M_i$
    fin de pour
    $C \leftarrow C_0, \ldots, C_m$
    renvoyer (N, C, D)
fin de fonction

Algorithme 2 Déchiffrement de CCM(N, C, D)

Générer $A_0, \ldots, A_m$ à partir de N.
$T \leftarrow \mathsf{lsb}_t(E(A_0)) \oplus C_0$
pour $i \leftarrow 1,m$ **faire**
    $M_i \leftarrow E(A_i) \oplus C_i$
fin de pour
$M \leftarrow M_0, \ldots, M_m$
Générer $B_0, \ldots, B_r$ à partir de N, M et D.
$X_0 \leftarrow E(B_0)$
pour $i \leftarrow 1,r$ **faire**
    $X_i \leftarrow E(B_i \oplus X_{i-1})$
fin de pour
si $(T == \mathsf{lsb}_t(X_r))$ **alors renvoyer** (M, D)
sinon renvoyer $\perp$
fin de si

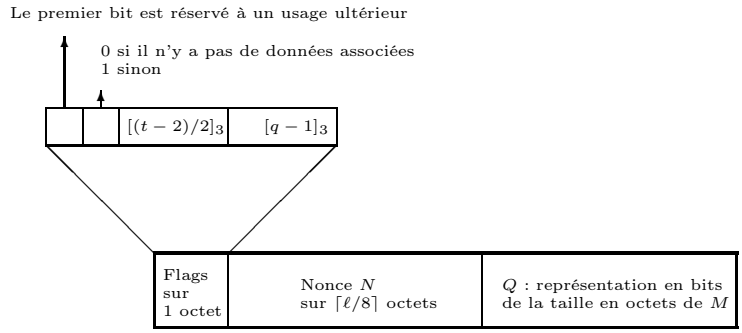


FIG. 4.2 – Le format des la valeur pré-initiale B_0 pour le CBC-MAC.

n bits. S'il n'y a pas de données associées, $B_1 \| \ldots \| B_u$ est la chaîne vide. Soit $B_{u+1} \| \ldots \| B_r$ la concaténation de M suivi d'autant de '0' que nécessaire pour obtenir une chaîne de bits qui peut être divisée en blocs de n bits. On remarquera que le format de B est fait de telle sorte que l'ensemble de toutes les chaînes de bits B possibles est sans préfixe.

Enfin, les entrées pour le mode compteur $A_0, A_1, \ldots, A_m$ sont formatées ainsi : le premier octet contient un marqueur (2 bits réservés à un usage futur, 3 bits fixés à 000 et trois bits contiennant la représentation binaire de $q-1$). Le reste contient le *nonce*, déjà utilisé pour le format de B_0, et le numéro du bloc.

4.4.5 Conseils du NIST

Les spécifications de CCM [SPC04] donnent des indications pour guider les choix d'implémentation. Bien sûr, CCM doit être utilisé avec l'AES dont les longueurs de clef sont égales à 128, 192, ou 256 bits. La taille du bloc est, comme attendu, de 128 bits.

La spécification de CCM fournit aussi l'obligation d'utiliser la même clef pour le CBC-MAC et le mode compteur. Un tel choix est bien sûr discutable puisqu'il va à l'encontre du sens commun qui veut que l'on utilise des clefs différentes pour des usages différents. Cependant, la preuve de sécurité donnée par Jonsson dans [Jon03] est adaptée à ce cas et nous dit que CCM assure la confidentialité et l'authentification des messages échangés tant que moins de $2^{n/2}$ appels à l'algorithme de chiffrement par bloc ont été faits. Par conséquent, choisir la même clef pour les deux modes combinés dans CCM n'est pas une erreur vis à vis de la sécurité puisque la borne de sécurité est très proche de celle obtenue pour de nombreux modes de chiffrement authentifié [RBBK01, Jut01].

Cependant, si ce choix est assez compréhensible dans le cas où l'on obtient les mêmes résultats de sécurité avec une ou deux clefs, il devient maladroit si la sécurité avec deux clefs est bien meilleure que celle avec une seule clef. Notre version modifiée de CCM permet d'obtenir une bien meilleure sécurité avec deux clefs. C'est pourquoi dans la section 4.6, nous nous concentrons sur les résultats de sécurité de notre version modifiée dans le cas de *nonces* qui ne se répètent pas et avec deux clefs différentes.

Une autre obligation faite par les spécifications de CCM concernent le choix des *nonces*. Il est explicitement défini que *« les blocs du compteur doivent être distincts au sein d'une même invocation et à travers toutes les autres invocations du mode CTR pour une clef donnée »*. Cette obligation est remplie en imposant des *nonces* qui ne se répètent pas. Elle est largement plus compréhensible que la précédente : en effet, une collision sur les *nonces* peut souvent être exploitée pour contrefaire un chiffré valide ou pour attaquer la confidentialité du schéma. Cependant, en

pratique, il est difficile de maintenir des *nonces* qui ne se répètent pas, particulièrement dans un scénario où 3 utilisateurs ou plus partagent la même clef pour utiliser CCM. Par conséquent, il peut être intéressant de regarder attentivement la sécurité de CCM quand les *nonces* sont choisis aléatoirement et donc qu'une collision entre deux *nonces* peut se produire. Ceci est fait dans la section 4.5.

4.5 Attaques contre CCM en relâchant des hypothèses

Dans ce chapitre, nous considérons que les *nonces* utilisés pour CCM sont choisis uniformément au hasard dans $\{0,1\}^\ell$. Dans ce cas, une collision entre deux valeurs des *nonces* est possible et peut être exploitée par un adversaire pour contrefaire un chiffré valide. Bien sûr, la confidentialité ne peut plus être assurée dès qu'il y a collision entre deux *nonces*. En effet, si une telle collision se produit, l'adversaire peut aisément distinguer quel message est chiffré et donc casser la sécurité sémantique du schéma. Cependant, réussir à casser la confidentialité n'implique pas nécessairement une faiblesse de l'intégrité, même si, quand l'un est compromis, l'autre l'est souvent. Ainsi la technique utilisée ici pour contrefaire un chiffré valide est complètement différente de celle utilisée pour casser la sécurité sémantique dans le cas où il y a une collision entre les *nonces*.

On remarquera que les attaques suivantes s'appliquent tant à la version originelle de CCM qu'à la version modifiée présentée dans la section 4.6, tant que les *nonces* sont choisis uniformément aléatoirement. Nous concentrons notre analyse sur CCM car il est standardisé, mais les mêmes conclusions peuvent être établies pour la version modifiée de CCM.

Enfin, nous soulignons que dans ces attaques l'adversaire peut faire autant de requêtes de chiffrement qu'il veut mais que les *nonces* utilisés pour chiffrer sont choisis par l'oracle de chiffrement (nous verrons que ce choix se fera aléatoirement selon la distribution uniforme sur l'ensemble des *nonces*).

4.5.1 Attaque générique

Nous présentons dans cette section une attaque générique contre les versions originale et modifiée de CCM, en supposant uniquement que les *nonces* sont choisis aléatoirement selon la distribution uniforme sur l'ensemble des *nonces*. Nous utiliserons de façon cruciale le fait qu'une collision entre les *nonces* peut se produire. La complexité de cette attaque est de $\mathcal{O}(2^\ell + 2^{(t+\ell)/2})$ requêtes de chiffrement et d'*une* requête de vérification, où t est la taille en bits du mac et ℓ est la taille du *nonce* en bits, et sa probabilité de succès est proche de 1. Soient M_3 un bloc de message et $(M_1^i, M_2^i)_i$ un ensemble de $2^{(t+\ell)/2}$ messages de 2 blocs. L'adversaire fait les $2^{(t+\ell)/2}$ requêtes de chiffrement (M_1^i, M_2^i, M_3) (on remarquera que le dernier bloc est toujours le même) et reçoit la réponse de l'oracle $N^i, (C_0^i, C_1^i, C_2^i, C_3^i)$. Avec grande probabilité, il y a deux indices différents i et k tels que $N^i = N^k$ et $C_0^i = C_0^k$. Puisque les *nonces* sont identiques pour ces deux indices, les valeurs de compteur utilisées pour le chiffrement sont identiques et par conséquent, puisque les chiffrés du mac sont identiques, les deux mac pour ces deux indices sont identiques. Puisque le dernier bloc du message est le même, cela signifie que la collision apparaît en fait avant le traitement du dernier bloc, et donc $CBC(B_0\|M_1^i\|M_2^i\|M_3') = CBC(B_0\|M_1^k\|M_2^k\|M_3')$ pour n'importe quel bloc M_3' de n bits (on fait remarquer qu'une collision entre les *nonces* implique une collision entre les valeurs B_0 puisque les messages ont la même taille). Soit $M_3' \neq M_3$, on répète $\mathcal{O}(2^\ell)$ fois la requête de chiffrement de $M_1^i\|M_2^i\|M_3'$ jusqu'à ce que la réponse fasse intervenir le *nonce* N^i. Notons alors C_0, C_1, C_2, C_3 le chiffré correspondant et soient $C_0' = C_0$, $C_1' = C_1 \oplus M_1^i \oplus M_1^k$, $C_2' = C_2 \oplus M_2^i \oplus M_2^k$, et $C_3' = C_3$. Le chiffré $N^k, (C_0', C_1', C_2', C_3')$ est un

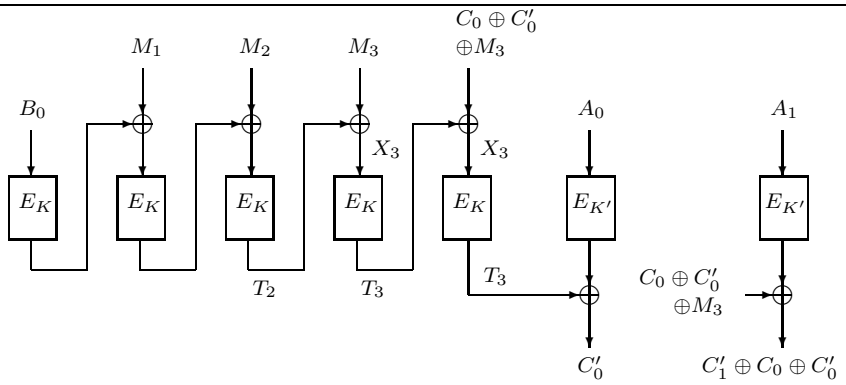


FIG. 4.3 – Tentative de contrefaçon : $C_0 \oplus C'_0$ égale $T_2 \oplus T_3$.

chiffré valide pour le message $M_1^k \| M_2^k \| M'_3$ (message qui n'a pas encore était demandé à l'oracle de chiffrement).

On remarquera que dans le cas où $t \leq \ell$, cette attaque nécessite $\mathcal{O}(2^\ell)$ requêtes de chiffrement, 1 requête de vérification, chacune constituée d'au plus 3 blocs. Par conséquent, si le théorème 4.4 s'appliquait, la probabilité de succès d'un tel adversaire serait majorée par $2^{9-n} + 2^{-t}$, alors qu'elle est presque égale à 1. Si cette attaque n'est pas applicable en pratique, elle illustre le fait que choisir les *nonces* de façon aléatoire décroît fortement la sécurité de CCM.

4.5.2 Quand la taille des données associées n'est pas précisée

Dans l'attaque précédente nous avons relâché la principale contrainte de CCM : il ne doit pas y avoir de collision entre les *nonces*. Dans les deux prochaines attaques, nous allons également autoriser les collisions entre les *nonces* et en plus nous allons supposer que les entrées formatées ne forment pas un ensemble sans préfixe. Nous prouvons que, dans ce cas, des attaques plus fortes que la précédente peuvent être montées.

Nous rappelons que dans CCM, B_0 dépend de la taille du message, mais pas de la taille des données associées (cependant s'il y a des données associées, le premier bit du marqueur est fixé à 1). S'il y a des données associées, leur longueur peut être encodée au plus sur les 10 premiers blocs de B. Puisque la longueur des données associées doit être connue pour l'authentification, l'authentification ne peut être faite en ligne ce qui réduit l'efficacité de CCM. Pour l'attaque suivante, nous relâchons cette contrainte, en supposant que la longueur des données associées n'est pas concaténée *avant* les données associées elles-mêmes (elle pourrait être encodée et rajoutée après les données associées, cependant pour simplifier l'exposition faite ici, nous l'omettons tout simplement). L'attaquant va utiliser cette remarque pour contrefaire un chiffré valide à partir de $2^{\ell/2}$ requêtes de chiffrement.

Pour cette attaque les données associées sont toujours utilisées, le premier bit du marqueur présent dans B_0 est donc toujours égal à 1. Cette attaque exploite certaines relations qui peuvent être déduites de la collision sur les valeurs des *nonces* : l'attaquant fait des requêtes à l'oracle de chiffrement avec des messages de 2 ou 3 blocs. Dans les deux cas, seul le dernier bloc est chiffré. Il collecte par conséquent les chiffrés $(N_i, C_0^i \| C_1^i)$ correspondant à l'authentification de $M_1 \| M_2$ et au chiffrement de M_2 sous différents *nonces*, et les chiffrés $(N'_j, C_0^j \| C_1^j)$ correspondant à l'authentification de $M_1 \| M_2 \| M_3$ et au chiffrement de M_2 sous différents *nonces*. D'après le paradoxe des anniversaires, après $2^{\ell/2}$ requêtes, il y a une collision entre deux *nonces*, un utilisé pour chiffrer une requête du premier ensemble, le second utilisé pour chiffrer une requête du deuxième ensemble. Soient i et j les indices tels que $N_i = N_j$. Puisque les textes clairs à

authentifier et à chiffrer sont de la même taille dans les deux cas, nous pouvons également avoir $B_0^i = B_0^j$. Pour simplifier les notations, les chiffrés correspondants seront désignés par $(N, C_0 \| C_1)$ et $(N, C'_0 \| C'_1 \| C'_2)$.

L'attaquant peut maintenant calculer la valeur de

$$C_0 \oplus C'_0 = T \oplus T'$$

où T est la valeur de sortie du CBC-MAC calculé pour le premier message et T' est la valeur de sortie du CBC-MAC calculé pour le second. On peut par conséquent contrefaire un chiffré valide pour le message $M_1 \| M_2 \| M_3 \| M_3 \oplus C_0 \oplus C'_0$ où seul le dernier bloc est chiffré. Le chiffré correspondant est par conséquent $(N, C'_0 \| C'_1 \oplus C_0 \oplus C'_0)$ et les données associées sont $M_1 \| M_2 \| M_3$. La figure 4.3 résume cette contrefaçon.

La complexité de l'attaque est de $\mathcal{O}(2^{\ell/2})$ requêtes de chiffrement. Puisque $\ell \leq n$, cela signifie que relâcher la contrainte sur le format particulier des données authentifiées amène à des attaques meilleures que la borne du paradoxe des anniversaires prouvée par Jonsson.

4.5.3 Quand les *nonces* sont aléatoires et les entrées non formatées

Enfin, dans la prochaine attaque, nous considérons le cas où les *nonces* sont aléatoires et la taille du message n'est pas mise dans le bloc pré-initial B_0. De cette manière le calcul de CCM peut être faite en ligne, cependant nous prouvons que dans ce cas encore, même sans donnée authentifiée supplémentaire, la sécurité de CCM baisse.

L'attaquant va faire 3 sortes de requêtes. Les requêtes de la première sorte sont composées d'un message formé d'un seul bloc noté M_1. Ce bloc est le même pour toutes les requêtes. Les requêtes de la deuxième sorte sont composées de deux blocs, $M_1 \| M_2$ où M_1 est le même bloc que celui choisi pour les requêtes de la première sorte. Enfin, dans le troisième ensemble de requêtes, les messages sont composées de 3 blocs $M_1 \| M_2 \| M_3$ où M_1 et M_2 sont identiques aux blocs utilisés pour les requêtes de deuxième type. L'attaquant fait des requêtes à l'oracle de chiffrement pour ces messages jusqu'à ce qu'une 3-collision se produise sur les *nonces* utilisés pour le chiffrement, c'est-à-dire qu'il existe un même *nonce* utilisé pour chiffrer à la fois M_1, $M_1 \| M_2$ et $M_1 \| M_2 \| M_3$. Pour obtenir une telle 3-collision, en moyenne $2^{2\ell/3}$ requêtes par message sont nécessaires. Une fois la 3-collision obtenue, il existe donc trois indices i, j, k tels que : $N_i = N_j = N_k$. On a donc $B_0^i = B_0^j = B_0^k$ et $A_0^i = A_0^j = A_0^k$ (puisque B_0 ne dépend plus de la longueur du message). L'attaquant est maintenant capable de contrefaire un chiffré valide. On désigne par $(N, C_0^1 \| C_1^1)$ le chiffré correspondant au message clair d'un bloc, par $(N, C_0^2 \| C_1^2 \| C_2^2)$ le chiffré correspondant au message clair de deux blocs et par $(N, C_0^3 \| C_1^3 \| C_2^3 \| C_3^3)$ le chiffré correspondant au message clair de trois blocs. Puisqu'une collision a lieu entre les *nonces* et comme on a choisi des messages utilisant les mêmes blocs M_1 et M_2, on remarque que $C_1^1 = C_1^2 = C_1^3$ et $C_2^2 = C_2^3$. Pour simplifier les notations, on note simplement C_1 le premier bloc précédent et C_2 le second. On désigne également par T_1 la valeur du CBC-MAC pour M_1, par T_2 celle pour $M_1 \| M_2$, et par T_3 celle pour $M_1 \| M_2 \| M_3$. Ces notations sont données sur la figure 4.1.

À cause des collisions entre les *nonces* utilisés pour ces trois chiffrements, la valeur $C_0^1 \oplus C_0^2$ est égale à $T_1 \oplus T_2$. Bien que l'attaquant ne connaisse pas les valeurs T_1 et T_2, il peut exploiter la connaissance de leur xor pour contrefaire un chiffré valide : en effet, le chiffré $C_0^3 \| C_1 \| C_2 \oplus M_2 \oplus C_0^1 \oplus C_0^2 \oplus M_3$ est valide pour le *nonce* N et le message clair $M_1 \| C_0^1 \oplus C_0^2 \oplus M_3$. Pour ce dernier, l'entrée de la troisième boîte de chiffrement dans la chaîne CBC est un xor du bloc du message clair $C_0^1 \oplus C_0^2 \oplus M_3$ avec la sortie précédente T_1. Puisque $C_0^1 \oplus C_0^2 = T_1 \oplus T_2$, l'entrée est donc $T_2 \oplus M_3$, ce qui était aussi l'entrée du quatrième chiffrement dans le CBC pour le chiffrement de $M_1 \| M_2 \| M_3$. Par conséquent, la sortie est T_3 (inconnue de l'adversaire) et le premier bloc du

chiffré est C_0^3. Le second bloc du chiffré est facilement calculable à partir de C_2 à cause de la malléabilité du mode compteur.

Enfin, l'attaquant peut contrefaire un chiffré valide avec l'aide de $3 \times 2^{2\ell/3}$ requêtes de chiffrement. Si $\ell \leq 96 = 3n/4$ alors $2\ell/3 \leq n/2$ et cette attaque est meilleure que la borne de sécurité donnée par Jonsson.

Cette attaque illustre le fait que si on veut préserver la sécurité, on ne peut pas augmenter l'efficacité de CCM en enlevant le format particulier, avec la taille du message ajoutée au début de celui-ci, et en permettant de répéter les *nonces*.

4.6 Version transformée de CCM

Nous allons maintenant présenter une version modifiée de CCM qui présente l'avantage d'être prouvée sûre jusqu'à une borne de 2^n tant en confidentialité qu'en authentification. La modification apportée est légère afin de préserver l'architecture globale de CCM, telle qu'elle est décrite dans les normes. Pour être précis, nous proposons d'utiliser deux clefs différentes pour l'authentification et le chiffrement et de remplacer l'algorithme de chiffrement par bloc utilisé dans le mode compteur par une fonction pseudo-aléatoire. Si l'on souhaite baser la sécurité du schéma sur celle d'un algorithme de chiffrement par blocs, on peut construire cette fonction pseudo-aléatoire en utilisant plusieurs algorithmes de chiffrement par blocs différents, tel que cela est fait dans [BI99, HWKS98, Luc00]. Une autre possibilité est d'utiliser la fonction de compression d'une fonction de hachage, où la clef de la fonction pseudo-aléatoire prend la place de la valeur de chaînage dans la fonction de hachage. Cette solution s'appuie sur une hypothèse peu classique sur les fonctions de compression, à savoir qu'il s'agit d'une bonne fonction pseudo-aléatoire. Cependant cette hypothèse est de plus en plus commune [Bel06, DGH$^+$04a] et semble réaliste. Par ailleurs, elle présente l'avantage d'être très efficace en pratique.

4.6.1 Description

Dans cette section nous proposons une version modifiée de CCM (notée mCCM) que nous prouvons sûre au delà de la borne du paradoxe des anniversaires. La principale différence entre les versions originale et modifiée de CCM est l'utilisation d'une fonction pseudo-aléatoire pour chiffrer le mac et le message. Soient F une famille de fonctions pseudo-aléatoires de $\{0,1\}^n$ dans $\{0,1\}^n$, E un algorithme de chiffrement par blocs sur $\{0,1\}^n$ et K et K' deux clefs choisies uniformément et de façon indépendante. On souligne le fait que dans la version modifiée, nous imposons que les deux clefs soient choisies indépendamment alors que dans la version originale les deux clefs sont égales.

Pour chiffrer un message M, D, on choisit un *nonce* N, on en déduit l'entrée formatée B correspondante et on calcule un mac T en utilisant l'algorithme CBC-MAC, grâce à l'algorithme de chiffrement par blocs E_K. Ensuite le texte chiffré est calculé de la manière suivante $C_0 = T \oplus \mathsf{lsb}_t\left(F_{K'}(A_0)\right)$ et $C_i = M_i \oplus F_{K'}(A_i)$.

Le processus de déchiffrement commence par déchiffrer C avec le mode compteur et obtient un message M' et un tag T, ensuite il calcule le tag T' à partir de l'entrée formatée B' déduite du *nonce* N, des données associées et de M'. Si le chiffré est valide, c'est-à-dire si $T = T'$, alors il retourne M', sinon il renvoie un message d'erreur.

Nous rappelons que dans le cas de la version modifiée de CCM, de même que dans la version originale, nous imposons que les *nonces* ne se répètent pas. Ceci implique que les adversaires contre la version modifiée de CCM peuvent choisir le *nonce* à utiliser pour chiffrer la requête, tant qu'il est différent des *nonces* déjà employés pour une requête de chiffrement (mais pas

nécessairement d'une requête de vérification). L'adversaire est donc astreint à faire au plus 2^ℓ requêtes, une pour chaque *nonce* (ceux-ci étant de taille au plus ℓ). Au contraire, pour les requêtes de vérification aucune restriction n'est imposée : l'adversaire est autorisé à utiliser n'importe quel *nonce*, même un *nonce* qui a déjà été employé dans une requête de chiffrement ou dans une autre requête de vérification.

Nous allons maintenant prouver les sécurités INT-CTXT et IND-CPA de cette version modifiée de CCM. On remarquera que comme prouvé dans [BN00] ces deux notions de sécurité impliquent une sécurité IND-CCA en confidentialité (avec une réduction fine), ce qui signifie que ce protocole atteint le meilleur niveau de sécurité tant en confidentialité qu'en intégrité (voir [BN00] pour les définitions et les relations entre ces notions).

4.6.2 Confidentialité

La confidentialité de la version modifiée de CCM est une conséquence directe de la confidentialité du mode compteur en utilisant une fonction pseudo-aléatoire, résultat établi dans le théorème 4.2. La sécurité de mCCM est une conséquence simple de ce théorème qui permet d'aller au-delà du paradoxe des anniversaires.

Théorème 4.3. *Soit F une famille de PRF de $\{0,1\}^n$ dans $\{0,1\}^n$. Alors, pour tout adversaire $\mathcal{A}$ contre la confidentialité de mCCM, avec un temps de calcul T, et qui peut faire au plus q_e requêtes de chiffrement d'au plus s blocs chacune, il existe un adversaire $\mathcal{A}'$ contre F, avec un temps de calcul d'au plus T et qui fait au plus $(s+1)q_e$ requêtes de chiffrement, tel que :*

$$\mathsf{adv}^{\texttt{ind-cpa}}_{mCCM}(\mathcal{A}) \leq 2\mathsf{adv}^{\texttt{prf}}_{F}\left(\mathcal{A}'\right).$$

Démonstration. L'adversaire $\mathcal{A}$ contre mCCM peut très facilement être utilisé pour créer un adversaire en confidentialité contre le mode compteur. Soit $\mathcal{A}''$ l'adversaire contre le mode compteur suivant : il choisit une clef K' au hasard et lance $\mathcal{A}$. À la j^{e} requête (M_j^0, M_j^1) de $\mathcal{A}$, $\mathcal{A}''$ calcule $T_j^0 = CBC_{K'}(M_j^0)$ et $T_j^1 = CBC_{K'}(M_j^1)$ et envoie $(T_j^0 \| M_j^0, T_j^1 \| M_j^1)$ au challenger du mode compteur et fait suivre la réponse de celui-ci à $\mathcal{A}$. À la fin du jeu, $\mathcal{A}$ renvoie un bit b' que $\mathcal{A}''$ fait suivre au challenger du mode compteur. Il est facile de vérifier que les avantages de $\mathcal{A}$ et $\mathcal{A}''$ sont égaux : $\mathsf{adv}^{\texttt{ind-cpa}}_{mCCM}(\mathcal{A}) = \mathsf{adv}^{\texttt{ind-cpa}}_{CTR}(\mathcal{A}'')$. Le théorème 4.2 permet de conclure. □

4.6.3 Intégrité

Pour prouver l'intégrité des textes chiffrés du mode mCCM, nous avons besoin de deux résultats. Le premier est le théorème 4.1. Le second résultat provient de [BGM04] et prouve que si un protocole est INT-CTXT sûr contre un adversaire qui peut faire au plus 1 requête de vérification, alors il est sûr contre un adversaire qui peut faire plusieurs requêtes de vérification, la perte de sécurité étant linéaire en le nombre de requêtes de vérification pouvant être faites.

Lemme 4.1. *Soit $\mathcal{A}$ un adversaire contre l'intégrité d'un schéma de chiffrement symétrique Π. Supposons que $\mathcal{A}$ fait au plus q_e requêtes de chiffrement et q_v requêtes de vérification toutes constituées d'au plus s blocs. Il existe donc un adversaire $\mathcal{A}'$ contre l'intégrité de Π, tel que $\mathcal{A}'$ fait au plus q_e requêtes de chiffrement et 1 requête de vérification toutes constituées d'au plus s blocs et tel que :*

$$\mathsf{Succ}^{\texttt{int-ctxt}}_{\Pi}(\mathcal{A}) \leq q_v \cdot \mathsf{Succ}^{\texttt{int-ctxt}}_{\Pi}\left(\mathcal{A}'\right).$$

Les adversaires $\mathcal{A}$ et $\mathcal{A}'$ ont le même temps de calcul.

Grâce à ces résultats, nous pouvons majorer l'avantage de n'importe quel adversaire contre l'INT-CTXT de mCCM. La méthode de preuve employée est inspirée de la méthode employée dans [Kra01] pour prouver que le mode authentifier-puis-chiffrer employé avec l'algorithme de chiffrement *one-time-pad* est un bon mode.

Théorème 4.4. *Soit $\mathcal{A}$ un adversaire contre l'intégrité de mCCM qui a un temps de calcul égal à T, qui peut faire au plus q_e requêtes de chiffrement d'au plus s blocs et q_v requêtes de vérification d'au plus s blocs. Sa probabilité de succès est alors majorée par :*

$$\begin{aligned}\mathsf{Succ}^{\texttt{int-ctxt}}_{mCCM}(\mathcal{A}) \;\leq\; & q_v\left(\frac{48(s+1)}{2^n}+256\left(\frac{(s+1)^2}{2^n}\right)^2+\frac{1}{2^t}\right)\\ & +\mathsf{adv}^{\texttt{prf}}_F(\mathcal{A}_1)+\mathsf{adv}^{\texttt{prf}}_E(\mathcal{A}_2),\end{aligned}$$

où $\mathcal{A}_1$ est un adversaire en PRF contre F de complexité temporelle T, qui peut faire au plus $(s+1)(q_e+q_v)$ requêtes et $\mathcal{A}_2$ est un adversaire en prp contre E avec une complexité temporelle T, qui peut faire au plus $(s+1)(q_e+q_v)$ requêtes.

Démonstration. La preuve suivante est une preuve basée sur les jeux. Dans le premier jeu (le jeu 1), l'adversaire fait face à un oracle de chiffrement et à un oracle de vérification qui sont simulés en respectant strictement le protocole.

Dans les jeux 2 et 3, nous remplaçons successivement la PRF F et la PRP E par respectivement une fonction parfaitement aléatoire et un permutation parfaitement aléatoire. On peut montrer facilement qu'il existe deux adversaires $\mathcal{A}_1$ et $\mathcal{A}_2$, où $\mathcal{A}_1$ est un adversaire en PRF contre F de complexité temporelle T qui peut faire au plus $(s+1)(q_e+q_v)$ requêtes et $\mathcal{A}_2$ est un adversaire en prp contre E avec une complexité temporelle T qui peut faire au plus $(s+1)(q_e+q_v)$ requêtes, et qui sont tels que la distance entre le jeu 1 et le jeu 2 et la distance entre le jeu 2 et le jeu 3 sont majorées respectivement par $\mathsf{adv}^{\texttt{prf}}_F(\mathcal{A}_1)$ et $\mathsf{adv}^{\texttt{prf}}_E(\mathcal{A}_2)$.

Soit $\mathcal{A}_3$ un adversaire contre l'intégrité de mCCM dans le jeu 3, il peut faire au plus q_e requêtes de chiffrement et q_v requêtes de vérification, toutes d'au plus s blocs. Soit $\mathcal{A}'$ un adversaire contre l'intégrité de mCCM qui fait q_e requêtes de chiffrement d'au plus s blocs et 1 requête de vérification d'au plus s blocs tel que : $\mathsf{Succ}^{\texttt{int-ctxt}}_{mCCM}(\mathcal{A}_3) \leq q_v \cdot \mathsf{Succ}^{\texttt{int-ctxt}}_{mCCM}(\mathcal{A}')$ (cet adversaire existe grâce au lemme 4.1). Pour majorer la probabilité de succès de $\mathcal{A}_3$, nous bornons la probabilité de succès de $\mathcal{A}'$. Pour ceci, grâce à $\mathcal{A}'$ nous construisons $\mathcal{D}$ un adversaire en PRF sans préfixe contre CBC-MAC avec 2 requêtes de MAC d'au plus s blocs, et on reliera alors la probabilité de succès de $\mathcal{D}$ avec celle de $\mathcal{A}'$.

Comme décrit dans la définition de la sécurité des PRF, un adversaire PRF contre CBC-MAC fait face soit à un oracle de CBC-MAC, soit à une fonction parfaitement aléatoire et doit deviner devant lequel des deux il se trouve. Pour construire $\mathcal{D}$, nous lançons $\mathcal{A}'$ et à chacune de ses requêtes de chiffrement (N^i, M^i, D^i), nous répondons de la manière suivante :

- $\perp$ s'il existe $k < i$ tel que $N^i = N^k$,
- $(N^i, D^i, C^i = C^i_0 \| \dots \| C^i_{m_i})$ avec $C^i_0 \xleftarrow{\$} \{0,1\}^n$, $C^i_k = M^i_k \oplus F(A^i_k)$.

Pour une requête de vérification $(N, D, C = C_0 \| \dots \| C_m)$ de $\mathcal{A}'$, nous répondons par :

- Si $N \neq N^k$ pour tout $k \leq q_e$, alors on choisit au hasard une chaîne de t bits T et nous calculons un message M de m blocs avec $M_i = C_i \oplus F(A_i)$; on donne B, où B est l'entrée formatée correspondante, à l'oracle de vérification qui répond par T' et on répond 1 à $\mathcal{A}'$ si et seulement si $T = \mathsf{lsb}_t(T')$,
- s'il y a $k \leq q_e$ tel que $N = N^k$, $D = D^k$ et $C = C^k$, alors on répond $\perp$,

– S'il y a $k \leq q_e$ tel que $N = N^k$, mais $D \neq D^k$ ou $C \neq C^k$, alors, nous calculons les blocs $M_i = F(A_i) \oplus C_i$ du message, on en déduit alors l'entrée formatée correspondante B ; on envoie alors la k^{e} entrée formatée B^k (de la k^{e} requête de chiffrement) à l'oracle CBC-MAC et on reçoit le mac correspondant T^k ; alors, on calcule le mac de $B : T = \mathsf{lsb}_t\left(T^k\right) \oplus C_0 \oplus C_0^k$; enfin, on envoie B à l'oracle CBC-MAC, on reçoit T', on vérifie si $T = \mathsf{lsb}_t\left(T'\right)$ et on fait suivre la réponse à $\mathcal{A}'$ (puisque $D \neq D^k$ ou $C \neq C^k$, on a $B \neq B^k$ et puisque les entrées formatées forment un ensemble sans préfixe, les deux requêtes faites à l'oracle CBC-MAC ne sont pas préfixes l'une de l'autre).

À la fin, $\mathcal{D}$ décide qu'il fait face à un vrai oracle CBC-MAC si la réponse faite à la requête de vérification de $\mathcal{A}'$ est égale à 1.

Dès que les *nonces* sont différents les uns des autres, pour un attaquant contre mCCM les C_0^i sont distribués aléatoirement, et donc les réponses à $\mathcal{A}'$ sont bien simulées. Puisqu'il n'y a pas de collision entre les *nonces*, la probabilité de succès de $\mathcal{A}'$ est exactement la probabilité que $\mathcal{D}$ sorte 1 quand il fait face à un vrai adversaire CBC-MAC. Quand $\mathcal{D}$ fait face à une fonction aléatoire, sa probabilité de sortir 1 est égale à $1/2^t$, par conséquent on a :

$$\mathsf{Succ}^{\texttt{int-ctxt}}_{mCCM}\left(\mathcal{A}'\right) \leq \frac{1}{2^t} + \mathsf{adv}^{\texttt{pf-prf}}_{CBC\text{-}MAC}\left(\mathcal{D}\right).$$

Nous rappelons que pf dans pf-PRF signifie sans préfixe. Le théorème 4.1 permet alors de conclure. □

En pratique, nous pouvons considérer que $s \leq 2^{40} - 1$. Dans l'exemple qui suit, nous omettons les termes en référence aux avantages en PRF et en PRP pour des raisons de simplicité, puisque ces termes sont présents dans les bornes de sécurité de CCM et dans celles de mCCM. Supposons que $t \geq 82$, le théorème précédent donne alors une borne sur la sécurité en intégrité de mCCM inférieure à $q_v \cdot 2^{-80}$. Pour la même valeur de t, le théorème de Jonsson [Jon03] donne une borne sur la sécurité approximativement égale à $(q_v + q_e)^2 \cdot 2^{-48}$. On remarquera que pour une valeur de $t = 82$, notre borne est fine puisque l'attaque simple consistant à essayer de deviner la valeur CBC-MAC a une probabilité de succès de $q_v/2^t = q_v \cdot 2^{-82}$.

4.7 Conclusion

Dans ce chapitre, nous avons étudié la sécurité du schéma de chiffrement authentifié CCM et d'une version modifiée. Nous avons montré qu'en modifiant légèrement CCM, on peut prouver une sécurité en intégrité en $\mathcal{O}(2^n)$, sécurité qui n'est que conjecturée pour CCM. Par ailleurs, la version modifiée de CCM garantit une sécurité optimale en confidentialité.

Par ailleurs, nous avons étudié les propriétés de CCM (ainsi que celles de la version modifiée de CCM) qui réduise l'efficacité. Nous avons montré que si on relâche certaines d'entre elles (si on laisse une collision entre les *nonces* se produire, si on n'oblige plus à ce que les messages authentifiés soient sans préfixe en enlevant du format la longueur du message et/ou celle des données authentifiées) alors on peut monter des attaques qui sont meilleures que la borne de sécurité conjecturée et même meilleures que la borne prouvée par Jonsson.

Troisième partie

Dérivation de clefs

5

Extraction de clefs à partir d'un élément Diffie-Hellman

Sommaire

Dans la partie précédente, nous avons présenté deux protocoles qui, mis bout à bout, permettent de créer un canal confidentiel, authentifié et intègre. Dans le chapitre 3, l'échange de clefs authentifié est fait dans le modèle de l'oracle aléatoire, le secret généré est donc une chaîne de bits uniformément distribuée et peut donc être utilisé, sans modification, comme clef pour l'algorithme de chiffrement authentifié présenté dans le chapitre 4. Cependant, le modèle de l'oracle aléatoire est un modèle fort et on préfère quand cela est possible utiliser des algorithmes d'échange de clefs sûrs sans oracles aléatoires. Dans ce cas, les secrets générés ne sont pas toujours des chaînes de bits uniformément distribuées et on ne peut pas les utiliser en l'état comme clef pour un algorithme de chiffrement authentifié, il faut ajouter une étape dite de dérivation de clefs. Plus précisément, on obtient un élément secret qui, même s'il n'est pas une chaîne de bits uniformément distribuée, possède aux yeux de l'adversaire beaucoup d'entropie. Aussi, dans un premier temps, on procède à une extraction d'entropie pour créer une chaîne de bits qui peut être petite mais qui est, elle, uniformément distribuée. Dans un second temps, grâce à un générateur pseudo-aléatoire, on génère une longue chaîne de bits qui pour l'attaquant semble parfaitement aléatoire. La première étape est l'étape d'extraction de clef, la seconde est l'étape d'expansion, les deux étapes réunis formant l'étape de dérivation de clef.

Comme on l'a vu en introduction à la section 1.3, la confidentialité des clefs est essentielle pour assurer l'authentification et plus généralement la sécurité du canal. Aussi, il faut porter une

attention toute particulière à l'étape de dérivation de clefs et plus particulièrement à la phase d'extraction de clefs, qui est la partie la plus "négligée" en pratique, afin de vérifier qu'aucune information sur la valeur des clefs ne fuit à ce moment du protocole. Dans ce chapitre et le suivant, nous étudions plus en détails cette étape d'extraction de clefs, c'est-à-dire que nous nous intéressons aux algorithmes qui permettent de générer, à partir d'un élément secret de grande taille et possédant beaucoup d'entropie, une chaîne de bits plus petite mais qui semble uniformément distribuée.

Un des échanges de clefs les plus connus et les plus répandus en pratique est l'échange de clefs Diffie-Hellman. Nous nous intéressons ici à sa variante sur corps fini. Lorsque deux parties, Alice et Bob, veulent procéder à un échange de clefs, ils se mettent d'accord sur un nombre premier p sur n bits et sur un sous-groupe $\mathbb{G}$ de $\mathbb{Z}_p^\star$ engendré par un élément g, puis Alice génère x de façon uniforme entre 1 et $|\mathbb{G}|$ et envoie g^x à Bob, ce dernier génère y de façon uniforme entre 1 et $|\mathbb{G}|$ et envoie g^y à Alice. Ils calculent alors tous deux l'élément g^{xy}. Si l'hypothèse CDH est vraie dans le groupe $\mathbb{G}$ alors l'adversaire n'est pas en mesure de calculer l'élément g^{xy} qui est donc secret aux yeux de l'adversaire.

Avec un oracle aléatoire, en hachant g^{xy} il est maintenant facile de dériver une chaîne de bits pouvant être utilisée comme clef secrète pour un algorithme symétrique. Par contre, construire un algorithme de dérivation de clefs sans oracle aléatoire est plus difficile. Dans ce cas, il est classique de supposer qu'Alice et Bob ont choisi un groupe $\mathbb{G}$ tel que l'hypothèse DDH est vraie. Cela implique que pour l'adversaire, même s'il connaît g^x et g^y, l'élément g^{xy} est indistinguable d'un élément aléatoire du groupe $\mathbb{G}$. Nous sommes ramenés au problème de dériver une clef à partir d'un élément g^z uniformément distribué dans le groupe $\mathbb{G}$.

Il faut faire attention à ne pas confondre chaîne de bits uniformément distribuée dans $\{0,1\}^n$ et élément uniformément distribué dans $\mathbb{G}$: il n'est pas possible d'utiliser l'élément g^z tel quel car certains bits de g^z peuvent être connus de l'adversaire. C'est effectivement le cas du bit de poids fort lorsque p est un nombre premier sur n bits tel que $p = 2^{n-1} + \delta$ avec $\delta \ll |\mathbb{G}|$. En effet, si $0 \leq g^z \leq 2^{n-1} - 1$, le bit de poids fort de g^z, noté $\mathsf{msb}\,(g^z)$, vaut 0, alors que si $2^{n-1} \leq g^z \leq p - 1$, on a $\mathsf{msb}\,(g^z) = 1$. Or la probabilité que g^z soit compris entre 2^{n-1} et $p - 1$ est inférieure à $(\delta - 1)/|\mathbb{G}|$. D'après l'hypothèse $\delta \ll |\mathbb{G}|$, la probabilité que ceci arrive est très faible et, par conséquent, le bit de poids fort de g^z vaut 1 avec une écrasante probabilité ! Un extracteur d'entropie est donc indispensable.

Il existe déjà plusieurs extracteurs d'entropie dans la littérature que l'on peut utiliser après un échange Diffie-Hellman, comme par exemple le *leftover hash lemma* ou l'extracteur déterministe évoqué par Chevassut *et al.* [CFGP05, CFGP06] et présenté à la sous-section 5.2.2. En ce qui concerne l'échange de clefs Diffie-Hellman sur courbes elliptiques, Chevassut *et al.* [CFGP06] ont présenté une nouvelle technique appelée TAU, mais elle est spécifique à certaines courbes elliptiques, et Gürel [GÖ5] a prouvé que si la courbe elliptique est définie sur une extension quadratique d'un corps fini et si l'hypothèse DDH est vraie, les bits de poids fort de l'abscisse de l'élément Diffie-Hellman sont statistiquement proches d'une chaîne de bits aléatoire.

Dans ce chapitre, nous proposons et prouvons la sécurité d'un extracteur d'entropie déterministe pour l'échange Diffie-Hellman, qui ressemble à celui de Gürel, mais pouvant être utilisé dans des sous-groupes de $\mathbb{Z}_p^\star$ suffisamment grands. L'avantage de cet extracteur d'entropie est double : il est très simple à mettre en œuvre et il est prouvé sûr dans le modèle standard. En effet, il consiste simplement à prendre les k bits de poids faible de l'élément g^z et le principal résultat de ce chapitre est le théorème 5.2 qui établit que les bits de poids faible d'un élément aléatoire de $\mathbb{G}$ sont *statistiquement* indistinguables d'une chaîne de bits parfaitement aléatoire. Notre résultat en lui-même ne nécessite pas l'hypothèse DDH (et est donc sûr au sens de la théorie de l'information). Cependant, comme cela est évoqué ci-dessus, pour utiliser ce résul-

tat dans un protocole cryptographique, nous avons besoin de l'hypothèse DDH afin d'obtenir un élément aléatoire dans un sous-groupe de $\mathbb{Z}_p^\star$. Pour prouver le résultat, nous utiliserons des techniques à base de sommes d'exponentielles [CFK+00, CFS99] qui nous permettent d'établir une borne supérieure pour une distance statistique. Enfin, cet extracteur d'entropie peut avoir d'autres applications que l'échange de clefs, nous les évoquons à la fin du chapitre. Notons que les résultats présentés dans ce chapitre ont donné lieu à une publication [FPSZ06].

5.1 Extracteur d'entropie dans un sous-groupe de $\mathbb{Z}_p^\star$

5.1.1 Notations et préliminaires mathématiques

Avant de présenter les résultats centraux de ce chapitre, nous donnons quelques préliminaires mathématiques nécessaires.

Soient p un nombre premier sur n bits, c'est-à-dire que $2^{n-1} < p < 2^n$, $\mathbb{G}$ un sous-groupe de $\mathbb{Z}_p^\star$ d'ordre q avec $q \gg \sqrt{p}$, g un élément qui engendre $\mathbb{G}$, ℓ un entier tel que $2^{\ell-1} \leq q < 2^\ell$ et enfin X une variable uniformément distribuée dans $\mathbb{G}$. Dans la suite, nous noterons par k un entier, s désignera à la fois une chaîne de bits de taille k et l'entier de $[\![0, 2^k - 1]\!]$ associé, et U_k sera une variable uniformément distribuée dans $\{0,1\}^k$. Si x est un entier, nous noterons par $\mathsf{lsb}_k(x)$ les k bits de poids faible de x et par $\mathsf{msb}_k(x)$ les k bits de poids fort de x.

Pour les démonstrations de ce chapitre, nous utiliserons des résultats sur les caractères. Un caractère est un morphisme de $(\mathbb{Z}_p, +)$ dans $(\mathbb{C}^*, \cdot)$. Afin de simplifier les preuves nous ne nous intéresserons qu'au caractère noté e_p qui à tout $y \in \mathbb{Z}_p$, associe $e_p(y) = e^{\frac{2i\pi y}{p}} \in \mathbb{C}^*$. Plus précisément nous allons nous concentrer sur certaines sommes faisant intervenir ce caractère : $\sum_{x\in\mathbb{G}} e_p(ax)$. Pour tout $a \in \mathbb{Z}^\star$, fixons la notation

$$S(a, \mathbb{G}) = \sum_{x\in\mathbb{G}} e_p(ax).$$

Commençons par donner quelques résultats simples.

Lemme 5.1. *Soient p un nombre premier et $\mathbb{G}$ un sous-groupe de $\mathbb{Z}_p^\star$ muni de la multiplication.*
(1) Si $a = 0$, $\sum_{x=0}^{p-1} e_p(ax) = p$,
(2) Pour tout $a \in \mathbb{Z}_p^\star$, $\sum_{x=0}^{p-1} e_p(ax) = 0$,
(3) Pour tout $x_0 \in \mathbb{G}$ et tout $a \in \mathbb{Z}_p^\star$, $S(ax_0, \mathbb{G}) = S(a, \mathbb{G})$.

Démonstration. La remarque (1) est immédiate puisque $e_p(0) = 1$.
Si $a \neq 0$, la somme $\sum_{x=0}^{p-1} e_p(ax)$ est une série géométrique égale à $(1 - e_p(ap))/(1 - e_p(a)) = 0$, ce qui prouve la remarque (2).
Enfin, comme l'application de $\mathbb{G}$ dans $\mathbb{G}$ qui à tout x associe $x \cdot x_0$ est bijective, le changement de variable $x' = x \cdot x_0$ permet de prouver que $\sum_{x\in\mathbb{G}} e_p(ax \cdot x_0) = \sum_{x'\in\mathbb{G}} e_p(ax')$, c'est-à-dire de prouver la remarque (3). □

Nous présentons maintenant un résultat plus élaboré découlant directement des remarques ci-dessus. Il s'agit d'une majoration de $|S(a, \mathbb{G})|$, pour $a \neq 0$, appelée borne de Polya-Vinogradov car elle a été trouvée par ces deux auteurs indépendamment.

Théorème 5.1 (Borne de Polya-Vinogradov)**.** *Soient p un nombre premier et $\mathbb{G}$ un sous-groupe de $\mathbb{Z}_p^\star$ muni de la multiplication. On a alors pour tout $a \in \mathbb{Z}_p^\star$:*

$$\left| \sum_{x\in\mathbb{G}} e_p(ax) \right| \leq \sqrt{p}.$$

Démonstration. On a $\sum_{a\in\mathbb{Z}_p^\star} |S(a,\mathbb{G})|^2 = p|\mathbb{G}|$. En effet, comme $\overline{S(a,\mathbb{G})} = S(-a,\mathbb{G})$:

$$\begin{aligned}\sum_{a\in\mathbb{Z}_p^\star} |S(a,\mathbb{G})|^2 &= \sum_{a\in\mathbb{Z}_p^\star} S(a,\mathbb{G})S(-a,\mathbb{G}) = \sum_{a\in\mathbb{Z}_p^\star}\sum_{x_1\in\mathbb{G}} e_p(ax_1)\sum_{x_2\in\mathbb{G}} e_p(-ax_2)\\ &= \sum_{a\in\mathbb{Z}_p^\star}\sum_{x_1,x_2\in\mathbb{G}} e_p(a(x_1-x_2)) = \sum_{x_1,x_2\in\mathbb{G}}\sum_{a\in\mathbb{Z}_p^\star} e_p(a(x_1-x_2))\\ &= \sum_{x_1=x_2\in\mathbb{G}} p = p|\mathbb{G}|.\end{aligned}$$

L'avant-dernière égalité provient des remarques (1) et (2) du lemme précédent.

Grâce à la troisième remarque du lemme précédent, on sait que l'on a $|\mathbb{G}||S(a,\mathbb{G})|^2 = \sum_{x\in\mathbb{G}} |S(ax,\mathbb{G})|^2$, ce qui est inférieur à $\sum_{a\in\mathbb{Z}_p^\star} |S(a,\mathbb{G})|^2$ qui vaut $p|\mathbb{G}|$. On en déduit donc que $|S(a,\mathbb{G})|^2 \leq p$, ce qui est le résultat attendu. □

Cette borne est non triviale dès lors que $\sqrt{p} \leq |\mathbb{G}|$. Le lecteur intéressé par des résultats avancés sur les sommes introduites ici pourra consulter [KS99].

5.1.2 Extraction d'entropie sur les bits de poids faible

Dans cette partie, nous montrons que les k bits de poids faible d'un élément aléatoire de $\mathbb{G}$ sont statistiquement proches d'une chaîne de k bits parfaitement aléatoire, pourvu que $\mathbb{G}$ soit suffisament grand. De façon informelle, une conséquence directe de ceci est que la fonction de $\mathbb{Z}_p^\star$ dans $\{0,1\}^k$ qui conserve uniquement les k bits de poids faible de l'entrée est un bon extracteur d'entropie déterministe pour l'ensemble des variables aléatoires uniformément distribuées dans un des grands sous-groupes de $\mathbb{Z}_p^\star$.

Théorème 5.2. *Soient p un nombre premier sur n bits, $\mathbb{G}$ un sous-groupe de $\mathbb{Z}_p^\star$ de cardinal $q \geq 2^\ell$, X une variable aléatoire uniformément distribuée dans $\mathbb{G}$ et $k \leq n$ un entier. On a le résultat suivant :*

$$\mathbf{SD}(\mathsf{lsb}_k(X), U_k) < \frac{2^k}{p} + \frac{2^k\sqrt{p}\log_2(p)}{q} < 2^{k+n/2+\log_2(n)+1-\ell}.$$

Ces inégalités sont non triviales si $k < \ell - n/2 - \log_2(n) - 1$.

Démonstration. Soient $K = 2^k$ et $H_s = \left\lfloor \frac{p-1-s}{K} \right\rfloor$ pour $s \in [\![0,\, K-1]\!]$. On note Δ la distance statistique entre les variables aléatoires $\mathsf{lsb}_k(X)$ et U_k.

Des remarques (1) et (2) du lemme 5.1, il découle facilement que

$$\frac{1}{p} \times \sum_{a=0}^{p-1} e_p(a(g^x - s - Ku)) = \mathbb{1}(x,s,u),$$

où $\mathbb{1}(x,s,u)$ est la fonction caractéristique égale à 1 si $g^x = (s+Ku) \bmod p$ et 0 sinon. Puisque l'on considère une distribution uniforme, on a :

$$\begin{aligned}\Pr_{X\in\mathbb{G}}[\mathsf{lsb}_k(X) = s] &= \frac{1}{q} \times \left|\{(x,u) \in [\![0,\, q-1]\!] \times [\![0,\, H_s]\!] \,|\, g^x = s + Ku \bmod p\}\right|\\ &= \frac{1}{qp} \times \sum_{x=0}^{q-1}\sum_{u=0}^{H_s}\sum_{a=0}^{p-1} e_p(a(g^x - s - Ku)).\end{aligned}$$

Nous allons changer l'ordre des sommations en mettant la sommation sur a en premier et séparer cette dernière en deux termes :

1. le premier vient du cas où $a = 0$ et est égal à $(H_s + 1)/p$, c'est-à-dire approximativement $1/2^k$,
2. le second regroupe les restes de la somme et sera le terme principal dans le calcul de la distance statistique dans laquelle nous pouvons séparer les sommes sur x et sur u.

Puisque par ailleurs $|e_p(-as)| = 1$, le double de la distance statistique, c'est-à-dire 2Δ, peut donc être majoré ainsi :

$$\begin{aligned} 2\Delta &= \sum_{s\in\{0,1\}^k} \left| \Pr_{X\in\mathbb{G}}[\mathsf{lsb}_k(X) = s] - 1/2^k \right| \\ &\leq \sum_{s\in\{0,1\}^k} \left| \frac{H_s+1}{p} - \frac{1}{2^k} \right| + \sum_{s\in\{0,1\}^k} \frac{1}{qp} \sum_{a=1}^{p-1} \left| \left(\sum_{x=0}^{q-1} e_p(ag^x) \right) \left(\sum_{u=0}^{H_s} e_p(-aKu) \right) \right|. \end{aligned}$$

Pour le premier terme, nous remarquons que $\left|(H_s+1)/p - 1/2^k\right| \leq 1/p$, puisque $K = 2^k$, $H_s = \left\lfloor \frac{p-1-s}{K} \right\rfloor$ et :

$$-\frac{1}{p} \leq -\frac{1+s}{Kp} \leq \left(1 + \left\lfloor \frac{p-1-s}{K} \right\rfloor\right)\frac{1}{p} - \frac{1}{K} \leq \frac{K-(1+s)}{Kp} \leq \frac{1}{p}.$$

Pour le second terme, nous introduisons $M = \max_a \left(\left| \sum_{x=0}^{q-1} e_p(ag^x) \right| \right)$, et montrons que :

$$\begin{aligned} \sum_{a=1}^{p-1} \left| \sum_{u=0}^{H_s} e_p(-aKu) \right| &= \sum_{a=1}^{p-1} \left| \sum_{u=0}^{H_s} e_p(-au) \right| = \sum_{a=1}^{p-1} \left| \frac{1 - e_p(-a(H_s+1))}{1 - e_p(-a)} \right| \\ &= \sum_{a=1}^{p-1} \left| \frac{\sin(\frac{\pi a(H_s+1)}{p})}{\sin(\frac{\pi a}{p})} \right| = 2\sum_{a=1}^{\frac{p-1}{2}} \left| \frac{\sin(\frac{\pi a(H_s+1)}{p})}{\sin(\frac{\pi a}{p})} \right| \\ &\leq 2\sum_{a=1}^{\frac{p-1}{2}} \left| \frac{1}{\sin(\frac{\pi a}{p})} \right| \leq \sum_{a=1}^{\frac{p-1}{2}} \left| \frac{p}{a} \right| \leq p\log_2(p). \end{aligned}$$

La première égalité provient d'un changement de variables. La seconde vient du fait que $[\![0, H_s]\!]$ est un intervalle, donc la somme est une somme géométrique. Nous utilisons l'inégalité $\sin(y) \geq 2y/\pi$ si $0 \leq y \leq \pi/2$ pour la seconde inégalité. En résumé, nous avons :

$$2\Delta \leq \frac{2^k}{p} + \frac{2^k M \log_2(p)}{q}. \tag{5.1}$$

En utilisant la borne de Polya-Vinogradov $M \leq \sqrt{p}$ prouvée dans le théorème 5.1, et les inégalités $2^{n-1} < p < 2^n$ et $2^{\ell-1} \leq q < 2^\ell$, nous obtenons le résultat attendu. □

Par conséquent, puisque la min-entropie de X, comme élément de $\mathbb{Z}_p^\star$ mais uniformément distribué dans $\mathbb{G}$, est égale à $\log_2(|\mathbb{G}|) = \log_2(q)$, la proposition précédente peut aussi être formulée en terme d'extraction d'entropie. Pour cela nous introduisons l'ensemble suivant :

Définition 5.1. *Soient ℓ un entier et $\mathcal{W}_\ell$ l'ensemble de variables aléatoires défini ainsi :*

$$\mathcal{W}_\ell = \left\{ X \,|\, X \text{ uniformément distribuée dans } \mathbb{G} \text{ sous-groupe de } \mathbb{Z}_p^\star, \text{ avec } |\mathbb{G}| \geq 2^\ell \right\}.$$

Le résultat s'exprime donc en termes d'extracteur déterministe (voir l'introduction, section 2.2.3 pour la définition de cette notion).

Corollaire 5.1. *Soit e un entier positif et $\ell = n/2+k+e+\log_2(n)+1$. Si $n \geq \ell$ alors l'application Ext_k qui va de $\{0,1\}^n$ dans $\{0,1\}^k$ et qui à tout x associe $\mathsf{lsb}_k(x)$ est un $(\mathcal{W}_\ell, 2^{-e})$-extracteur déterministe.*

5.1.3 Améliorations

Un inconvénient du résultat précédent réside dans le fait que nous avons besoin d'un sous-groupe d'ordre au moins $\sqrt{p}$. Or, afin d'avoir un échange de clefs Diffie-Hellman efficace, on préfère utiliser des sous-groupes plus petits. Par conséquent, pour améliorer les résultats obtenus sur cet extracteur d'entropie, une idée serait de trouver une meilleure borne que $\sqrt{p}$ pour $M = \max_a \left(\left| \sum_{x=0}^{q-1} e_p(ag^x) \right| \right)$. Il existe plusieurs résultats qui diminuent cette borne, comme ceux de [BK03, HBK00]. Plusieurs d'entre eux sont asymptotiques et ne spécifient pas les constantes qui interviennent dans la majoration. Cependant, en reprenant attentivement les calculs des preuves dans [HBK00] ou dans [KS99] on peut les retrouver :

Théorème 5.3 ([KS99])**.** *Avec les notations précédentes, si $q \geq 256$ alors, pour tout $x \in \mathbb{Z}_p^\star$, nous avons :*

$$M \leq \begin{cases} p^{1/2} & \left(\text{intéressant si } p^{2/3} \leq q\right) \\ 4p^{1/4}q^{3/8} & \left(\text{intéressant si } p^{1/2} \leq q \leq p^{2/3}\right) \\ 4p^{1/8}q^{5/8} & \left(\text{intéressant si } 256 \leq q \leq p^{1/2}\right) \end{cases}$$

La borne $\sqrt{p}$ est toujours valide quels que soient p et q, mais elle n'est plus intéressante que les deux autres que sur l'intervalle $p^{2/3} \leq q \leq p$. En appliquant ces bornes, dans la preuve du théorème 5.2 sur les bits de poids faible, on peut reformuler ce théorème de la façon suivante :

Théorème 5.4. *Soient p un nombre premier sur n bits, $\mathbb{G}$ un sous-groupe de $\mathbb{Z}_p^\star$ de cardinal $q \geq 2^\ell$, X une variable aléatoire uniformément distribuée dans $\mathbb{G}$ et $k \leq n$ un entier. On a le résultat suivant :*

$$\mathbf{SD}(\mathsf{lsb}_k(X), U_k) < \frac{2^k}{p} + \begin{cases} \frac{2^k\sqrt{p}\log_2(p)}{q} < 2^{k+n/2+\log_2(n)+1-\ell} & \left(\text{ si } p^{2/3} \leq q\right) \\ \frac{4\cdot 2^k p^{1/4}\log_2(p)}{q^{5/8}} < 2^{k+n/4+\log_2(n)+3-5\ell/8} & \left(\text{ si } p^{1/2} \leq q \leq p^{2/3}\right) \\ \frac{4\cdot 2^k p^{1/8} q\log_2(p)}{q^{3/8}} < 2^{k+n/8+\log_2(n)+3-3\ell/8} & \left(\text{ si } 256 \leq q \leq p^{1/2}\right). \end{cases}$$

Par exemple pour $n = 2048$, $\ell = 1176$ et $e = 80$, le théorème 5.2 nous dit que nous pouvons extraire 60 bits. Avec le théorème ci-dessus on obtient que $k \leq 5\ell/8 - (e + n/4 + \log_2(n) + 3)$, ce qui signifie que nous pouvons extraire 129 bits et obtenir une chaîne de bits de taille raisonnable. Cependant, dans la plupart des cas pratiques, la borne classique $\sqrt{p}$ reste la borne la plus appropriée et le théorème 5.2 suffit.

Par ailleurs, Gauss a prouvé que $\left| \sum_{x=0}^{p-1} e_p(ax^2) \right| = \sqrt{p}$, pour tout $a \in \mathbb{Z}_p^\star$. Par conséquent, quand $\mathbb{G}$ est le groupe des résidus quadratiques, $\left| \sum_{x\in\mathbb{G}} e_p(ax) \right| \geq (\sqrt{p}-1)/2$. Ceci signifie que dans le cas où p est un nombre premier sûr, c'est-à-dire que $p = 2q+1$ avec q nombre premier, et où $\mathbb{G}$ est le sous-groupe des résidus quadratiques de cardinal q, *en utilisant cette technique de preuve*, notre résultat est presque optimal.

5.1.4 Le cas des bits de poids fort

Les théorèmes présentés précédemment s'intéressent aux bits de poids faible. Un résultat similaire pour les bits de poids fort peut être établi en utilisant pour le démontrer les mêmes techniques. Nous obtenons le théorème suivant :

Théorème 5.5. *Soit $\delta = (2^n - p)/2^n$. Si ℓ, k et e sont des entiers tels que $5\delta < 2^{-e}$ et $\ell = n/2 + k + e + \log_2(n)$, alors la fonction $\mathsf{msb}_k(\cdot)$ est un $(\mathcal{W}_\ell, 2^{-e})$-extracteur déterministe.*

Démonstration. La preuve de ce théorème est très similaire à celle du théorème 5.2. Soient $K = 2^{n-k}$, $s_0 = \mathsf{msb}_k(p-1)$ et H_s défini pour $s \in [\![0,\, s_0]\!]$ par $H_s = 2^{n-k} - 1$ pour $0 \leq s < s_0$ et $H_s = \mathsf{msb}_{n-k}(p-1) = p - 1 - Ks_0$. Puisque

$$\frac{1}{p} \times \sum_{a=0}^{p-1} e_p(a(g^x - Ks - u)) = \mathbb{1}(x, s, u),$$

où $\mathbb{1}(x,s,u)$ est la fonction caractéristique égale à 1 si $g^x = s + Ku \bmod p$ et 0 sinon, pour $0 \leq s \leq s_0$ on a :

$$\begin{aligned}
\Pr_{X\in\mathbb{G}}[\mathsf{lsb}_k(X) = s] &= \frac{1}{q} \times \left|\{(x,u) \in [\![0,\, q-1]\!] \times [\![0,\, H_s]\!] \,|\, g^x = Ks + u \bmod p\}\right| \\
&= \frac{1}{qp} \times \sum_{x=0}^{q-1} \sum_{u=0}^{H_s} \sum_{a=0}^{p-1} e_p(a(g^x - Ks - u)).
\end{aligned}$$

Par contre, on souligne que pour $s_0 < s \leq 2^k - 1$, $\Pr_{X\in\mathbb{G}}[\mathsf{lsb}_k(X) = s] = 0$. Nous allons procéder de façon identique à la preuve pour les bits de poids faible, c'est-à-dire nous allons changer l'ordre des sommations en mettant la sommation sur a en premier et séparer cette dernière en deux termes :

1. le premier vient du cas où $a = 0$ et est égal à $(H_s + 1)/p$, c'est-à-dire approximativement $1/2^k$,
2. le second regroupe les restes de la somme et sera le terme principal dans le calcul de la distance statistique dans laquelle nous pouvons séparer les sommes sur x et sur u.

Comme $|e_p(-as)| = 1$, le double de la distance statistique, 2Δ, est inférieur à :

$$\begin{aligned}
2\Delta &= \sum_{s\in\{0,1\}^k} \left| \Pr_{X\in\mathbb{G}}[\mathsf{lsb}_k(X) = s] - 1/2^k \right| \\
&\leq \sum_{s_0 < s \leq 2^k - 1} \left| \frac{1}{2^k} \right| + \sum_{0 \leq s \leq s_0} \left| \frac{H_s + 1}{p} - \frac{1}{2^k} \right| \\
&\quad + \sum_{0\leq s\leq s_0} \frac{1}{qp} \sum_{a=1}^{p-1} \left| \left(\sum_{x=0}^{q-1} e_p(ag^x) \right) \left(\sum_{u=0}^{H_s} e_p(-au) \right) \right|.
\end{aligned}$$

Le premier terme est égal à $(2^k - s_0 - 1)/2^k$. Or, on a

$$\delta = (2^n - p)/2^n = K(2^k - s_0)/2^n + (Ks_0 - p)/2^n = (2^k - s_0)/2^k - (p - Ks_0)/2^n,$$

donc on peut majorer $(2^k - s_0)/2^k$:

$$(2^k - s_0)/2^k \leq \delta + (p - 1 - Ks_0 + 1)/2^n \leq \delta + (2^{n-k} - 1 + 1)/2^n \leq \delta + 1/2^k. \tag{5.2}$$

Le premier terme est donc inférieur à δ.

Pour le deuxième terme, nous remarquons que pour $s < s_0$, $\left|(H_s+1)/p - 1/2^k\right| \leq 2\delta/2^k$, puisque $H_s = 2^{n-k} - 1$ et :

$$\left|\frac{(H_s+1)}{p} - \frac{1}{2^k}\right| = \frac{|2^n - p|}{p2^k} = \frac{\delta}{2^k} \cdot \frac{2^n}{p} \leq \frac{2\delta}{2^k}.$$

Pour $s = s_0$, $\left|(H_s+1)/p - 1/2^k\right| \leq 2\delta/2^k$ car $H_s = p - 1 - Ks_0$ et :

$$\begin{aligned}\left|\frac{(H_s+1)}{p} - \frac{1}{2^k}\right| &= \frac{\left|2^k p - 2^n s_0 - p\right|}{p2^k} = \frac{\left|-(2^k-1)(2^n-p) + 2^n(2^k - 1 - s_0)\right|}{p2^k} \\ &\leq 2\frac{\max\left((2^k-1)(2^n-p), 2^n(2^k-1-s_0)\right)}{p2^k} \leq 2\delta.\end{aligned}$$

La première inégalité est vraie car $|a-b| \leq \max(a,b)$ si a et b sont positifs. La seconde provient de la définition de δ et de la majoration 5.2. Le deuxième terme est donc majoré par $2\delta(1+s_0/2^k)$ soit par 4δ.

Pour le troisième terme, nous introduisons $M = \max_a\left(\left|\sum_{x=0}^{q-1} e_p(ag^x)\right|\right)$, et montrons que :

$$\begin{aligned}\sum_{a=1}^{p-1}\left|\sum_{u=0}^{H_s} e_p(-au)\right| &= = \sum_{a=1}^{p-1}\left|\frac{1 - e_p(-a(H_s+1))}{1 - e_p(-a)}\right| \\ &= \sum_{a=1}^{p-1}\left|\frac{\sin(\frac{\pi a(H_s+1)}{p})}{\sin(\frac{\pi a}{p})}\right| = 2\sum_{a=1}^{\frac{p-1}{2}}\left|\frac{\sin(\frac{\pi a(H_s+1)}{p})}{\sin(\frac{\pi a}{p})}\right| \\ &\leq 2\sum_{a=1}^{\frac{p-1}{2}}\left|\frac{1}{\sin(\frac{\pi a}{p})}\right| \leq \sum_{a=1}^{\frac{p-1}{2}}\left|\frac{p}{a}\right| \leq p\log_2(p).\end{aligned}$$

La première égalité vient du fait que $[\![0, H_s]\!]$ est un intervalle, donc la somme est une somme géométrique. Nous utilisons l'inégalité $\sin(y) \geq 2y/\pi$ si $0 \leq y \leq \pi/2$ pour la seconde inégalité. En résumé, nous avons :

$$2\Delta \leq 5\delta + \frac{2^k M \log_2(p)}{q}. \tag{5.3}$$

En utilisant la borne de Polya-Vinogradov du théorème 5.1 $M \leq \sqrt{p}$ et les inégalités $2^{n-1} < p < 2^n$ et $2^{\ell-1} \leq q < 2^\ell$, nous obtenons le résultat attendu. □

La seule différence entre le cas des bits de poids fort et celui des bits de poids faible est l'ajout de l'hypothèse sur p selon laquelle $5\delta < 1/2^{e+1}$. Elle signifie que p doit être proche de 2^n par valeur inférieure. On peut facilement vérifier que cette hypothèse est nécessaire, car si elle n'est pas vraie, le bit de poids fort peut être prédictible. En effet, considérons le cas extrême où p est un nombre premier très légèrement supérieur à 2^{n-1} (et donc très loin de 2^n). Dans ce cas, avec une probabilité écrasante un élément choisi aléatoirement dans $\mathbb{Z}_p^\star$ est compris entre 0 et 2^{n-1} et son bit de poids fort est égal à 0.

Les améliorations présentées précédemment sont également valables pour les bits de poids fort et on obtient exactement le même théorème que le théorème 5.4 avec en plus l'hypothèse $5\delta < 1/2^{e+1}$.

5.2 Comparaisons

Dans la littérature il existe d'autres extracteurs d'entropie prouvés sûrs au sens de la théorie de l'information. Nous allons présenter les deux meilleurs extracteurs respectivement fort et déterministe existant qui peuvent s'appliquer à notre cas, afin de les comparer avec notre proposition.

5.2.1 Le *leftover hash lemma*

Le plus célèbre extracteur d'entropie est probablement le *leftover hash lemma*, déjà présenté dans la partie 2.2.3. On rappelle que, parmi tous les extracteurs forts, il est optimal en terme de nombre de bits extraits. Il est construit grâce à une famille de fonctions de hachage universelle et peut extraire jusqu'à $\log_2(|\mathbb{G}|) - 2e + 2$ bits à partir d'un élément aléatoire de $\mathbb{G}$. Avec notre extracteur, nous extrayons approximativement $\log_2(|\mathbb{G}|) - (n/2 + \log_2(n) - e + 1)$ bits, soit $n/2 + \log_2(n) - 3e + 1$ bits de moins. Cependant, le *leftover hash lemma* nécessite l'utilisation d'une famille de fonctions universelle de hachage et d'une source de bits parfaitement aléatoires.

Dans le cadre d'un échange de clefs, cela signifie que les participants ont à échanger des *nonces* qui rempliront le rôle de cette source de bits parfaitement aléatoire. On rappelle qu'un *nonce* (nous utilisons ici un mot anglais) est une chaîne de bits publique, utilisée une seule fois. Comme le *leftover hash lemma* requiert que ces *nonces* soient générés de façon parfaitement aléatoire, il faut s'assurer que l'adversaire ne les altère pas et ainsi ne biaise pas la manière dont ils ont été générés. Afin d'assurer la sécurité du protocole, les *nonces* doivent donc être authentifiés, ce qui augmente la complexité temporelle du protocole. Au contraire, l'extracteur décrit dans cette partie est très facile à programmer et ne nécessite pas d'outils supplémentaires pour protéger la sécurité du protocole.

En pratique, comme expliqué au chapitre suivant, sous-section 6.4.4, le *leftover hash lemma* peut être utilisé avec toujours le même IV, pourvu que celui-ci soit choisi au hasard une fois pour toutes lors de la première utilisation ou quand il est écrit en dur dans le programme. Shoup [Sho05] a prouvé que dans ce cas, il y a une perte de sécurité linéaire en le nombre d'appels à la famille de fonctions de hachage universelle.

5.2.2 Un extracteur d'entropie optimal dans un cas particulier

On considère le cas particulier où p est un nombre premier sûr (c'est-à-dire que $p = 2q+1$ où q est aussi un nombre premier) et où $\mathbb{G}$ est le sous-groupe des résidus quadratiques de cardinal q. Il existe alors un extracteur déterministe optimal évoqué notamment dans [CFGP05, CFGP06]. Soit f la fonction qui à tout $g \in \mathbb{G}$ associe :

$$f(g) = \begin{cases} g & \text{si } g \leq (p-1)/2 \\ p - 1 - g & \text{sinon} \end{cases}$$

Cette fonction est une bijection de $\mathbb{G}$ dans $\mathbb{Z}_q$. Pour obtenir une chaîne de bits aléatoire, on doit tronquer le résultat de f. La composition de f et de la troncature est un bon extracteur déterministe. Puisque f est une bijection, dans un certain sens cet extracteur est optimal : tout l'aléa est extrait. Cependant, cet extracteur très simple est aussi très contraint puisqu'il ne peut être utilisé qu'avec un nombre premier sûr alors que notre extracteur peut être utilisé avec un ensemble de nombres premiers beaucoup plus grand.

5.3 Applications en cryptographie

L'extracteur décrit ici est conçu pour extraire l'entropie à partir d'un élément aléatoire dans un groupe $\mathbb{G}$. C'est exactement ce qui est obtenu après un échange de clefs Diffie-Hellman dans un sous-groupe $\mathbb{G}$ de $\mathbb{Z}_p^\star$ où l'hypothèse DDH, décrite dans la section 2.3.1, est supposée vraie. Cette dernière permet donc d'utiliser notre extracteur dans les protocoles d'échange de clefs Diffie-Hellman, mais pas uniquement. On peut lui trouver d'autres applications en cryptographie. De manière générale, il peut être utilisé dans chaque protocole qui génère un élément aléatoire dans $\mathbb{Z}_p^\star$ et où un extracteur d'entropie est nécessaire.

5.3.1 Protocole d'échange de clefs

L'extracteur d'entropie déteministe décrit ici offre une solution simple et efficace au problème de mise en accord de clefs, solution qui est prouvée sûre dans le modèle standard sous l'hypothèse DDH : Alice envoie g^x, Bob envoie g^y et ils calculent $\mathsf{lsb}_k\left(g^{xy}\right)$.

Le groupe multiplicatif $\mathbb{Z}_p^\star$ n'est pas un groupe DDH mais si $p = \alpha q + 1$, avec q grand nombre premier et α petit, alors l'hypothèse DDH est raisonnable pour le sous-groupe de $\mathbb{Z}_p^\star$ à q éléments. Par conséquent, nous pouvons extraire jusqu'à $k = n/2 - (e + \log_2(n) + 2 + \log_2(\alpha))$ bits à partir d'une source d'aléa de min-entropie $n - \log_2(\alpha)$.

En pratique, les paramètres de sécurité sont souvent $n = 1024$, $e = 80$. Par conséquent, nous pouvons extraire approximativement $420 - \log_2(\alpha)$ bits en contrepartie de deux exponentiations modulo un entier de 1024 bits. Cela signifie que si on a besoin d'une chaîne de bits de 128 bits de long, le sous-groupe devrait avoir approximativement 2^{731} éléments.

5.3.2 Schémas de chiffrement

Schéma de chiffrement Elgamal [Elg84]. Dans le schéma de chiffrement Elgamal, le message doit être un élément du groupe cyclique $\mathbb{G}$ d'ordre q. Alice génère un élément aléatoire x dans $\mathbb{Z}_q$ qui sera la clef secrète et publie la clef publique $y = g^x$ où g est un générateur de $\mathbb{G}$. Pour chiffrer le message m, elle génère un élément aléatoire de $\mathbb{Z}_q^\star$ et calcule $(g^r, m \cdot y^r)$. Ce schéma est prouvé IND-CPA si $m \in \mathbb{G}$. Cependant en pratique les messages sont le plus souvent des chaînes de bits et non des éléments de $\mathbb{G}$. Une solution pour résoudre ce problème est d'extraire l'entropie de y^r et de xorer la chaîne de bits obtenue avec le message. De cette manière, le schéma de chiffrement est toujours IND-CPA. Notre extracteur peut être utilisé dans ce contexte pour extraire l'entropie.

Schéma de chiffrement de Cramer Shoup [CS98, Sho00b]. Le schéma de chiffrement de Cramer-Shoup est une amélioration du schéma de chiffrement Elgamal et est IND-CCA. Le principe est le même que dans Elgamal, on cache le message m en le multipliant avec un élément aléatoire h^r de $\mathbb{G}$. La preuve de sécurité nécesssite que m appartienne à $\mathbb{G}$. Afin d'utiliser des messages qui sont des chaînes de bits, nous pouvons utiliser la même solution : extraire l'entropie de h^r avec notre extracteur et xorer le résultat avec m.

5.4 Amélioration de l'efficacité

Dans cette section, nous allons examiner comment grâce à une hypothèse plus forte, l'hypothèse dite du logarithme discret avec exposant court ou DLSE (pour *Discrete Logarithm with*

Short Exponent), nous pouvons augmenter l'efficacité du protocole d'extraction de clefs. Initialement introduite dans [vOW96], cette hypothèse a été formalisée dans [PS98] et [GKR04] de la manière suivante :

Hypothèse 5.1 (s-DLSE [PS98])**.** *Soit s un entier, $\mathcal{G} = \{\mathbb{G}_n\}_n$ une famille de groupes cycliques et à chaque groupe $\mathbb{G}_n$ est associé un générateur g_n d'ordre $q_n > 2^n$. On dit que l'hypothèse s-DLSE est vérifiée dans $\mathcal{G}$ si pour toute machine de Turing probabiliste polynomiale I, pour chaque polynôme P et pour tout n suffisamment grand nous avons,* $\Pr_{x \in_R [\![1,\, 2^s]\!]} [I(g_n, q_n, s, g_n^x) = x] \leq 1/P(n)$.

Gennaro *et al.* ont prouvé dans [GKR04] que si les hypothèses s-DLSE et DDH sont vraies, les deux distributions suivantes sont indistinguables :

$$\left\{(g^x, g^y, Z) \middle|\ x, y \in_R [\![1,\, 2^s]\!],\ Z \in_R \mathbb{G}\right\} \text{ et } \left\{(g^x, g^y, g^{xy}) \middle|\ x, y \in_R [\![1,\, 2^s]\!]\right\}.$$

Ce resultat nous permet d'utiliser notre extracteur avec cette dernière distribution et ainsi d'être plus efficace. En effet, comme expliqué dans [GKR04], avec les connaissances actuelles, dans un groupe d'ordre premier, pour garantir un niveau de sécurité de 2^{-e}, il faut choisir $s \geq 2e$. Le paramètre usuel de sécurité est $e = 80$, soit $s \geq 160$. On est donc amené à choisir x et y uniformément dans $[\![1,\, 2^s]\!]$ ce qui conduit à un coût de calcul de (g^x, g^y, g^{xy}) plus faible que dans le cas où x et y sont choisis uniformément dans $\mathbb{G}$ avec $\log_2 |\mathbb{G}| = 731$.

6

Étude du mode cascade et de HMAC pour l'extraction de clefs

Sommaire

Au chapitre précédent, nous avons présenté un extracteur d'entropie déterministe qui peut être utilisé avec les éléments générés par un échange de clefs Diffie-Hellman, pour peu que l'hypothèse Diffie-Hellman décisionnelle soit vérifiée. Cet extracteur présente le grand avantage d'être simple à mettre en œuvre, mais souffre de ne pas pouvoir être utilisé avec n'importe quelle distribution de probabilité, ce qui est un inconvénient pour son utilisation dans un certain nombre de protocoles génériques. En effet, il est fréquent que les protocoles offrent aux utilisateurs plusieurs alternatives pour l'échange de clefs : soit via un échange Diffie-Hellman, soit via un échange basé sur l'algorithme RSA par exemple. La distribution de l'élément généré n'est donc pas la même dans ces deux cas et pour simplifier la programmation, les concepteurs de protocoles préfèrent utiliser une seule et même fonction d'extraction de clefs pour les deux distributions et non deux

fonctions différentes, une qui s'applique à l'élément Diffie-Hellman, comme c'est le cas de la fonction présentée dans le chapitre précédent, et une qui s'applique au secret issu de l'échange RSA.

Une solution simple pour l'extraction de clefs aurait été d'utiliser une famille de fonctions de hachage universelle, qui est un bon extracteur d'après le *leftover hash lemma*, et qui est relativement simple à mettre en place. Cependant, cette solution n'a pas été retenue pour deux raisons majeures : la première est qu'il était plus simple d'utiliser des fonctions déjà présentes dans les bibliothèques d'algorithmes, telles certaines fonctions de hachage ; la seconde est que certains protocoles sont anciens et, pour assurer la compatibilité avec les anciennes versions, on utilise des fonctions qui, même si elles ne sont pas prouvées sûres, n'ont pas révélé de défauts de sécurité.

Les concepteurs ont donc été amenés à créer des fonctions *ad hoc* et il existe actuellement beaucoup de constructions et de fonctions de dérivation de clefs différentes, et ce d'autant plus qu'à ce jour aucun consensus n'est encore apparu au sein de la communauté à ce sujet. C'est d'ailleurs ce qui motive la récente démarche de Krawczyk qui a présenté une fonction de dérivation de clefs [Kra] afin de créer une norme. Toutes ces fonctions d'extraction de clefs sont conçues à partir de fonctions de hachage et sont sûres au moins dans le modèle de l'oracle aléatoire. Cependant il serait intéressant d'étudier ces constructions dans un modèle moins fort, à savoir le modèle standard, en faisant reposer l'analyse sur des hypothèses aussi classiques et réalistes que possibles, telles certaines hypothèses calculatoires.

C'est dans cet esprit que Dodis *et al.* ont écrit le papier [DGH+04a]. Ils ont été les premiers à considérer l'extraction de clefs comme une étape importante du mécanisme de dérivation de clefs. Ils ont étudié comment des primitives classiques, telles que des MAC ou des fonctions de hachage, pouvaient être considérées comme des extracteurs d'entropie. Plus précisément, ils ont réduit la sécurité de l'extraction d'entropie à l'hypothèse que la fonction de compression se comporte comme un bon extracteur d'entropie, à savoir une famille de fonctions de hachage universelle. Nous nous plaçons ici dans cette voie de recherche avec la même motivation qu'eux : développer des méthodes et des outils pour pouvoir analyser des fonctions d'extraction de clefs utilisées en pratique dans certains protocoles célèbres. Plus particulièrement, notre objectif est l'étude de la fonction par défaut des protocoles IKE et TLS.

Datant du début des années 1990, ce dernier est actuellement un des protocoles les plus répandus pour sécuriser notamment la navigation sur internet, l'utilisation du courrier électronique et certains transferts de données. Nommé TLS (pour *Transport Layer Security*) depuis 2001 il est plus connu sous son ancien nom SSL (pour *Secure Socket Layer*). La version utilisée actuellement est la version 1.1 [DR06], mais une version 1.2 [DR08] est en cours d'élaboration. Les fonctions d'extraction de clefs dans les deux versions ne sont pas très différentes et les résultats établis pour l'un peuvent être transposés à l'autre. Pour cette raison, nous allons concentrer nos effort sur la version émergente de TLS, à savoir la version 1.2, même si celle-ci n'est pas encore la version officielle.

Pour mener à bien notre étude, nous allons développer une version calculatoire du célèbre *leftover hash lemma*, version grâce à laquelle nous prouvons qu'une bonne famille de fonctions pseudo-aléatoires peut être utilisée pour extraire de l'entropie, puisque la sortie obtenue est *calculatoirement indistinguable* d'une chaîne de bits parfaitement aléatoire. Cette outil est générique est peut être utilisé avec de nombreuses familles de fonctions pseudo-aléatoires.

Ensuite, nous allons appliquer cet outil pour analyser deux constructions basées sur HMAC. La première construction est celle utilisée pour IKE. La seconde est une version simplifiée de la version utilisée dans TLS et présente deux avantages : les résultats sont un peu plus généraux que ceux issus de l'analyse de TLS et ils sont facilement transposables à TLS puisqu'il suffit de calquer

les méthodes de preuve. D'ailleurs, nous donnons à la fin du chapitre les résultats obtenus pour TLS. Nous faisons remarquer que les principaux résultats de cette partie ont donné lieu à une publication [FPZ08].

6.1 Extracteur d'entropie calculatoire

Nous commençons notre étude en décrivant les nouveaux outils élaborés afin de permettre l'analyse de l'extraction d'entropie quand l'extracteur utilisé est un famille de fonctions pseudo-aléatoires. Nous adoptons un point de vue calculatoire et pour cela nous allons donc affaiblir plusieurs notions statistiques liées à l'entropie pour en énoncer des versions calculatoires. On pourra retrouver les versions statistiques de ces notions présentées dans l'introduction, section 2.2.

6.1.1 Définition

Un extracteur d'entropie calculatoire est l'extension de la notion d'extracteur d'entropie à la nuance près que, si l'entrée a suffisamment d'entropie, la sortie de l'extracteur doit être *calculatoirement* indistinguable d'une variable uniformément distribuée. Cette notion a déjà été utilisée de façon implicite dans [HILL99, DGH+04b].

Plus précisément, un extracteur d'entropie calculatoire ou cRE (pour *computational randomness extractor*) est une famille de fonctions cExt de $\{0,1\}^d \times \mathcal{D}om \times \mathcal{L}abel$ dans $\{0,1\}^t$ pour laquelle l'avantage d'un adversaire dans le jeu suivant est négligeable. Avant le début du jeu, le challenger choisit au hasard de façon uniforme un bit b et une chaîne de bits i de longueur d. Si $b = 0$, il choisit une fonction f au hasard dans $\mathcal{F}_{(\mathcal{D}om,\{0,1\}^t)}$, l'ensemble des fonctions de $\mathcal{D}om$ dans $\{0,1\}^t$, et affecte $\mathsf{ext} = f$. Si $b = 1$, il affecte $\mathsf{ext} = \mathsf{cExt}_i$, c'est-à-dire qu'il choisit la i^{e} fonction de la famille de fonctions. Le jeu commence alors et ce que l'on appelle l'adversaire cRE $\mathcal{A}$ envoie au challenger la description d'une distribution de probabilités $\mathcal{D}$ sur $\mathcal{D}om$ et éventuellement un label $label \in \mathcal{L}abel$. La distribution $\mathcal{D}$ doit avoir une min-entropie plus grande que m et doit pouvoir être échantillonnée efficacement. Le challenger choisit alors x selon la distribution $\mathcal{D}$ et envoie à l'adversaire $(i, \mathsf{ext}(x, label))$. Enfin, l'adversaire envoie un bit b' et gagne si $b = b'$. Son avantage en cRE, noté $\mathsf{adv}^{\mathsf{cre}}_{\mathsf{cExt}}(\mathcal{A})$, vaut :

$$\mathsf{adv}^{\mathsf{cre}}_{\mathsf{cExt}}(\mathcal{A}) = \left|\Pr[\mathsf{ext} \xleftarrow{\$} \mathsf{cExt} : \mathcal{A}^{\mathsf{ext}} \Rightarrow 1] - \Pr[\mathsf{ext} \xleftarrow{\$} \mathcal{F}_{(\mathcal{D}om,\{0,1\}^t)} : \mathcal{A}^{\mathsf{ext}} \Rightarrow 1]\right|.$$

Dans le cas où $\mathcal{A}$ est contraint à être sans préfixe (c'est-à-dire que $\mathcal{A}$ doit générer une distribution $\mathcal{D}$ sur $\mathcal{D}om$ telle que pour tout couple de chaînes de bits $(x, x') \in \mathcal{D}om \times \mathcal{D}om$ on a $\Pr\left[X \xleftarrow{\mathcal{D}} \mathcal{D}om : X = x\right] > 0$, $\Pr\left[X \xleftarrow{\mathcal{D}} \mathcal{D}om : X = x'\right] > 0$ et $x \sqsubset x'$ impliquent $x = x'$) alors on dit que $\mathcal{A}$ est un adversaire pf-cRE et on note $\mathsf{adv}^{\mathsf{pf-cre}}_{\mathsf{cExt}}(\mathcal{A})$ son avantage.

On attire l'attention du lecteur sur la particularité suivante : dans la définition d'extracteur d'entropie calculatoire que l'on donne ici, l'ensemble $\mathcal{D}om$ est *a priori* quelconque alors que pour les extracteurs d'entropie forts on restreint $\mathcal{D}om = \{0,1\}^n$ pour un certain n. Cette définition correspond à une réalité pratique : les extracteurs d'entropie calculatoires que l'on considèrera, et qui sont effectivement utilisés, sont des fonctions pseudo-aléatoires qui acceptent des entrées de taille variable. Il est évident que, de même qu'il faut que $n \geq t$ pour les extracteurs d'entropie forts, il faut que $\mathcal{D}om \geq 2^t$ pour que la notion d'extracteur d'entropie calculatoire ait un sens.

Cette notion d'extracteur d'entropie calculatoire implique directement la sécurité sémantique d'une clef générée avec un extracteur d'entropie déterministe à partir d'une source d'aléa à forte entropie dans la mesure où l'adversaire est passif. En effet, si un adversaire était capable de

casser la sécurité sémantique de la clef avec bonne probabilité, alors il serait capable avec la même probabilité de distinguer l'extracteur d'entropie calculatoire d'une fonction aléatoire, ce qui est exclu. Si on utilise des techniques d'authentification dans le protocole d'échange de clefs, alors la sécurité sémantique de la clef contre des adversaires actifs se réduit à la sécurité contre des adversaires passifs et donc, dans ce cadre, la notion d'extracteur d'entropie calculatoire implique directement la sécurité sémantique de la clef contre un adversaire actif.

6.1.2 Famille de fonctions de hachage calculatoirement presque universelle

Pour construire un extracteur d'entropie calculatoire nous allons établir une version calculatoire du *leftover hash lemma*. Pour cela nous devons d'abord introduire un équivalent calculatoire aux familles de fonctions de hachage presque universelles.

Soit $F : \mathcal{K}eys \times \mathcal{D}om \to \mathcal{R}ng$ une famille de fonctions. Nous généralisons dans cette partie la définition de [Bel06]. Une famille de fonctions de hachage est dite calculatoirement presque universelle si l'avantage d'un adversaire dans le jeu suivant est faible. Au début du jeu, ce que l'on appellera l'*adversaire cAU* $\mathcal{A}$ (pour *computationally almost universal*) génère une distribution efficacement échantillonable $\mathcal{D}$ sur $\mathcal{D}om \times \mathcal{D}om$ avec une min-entropie d'au moins m. Le challenger choisit une clef K au hasard selon la distribution uniforme et un couple (M_1, M_2) dans $\mathcal{D}om \times \mathcal{D}om$ selon la distribution $\mathcal{D}$ et l'adversaire gagne si $F_K(M_1) = F_K(M_2)$ alors que $M_1 \neq M_2$. L'avantage cAU de l'adversaire, noté $\mathsf{adv}_F^{\mathtt{cau}}(\mathcal{A})$, est défini par la formule :

$$\mathsf{adv}_F^{\mathtt{cau}}(\mathcal{A}) = \Pr\left[\begin{array}{c|c} \mathcal{A} \Rightarrow \mathcal{D};\ K \xleftarrow{\$} \mathcal{K}eys & F(K, M_1) = F(K, M_2) \\ (M_1, M_2) \xleftarrow{\mathcal{D}} \mathcal{D}om \times \mathcal{D}om & M_1 \neq M_2 \end{array}\right].$$

Dans le cas où $\mathcal{A}$ est contraint à être sans préfixe (c'est-à-dire que $\mathcal{A}$ doit générer une distribution $\mathcal{D}$ sur $\mathcal{D}om \times \mathcal{D}om$ telle que pour tout couple de chaînes de bits $(x, x') \in \mathcal{D}om \times \mathcal{D}om$ on a $\Pr\left[(X, X') \xleftarrow{\mathcal{D}} \mathcal{D}om \times \mathcal{D}om : (X, X') = (x, x')\right] > 0$ et $x \sqsubset x'$ impliquent $x = x'$) alors on dit que $\mathcal{A}$ est un adversaire pf-cAU et on note $\mathsf{adv}_F^{\mathtt{pf-cau}}(\mathcal{A})$ son avantage.

On notera que la définition de Bellare dans [Bel06] couvre le cas particulier où m, la min-entropie de $\mathcal{D}$, est égale à 0. Augmenter m revient à affaiblir l'hypothèse faite sur la famille de fonctions et donc quand $m \geq 1$ l'hypothèse introduite ici est plus faible que celle d'origine. En effet, si $\mathcal{A}$ est un adversaire cAU qui génère une distribution d'entropie $m \geq 1$ alors il est facile de construire un adversaire cAU $\mathcal{B}$ au sens de Bellare qui ait la même complexité temporelle que $\mathcal{A}$: l'adversaire $\mathcal{B}$ fait tourner A qui envoie une distribution $\mathcal{D}$, $\mathcal{B}$ choisit M_1, M_2 selon $\mathcal{D}$ et les envoie au challenger. La probabilité de succès de $\mathcal{B}$ est la même que celle de $\mathcal{A}$.

De manière générale, pour $m' \leq m$, tout adversaire cAU générant des distributions de min-entropie au moins m peut être transformé en un adversaire cAU générant des distributions de min-entropie au moins m' et ayant le même temps de calcul (l'adversaire cAU pour la min-entropie m' lance l'adversaire cAU pour la min-entropie m et fait suivre la distribution de probabilité $\mathcal{D}$ que ce dernier lui envoie).

6.1.3 *Leftover hash lemma* calculatoire

Cas classique

Nous sommes maintenant en mesure de présenter la version calculatoire du *leftover hash lemma* :

Lemme 6.1 (*Leftover hash lemma* calculatoire)**.** *Soit H une famille de fonctions de hachage de $\{0,1\}^n \times \mathcal{D}om$ dans $\{0,1\}^t$ telle que pour tout adversaire cAU $\mathcal{B}$ dont le temps de calcul est au plus T et qui produit une distribution sur $\mathcal{D}om \times \mathcal{D}om$ de min-entropie au moins $2m$, on a $\mathsf{adv}_H^{\mathsf{cau}}(\mathcal{B}) \leq 1/2^t + \varepsilon$. On a alors que pour tout adversaire $\mathcal{A}$ de temps de calcul au plus T et produisant une distribution de min-entropie au moins m :*

$$\mathsf{adv}_H^{\mathsf{cre}}(\mathcal{A}) \leq \sqrt{2^t \cdot (2^{-m} + \varepsilon)}.$$

Démonstration. La preuve de ce lemme est calquée sur celle du *leftover hash lemma* statistique, c'est-à-dire du lemme 2.1 présenté en introduction. Soit $\mathcal{A}$ un adversaire cRE contre H qui génère une distribution de probabilité sur $\mathcal{D}om$ de min-entropie au moins m et soit E_1 et E_2 les distributions de probabilité sur $\{0,1\}^n \times \mathcal{D}om$ suivantes :

$$E_1 = \left\{ \begin{array}{c} K \leftarrow \mathcal{U}_{\mathcal{K}eys}, \mathcal{A} \Rightarrow \mathcal{D}_{\mathcal{A}}, \\ X \leftarrow \mathcal{D}_{\mathcal{A}}, Y = H(K,X) \end{array} \right\}, \; E_2 = \left\{ \begin{array}{c} K \leftarrow \mathcal{U}_{\mathcal{K}eys}, \mathcal{A} \Rightarrow \mathcal{D}_{\mathcal{A}}, \\ Y \leftarrow \mathcal{U}_{\mathcal{R}ng} \end{array} \right\}.$$

Soit $\delta = \mathsf{adv}_H^{\mathsf{cre}}(\mathcal{A})$, par définition on a :

$$\delta = \left| \Pr\left[(K,Y) \leftarrow E_1 \colon \mathcal{A}(K,Y) \Rightarrow 1\right] - \Pr\left[(K,Y) \leftarrow E_2 \colon \mathcal{A}(K,Y) \Rightarrow 1\right] \right|.$$

Pour prouver ce lemme, nous montrons successivement que :

$$\frac{1+\delta^2}{2^n \cdot 2^t} \;\leq\; \Pr\left[\begin{array}{c|c} (K,Y) \leftarrow E_1 & K = K' \\ (K',Y') \leftarrow E_1 & Y = Y' \end{array} \right] \tag{6.1}$$

$$\Pr\left[\begin{array}{c|c} (K,Y) \leftarrow E_1 & K = K' \\ (K',Y') \leftarrow E_1 & Y = Y' \end{array} \right] \;\leq\; \frac{1}{2^n}\left(2^{-m} + \frac{1}{2^t} + \varepsilon\right) \tag{6.2}$$

Une fois établis ces deux résultats, nous pourrons conclure facilement.

Commençons par rappeler que si $(\alpha_x)_{x \in \mathcal{X}}$ est une suite telle que $\sum_{x \in \mathcal{X}} \alpha_x = 1$ alors on a $\sum_{x \in \mathcal{X}} \alpha_x^2 \geq 1/|\mathcal{X}|$. Cette inégalité simple est démontrée dans le lemme 2.2 présenté en introduction. Elle permet de prouver facilement la première inégalité dans le cas où $\delta = 0$, puisque

$$\Pr\left[\begin{array}{c|c} (K,Y) \leftarrow E_1 & K = K' \\ (K',Y') \leftarrow E_1 & Y = Y' \end{array} \right] = \sum_{k,y} \Pr\left[(K,Y) \leftarrow E_1 \colon \begin{array}{c} K = k \\ Y = y \end{array} \right]^2.$$

Pour prouver le résultat (6.1), nous pouvons donc supposer que $\delta \neq 0$. Remarquons d'abord que δ est égal à :

$$\delta = \left| \sum_{y,k} \Pr\left[(K,Y) \leftarrow E_1 \colon \begin{array}{c} \mathcal{A}(k,y) \Rightarrow 1 \\ K = k \\ Y = y \end{array} \right] - \Pr\left[(K,Y) \leftarrow E_2 \colon \begin{array}{c} \mathcal{A}(k,y) \Rightarrow 1 \\ K = k \\ Y = y \end{array} \right] \right|.$$

La *manière* dont (k,y) est choisi est indépendante de l'événement $\mathcal{A}(k,y) \Rightarrow 1$ avec $\mathcal{A} \Rightarrow \mathcal{D}_{\mathcal{A}}$, ce qui peut être utilisé pour reformuler δ de la façon suivante :

$$\delta = \left| \sum_{y,k} \Pr[\mathcal{A} \Rightarrow \mathcal{D}_{\mathcal{A}} \colon \mathcal{A}(k,y) \Rightarrow 1] \cdot \left(\Pr\left[(K,Y) \leftarrow E_1 \colon \begin{array}{c} K = k \\ Y = y \end{array} \right] - \frac{1}{2^n \cdot 2^t} \right) \right|.$$

Introduisons la notation $q_{k,y}$ définie par :

$$q_{k,y} = \frac{1}{\delta} \cdot \left| \Pr[\mathcal{A} \Rightarrow \mathcal{D}_{\mathcal{A}} \colon \mathcal{A}(k,y) \Rightarrow 1] \cdot \left(\Pr\left[(K,Y) \leftarrow E_1 \colon \begin{array}{c} K = k \\ Y = y \end{array} \right] - \frac{1}{2^n \cdot 2^t} \right) \right|.$$

Puisque $\sum_{k,y} q_{k,y} = 1$, d'après la petite inégalité rappelée ci-dessus, on a $\frac{1}{2^n \cdot 2^t} \leq \sum_{k,y} q_{k,y}^2$ et donc :

$$\begin{aligned}\frac{1}{2^n \cdot 2^t} &\leq \frac{1}{\delta^2} \sum_{k,y} \left(\Pr\left[(K,Y) \leftarrow E_1 \colon K = k \wedge Y = y\right] - \frac{1}{2^n \cdot 2^t} \right)^2 \\ &\leq \frac{1}{\delta^2} \left(\sum_{k,y} \Pr\left[(K,Y) \leftarrow E_1 \colon K = k \wedge Y = y\right]^2 - \frac{1}{2^n \cdot 2^t} \right).\end{aligned}$$

Il en résulte directement l'inéquation (6.1) :

$$\frac{1+\delta^2}{2^n \cdot 2^t} \leq \sum_{k,y} \Pr\left[(K,Y) \leftarrow E_1 \colon K = k \wedge Y = y\right]^2 = \Pr\left[\begin{array}{c} (K,Y) \leftarrow E_1 \\ (K',Y') \leftarrow E_1 \end{array} \colon K = K' \wedge Y = Y' \right]$$

Nous allons maintenant prouver l'inégalité (6.2), ce qui revient à majorer la probabilité de collision $\Pr\left[E \colon K = K' \wedge Y = Y'\right]$. Cette dernière est égale à

$$\Pr\left[\begin{array}{c} K \leftarrow \mathcal{U}_{\mathcal{K}eys} \\ K' \leftarrow \mathcal{U}_{\mathcal{K}eys} \end{array} \colon K = K' \right] \cdot \Pr\left[\begin{array}{c} \mathcal{A} \Rightarrow \mathcal{D}_{\mathcal{A}}, X \leftarrow \mathcal{D}_{\mathcal{A}}, \\ \mathcal{A} \Rightarrow \mathcal{D}'_{\mathcal{A}}, X' \leftarrow \mathcal{D}'_{\mathcal{A}}, \\ K \leftarrow \mathcal{U}_{\mathcal{K}eys} \end{array} \colon H(K,X) = H(K,X') \right].$$

Puisque

$$\Pr\left[\begin{array}{c} K \leftarrow \mathcal{U}_{\mathcal{K}eys} \\ K' \leftarrow \mathcal{U}_{\mathcal{K}eys} \end{array} \colon K = K' \right] = \frac{1}{2^n},$$

on est finalement amené à majorer :

$$\Pr\left[\begin{array}{c} \mathcal{A} \Rightarrow \mathcal{D}_{\mathcal{A}}, X \leftarrow \mathcal{D}_{\mathcal{A}}, \\ \mathcal{A} \Rightarrow \mathcal{D}'_{\mathcal{A}}, X' \leftarrow \mathcal{D}'_{\mathcal{A}}, \end{array} \colon X = X' \right] + \Pr\left[\begin{array}{c} \mathcal{A} \Rightarrow \mathcal{D}_{\mathcal{A}}, X \leftarrow \mathcal{D}_{\mathcal{A}}, \\ \mathcal{A} \Rightarrow \mathcal{D}'_{\mathcal{A}}, X' \leftarrow \mathcal{D}'_{\mathcal{A}}, \\ K \leftarrow \mathcal{U}_{\mathcal{K}eys} \end{array} \colon \begin{array}{c} X \neq X' \\ H(K,X) = H(K,X') \end{array} \right]. \tag{6.3}$$

Nous allons borner chacun des deux termes indépendamment en commençant par le second. Dans la suite de cette preuve nous noterons $\mathcal{D}_{\mathcal{A}}$ et $\mathcal{D}'_{\mathcal{A}}$ deux distributions données par deux exécutions indépendantes de $\mathcal{A}$ et par $\mathcal{D}_{\mathcal{B}}$ la distribution produit $\mathcal{D}_{\mathcal{A}} \times \mathcal{D}'_{\mathcal{A}}$. Soit $\mathcal{B}$ l'adversaire cAU suivant : $\mathcal{B}$ exécute $\mathcal{A}$ deux fois indépendamment en jouant le rôle du challenger pour $\mathcal{A}$. Si $\mathcal{D}_{\mathcal{A}}$ et $\mathcal{D}'_{\mathcal{A}}$ sont les deux distributions renvoyées par $\mathcal{A}$ alors $\mathcal{B}$ renvoie au challenger $\mathcal{D}_{\mathcal{B}}$ définie ci-dessus. Puisque $\mathcal{D}_{\mathcal{A}}$ et $\mathcal{D}'_{\mathcal{A}}$ ont toutes deux une min-entropie supérieure à m, la min-entropie de $\mathcal{D}_{\mathcal{B}}$ est supérieure à $2m$.

La complexité temporelle de $\mathcal{B}$ est égale à $2T_A + \mathcal{O}(1)$ et son avantage cAU est exactement :

$$\Pr\left[\begin{array}{c} \mathcal{A} \Rightarrow \mathcal{D}_{\mathcal{A}}, X \leftarrow \mathcal{D}_{\mathcal{A}}, \\ \mathcal{A} \Rightarrow \mathcal{D}'_{\mathcal{A}}, X' \leftarrow \mathcal{D}'_{\mathcal{A}}, \\ K \leftarrow \mathcal{U}_{\mathcal{K}eys} \end{array} \colon \begin{array}{c} X \neq X' \\ H(K,X) = H(K,X') \end{array} \right].$$

Par hypothèse, cet avantage est majoré par $1/2^t + \varepsilon$, et donc $\Pr\left[E \colon K = K' \wedge Y = Y'\right]$ est majorée par :

$$\frac{1}{2^n} \left(\Pr\left[\begin{array}{c} \mathcal{A} \Rightarrow \mathcal{D}_{\mathcal{A}}, X \leftarrow \mathcal{D}_{\mathcal{A}}, \\ \mathcal{A} \Rightarrow \mathcal{D}'_{\mathcal{A}}, X' \leftarrow \mathcal{D}'_{\mathcal{A}}, \end{array} \colon X = X' \right] + \frac{1}{2^t} + \varepsilon \right).$$

Nous allons maintenant borner le premier terme de l'équation (6.3). Pour tout couple de distributions $\mathcal{D}_\mathcal{A}$ et $\mathcal{D}'_\mathcal{A}$ *fixées* de min-entropie au moins m, la probabilité $\Pr\left[\begin{array}{l} X \leftarrow \mathcal{D}_\mathcal{A}, \\ X' \leftarrow \mathcal{D}'_\mathcal{A}, \end{array} : X = X'\right]$ est égale à :

$$\sum_x \Pr\left[\ X \leftarrow \mathcal{D}_\mathcal{A}\ : X = x\right] \Pr\left[\ X' \leftarrow \mathcal{D}'_\mathcal{A},\ : X' = x\right].$$

En appliquant l'inégalité de Cauchy-Schwarz, on obtient que ceci est inférieur à :

$$\sqrt{\sum_x \Pr\left[X \leftarrow \mathcal{D}_\mathcal{A} \colon X = x\right]^2} \sqrt{\sum_x \Pr\left[X \leftarrow \mathcal{D}'_\mathcal{A} \colon X' = x\right]^2}$$

Puisque $\mathcal{D}_\mathcal{A}$ et $\mathcal{D}_\mathcal{A}$ ont une min-entropie d'au moins m, ceci est plus petit que $2^{-m/2} \cdot 2^{-m/2} \leq 2^{-m}$. Puisque ceci est vrai pour *tous* les couples de distributions $\mathcal{D}_\mathcal{A}$ et $\mathcal{D}'_\mathcal{A}$ de min-entropie au moins m, cela reste vrai pour $\Pr\left[\begin{array}{l} \mathcal{A} \Rightarrow \mathcal{D}_\mathcal{A}, X \leftarrow \mathcal{D}_\mathcal{A}, \\ \mathcal{A} \Rightarrow \mathcal{D}'_\mathcal{A}, X' \leftarrow \mathcal{D}'_\mathcal{A}, \end{array} : X = X'\right]$ où $\mathcal{D}_\mathcal{A}$ et $\mathcal{D}'_\mathcal{A}$ sont générées aléatoirement par l'adversaire, et finalement on a donc :

$$\Pr\left[E \colon K = K' \wedge Y = Y'\right] \quad \leq \quad \frac{1}{2^n}\left(2^{-m} + \frac{1}{2^t} + \varepsilon\right).$$

En combinant cette dernière inégalité avec l'équation 6.1 on obtient l'inégalité :

$$\frac{1+\delta^2}{2^n \cdot 2^t} \quad \leq \quad \frac{1}{2^n}\left(2^{-m} + \frac{1}{2^t} + \varepsilon\right).$$

D'où l'on en déduit immédiatement l'inégalité :

$$\delta \leq \sqrt{2^t\,(2^{-m} + \varepsilon)}.$$

C'est le résultat attendu. □

On remarque que l'on peut imposer une sécurité en 2^{-e}, en imposant que :

$$\min(\log_2(1/\varepsilon), m) \geq t + 2e + 1.$$

Cas sans préfixe

En pratique, il va s'avérer parfois utile d'utiliser une autre version du *leftover hash lemma* calculatoire, dans le cas particulier où la famille de fonctions de hachage presque universelle utilisée n'est résistante qu'aux adversaires sans préfixe. Par exemple, nous allons appliquer ce résultat avec la construction cascade qui n'est résistante que contre les adversaires sans préfixe. Comme on peut le constater, l'énoncé est très similaire à celui de la version d'origine et le théorème n'est que légèrement plus faible. Cela n'est pas étonnant que le résultat soit légèrement plus faible, puisque l'on utilise une fonction plus faible pour effectuer l'extraction de clef. Cependant, la preuve, elle, est plus technique que la preuve du *leftover hash lemma* calculatoire.

Lemme 6.2 (*Leftover hash lemma* calculatoire sans préfixe)**.** *Soit H une famille de fonctions de hachage de $\{0,1\}^n \times \mathcal{D}om$ dans $\{0,1\}^t$ telle que pour tout adversaire pf-cAU $\mathcal{B}$, dont le temps de calcul est au plus T et produit une distribution sans préfixe sur $\mathcal{D}om \times \mathcal{D}om$ de min-entropie au moins $2m-1$, on a $\mathsf{adv}_H^{\mathtt{pf-cau}}(\mathcal{B}) \leq 1/2^t + \varepsilon$. On a alors que pour tout adversaire pf-cRE A de temps de calcul au plus T et produisant une distribution de min-entropie au moins m :*

$$\mathsf{adv}_H^{\mathtt{pf\text{-}cre}}(\mathcal{A}) \leq \sqrt{2^t \cdot (3 \cdot 2^{-m} + \varepsilon)}.$$

Démonstration. La preuve est similaire à la preuve de la partie précédente, exceptée la manière dont l'adversaire cAU $\mathcal{B}$ est simulé. On se contente donc de décrire cet aspect de la preuve.

On souligne que dans cette preuve A est un adversaire cRE *sans préfixe* et on veut construire un adversaire cAU $\mathcal{B}$ qui soit lui aussi sans préfixe.

Soit $\mathcal{D}_{\mathcal{A}}$ et $\mathcal{D}'_{\mathcal{A}}$ deux distributions données par deux exécutions indépendantes de $\mathcal{A}$ et par $\mathcal{D}_\pi$ la distribution produit $\mathcal{D}_{\mathcal{A}} \times \mathcal{D}'_{\mathcal{A}}$. On fait d'abord remarquer que puisque $\mathcal{D}_{\mathcal{A}}$ peut être différente de $\mathcal{D}'_{\mathcal{A}}$, pour chaque $X \leftarrow \mathcal{D}_{\mathcal{A}}$ et $X' \leftarrow \mathcal{D}'_{\mathcal{A}}$, rien n'interdit que $X \sqsubset X'$ ou $X' \sqsubset X$, $\mathcal{D}_\pi$ n'est donc pas forcément sans préfixe. Par conséquent, on ne peut pas utiliser un adversaire $\mathcal{B}$ qui génère la distribution de probabilité $\mathcal{D}_\pi$, comme c'est le cas dans la distribution précédente. Nous construisons donc la distribution de probabilité $\mathcal{D}_{\mathcal{B}}$ de la façon suivante : choisir (X, X') suivant $\mathcal{D}_\pi$, si $X \sqsubset X'$ ou $X' \sqsubset X$, alors choisir Y selon $\mathcal{D}_{\mathcal{A}}$ dans $\mathcal{D}om$ et sortir (Y, Y), sinon sortir (X, X'). Montrons maintenant que cette distribution, qui est efficacement échantillonable car $\mathcal{D}_{\mathcal{A}}$ et $\mathcal{D}'_{\mathcal{A}}$ le sont, a une min-entropie d'au moins $2m - 1$.

Pour chaque x' tel que $\Pr_{\mathcal{D}'_a}[X' = x'] > 0$, il y a au plus un x tel que $x \sqsubset x'$ et $\Pr_{\mathcal{D}_a}[X = x] > 0$. En effet, s'il existe x_1 et x_2 tels que $x_1 \sqsubset x'$, $x_2 \sqsubset x'$, $\Pr_{\mathcal{D}_a}[X = x_2] > 0$ et $\Pr_{\mathcal{D}_a}[X = x_1] > 0$ on a alors $x_1 \sqsubset x_2$ ou $x_2 \sqsubset x_1$ et par définition des adversaires sans préfixe $x_1 = x_2$.

Donc, pour tout x' tel que $\Pr_{\mathcal{D}'_a}[X' = x'] > 0$, il y a au plus un terme non nul dans la somme $\Pr_{\mathcal{D}_a}[X \sqsubset x'] = \sum_{x \sqsubset x'} \Pr_{\mathcal{D}_a}[X = x]$ et cette dernière est donc majorée par 2^{-m}. De ce fait, $\Pr[X \sqsubset X'] = \sum_{x'} \Pr_{\mathcal{D}_a}[X \sqsubset x'] \cdot \Pr_{\mathcal{D}'_a}[X' = x'] \leq 2^{-m} \cdot \sum_{x'} \Pr_{\mathcal{D}'_a}[X' = x'] = 2^{-m}$. La probabilité d'obtenir un couple (X, X') tel que $X \sqsubset X'$ ou $X' \sqsubset X$ est donc plus petite que $2 \cdot 2^{-m}$.

Soit y un élément de $\mathcal{D}om$. La probabilité que $\mathcal{B}$ renvoie le couple (y, y) est égale à la probabilité $\Pr[((X \sqsubset X') \wedge (X' \sqsubset X)) \vee Y = y]$ qui, d'après ce qui précède, est majorée par $\Pr[((X \sqsubset X') \wedge (X' \sqsubset X))] \Pr[Y \leftarrow \mathcal{D}_{\mathcal{A}} : Y = y] \leq 2 \cdot 2^{-m} \cdot 2^{-m}$, soit par $2^{-(2m-1)}$. Par ailleurs, pour tout couple (x, x') tel que $x \not\sqsubset x'$ et $x' \not\sqsubset x$, la probabilité que $\mathcal{B}$ renvoie (x, x') est inférieure à 2^{-2m}. Enfin, pour tout couple (x, x') tel que $x \neq x'$ et que $x \sqsubset x'$ ou $x' \sqsubset x$, la probabilité que $\mathcal{B}$ renvoie (x, x') vaut 0. La min-entropie de $\mathcal{D}_{\mathcal{B}}$ est donc supérieure à $2m - 1$ et $\mathcal{D}_{\mathcal{B}}$ est sans préfixe par construction.

La complexité temporelle de $\mathcal{B}$ est $2T_{\mathcal{A}} + \mathcal{O}(1)$ et son avantage cAU est exactement :

$$\mathsf{adv}_H^{\mathsf{cau}}(\mathcal{B}) = \Pr\left[\begin{array}{c|c} \mathcal{A} \Rightarrow \mathcal{D}_{\mathcal{A}}, \mathcal{A} \Rightarrow \mathcal{D}'_{\mathcal{A}}, & X \neq X' \\ (X, X') \leftarrow \mathcal{D}_{\mathcal{B}}, & H(K, X) = H(K, X') \\ K \leftarrow \mathcal{U}_{\mathcal{K}eys} & \end{array}\right].$$

On insiste sur le fait que dans cette probabilité le couple (X, X') est choisi selon $\mathcal{D}_{\mathcal{B}}$. Cet avantage est égal à :

$$\sum_{\substack{(x,x') \\ x \not\sqsubset x' \\ x' \not\sqsubset x}} \Pr\left[K \leftarrow \mathcal{U}_{\mathcal{K}eys} : \ H(K, x) = H(K, x')\ \right] \cdot \Pr_{\mathcal{D}_{\mathcal{B}}}\left[(X, X') = (x, x')\right].$$

Si (x, x') est un couple où aucun élément n'est préfixe de l'autre, $\Pr_{\mathcal{D}_\pi}[(X, X') = (x, x')]$ est égale à $\Pr_{\mathcal{D}_{\mathcal{B}}}[(X, X') = (x, x')]$, et donc la formule précédente devient

$$\begin{aligned} &\textstyle\sum_{(x,x')} \Pr\left[K \leftarrow \mathcal{U}_{\mathcal{K}eys} : \begin{array}{c} x \neq x' \\ H(K, x) = H(K, x') \end{array}\right] \cdot \Pr_{\mathcal{D}_\pi}\left[(X, X') = (x, x')\right] \\ -&\sum_{\substack{(x,x') \\ x \sqsubset x' \\ ou\ x' \sqsubset x}} \Pr\left[K \leftarrow \mathcal{U}_{\mathcal{K}eys} : \begin{array}{c} x \neq x' \\ H(K, x) = H(K, x') \end{array}\right] \cdot \Pr_{\mathcal{D}_\pi}\left[(X, X') = (x, x')\right]. \end{aligned}$$

La probabilité, où (X, X') est choisi selon $\mathcal{D}_\pi$,

$$\Pr\left[\begin{array}{cc} \mathcal{A} \Rightarrow \mathcal{D}_{\mathcal{A}}, \mathcal{A} \Rightarrow \mathcal{D}'_{\mathcal{A}}, & \\ (X, X') \leftarrow \mathcal{D}_\pi, & : \quad \begin{array}{c} X \neq X' \\ H(K, X) = H(K, X') \end{array} \\ K \leftarrow \mathcal{U}_{\mathcal{K}eys} & \end{array}\right]$$

est donc égale à

$$\sum_{\substack{(x,x') \\ x \sqsubset x' \text{ ou } x' \sqsubset x}} \Pr\left[K \leftarrow \mathcal{U}_{\mathcal{K}eys} : \begin{array}{c} x \neq x' \\ H(K,x) = H(K,x') \end{array}\right] \cdot \Pr_{\mathcal{D}_\pi}\left[(X, X') = (x, x')\right] + \mathsf{adv}_H^{\mathsf{cau}}(\mathcal{B}).$$

Le premier terme de la somme est plus petit que

$$\sum_{\substack{x \neq x' \\ x \sqsubset x' \text{ ou } x' \sqsubset x}} \Pr_{\mathcal{D}_\pi}\left[(X, X') = (x, x')\right] = \Pr_{\mathcal{D}_\pi}[((X \sqsubset X') \vee (X' \sqsubset X)) \wedge (X \neq X')] \leq 2 \cdot 2^{-m}.$$

De plus, par hypothèse, l'avantage de $\mathcal{B}$ est majoré par $1/2^t + \varepsilon$. Par conséquent, on a établi que :

$$\Pr\left[\begin{array}{cc} \mathcal{A} \Rightarrow \mathcal{D}_{\mathcal{A}}, X \leftarrow \mathcal{D}_{\mathcal{A}} & \\ \mathcal{A} \Rightarrow \mathcal{D}'_{\mathcal{A}}, X' \leftarrow \mathcal{D}'_{\mathcal{A}}, & : \quad \begin{array}{c} X \neq X' \\ H(K, X) = H(K, X') \end{array} \\ K \leftarrow \mathcal{U}_{\mathcal{K}eys} & \end{array}\right] \leq 2 \cdot 2^{-m} + 2^{-t} + \varepsilon.$$

Une fois cette majoration établie, on termine la preuve de façon identique au cas classique. □

6.1.4 Des fonctions pseudo-aléatoires aux extracteurs d'entropie calculatoires

Nous allons maintenant utiliser les deux résultats précédents pour montrer qu'une famille de fonctions pseudo-aléatoires est un bon extracteur d'entropie. Cela provient du fait qu'une famille de fonctions pseudo-aléatoires est aussi une famille de fonctions de hachage presque universelle calculatoire, comme nous allons le montrer. Encore une fois, nous séparons le cas où les familles de fonctions ne résistent qu'à des adversaires sans préfixe, du cas classiques où elles résistent à tous les adversaires.

Cas classique

Nous montrons d'abord qu'une famille de fonctions pseudo-aléatoires est une famille de fonctions de hachage presque universelle calculatoire.

Lemme 6.3 (PRF ⇒ cAU)**.** *Soit $F : \{0,1\}^n \times \mathcal{D}om \to \{0,1\}^t$ une famille de fonctions pseudo-aléatoires. Soit $\mathcal{A}$ un adversaire cAU contre F qui produit une distribution ayant une min-entropie d'au moins m et qui a une complexité en temps de T. Alors, il existe un adversaire PRF $\mathcal{B}$ tel que :*

$$\mathsf{adv}_F^{\mathsf{cau}}(\mathcal{A}) \leq \mathsf{adv}_F^{\mathtt{prf}}(\mathcal{B}) + \frac{1}{2^t},$$

où $\mathcal{B}$ a une complexité en temps en $\mathcal{O}(T)$ et fait au plus 2 requêtes.

Démonstration. Soit $\mathcal{B}$ l'adversaire PRF contre F suivant. Tout d'abord, $\mathcal{B}$ lance $\mathcal{A}$ qui lui envoie une distribution $\mathcal{D}_{\mathcal{A}}$ sue $\mathcal{D}om \times \mathcal{D}om$ de min-entropie au moins m. Il choisit alors (M_1, M_2) selon

la distribution $\mathcal{D}_{\mathcal{A}}$ et fait suivre M_1 puis M_2 au challenger. Ce dernier répond par C_1 puis C_2 successivement. Si $C_1 == C_2$ alors $\mathcal{B}$ renvoye '1', sinon il renvoye '0'.

Si le challenger génère C_1 et C_2 en utilisant un oracle aléatoire, alors la probabilité que $\mathcal{B}$ renvoie 1 est égale à $1/2^t$. Si le challenger utilise F_K, avec une clef K choisie uniformément dans $\{0,1\}^n$, alors la probabilité que $\mathcal{B}$ renvoie 1 est égale à $\mathsf{adv}_F^{\mathtt{cau}}(\mathcal{A})$. L'avantage de $\mathcal{B}$ est donc égal à $\mathsf{adv}_F^{\mathtt{prf}}(\mathcal{B}) = \left|\mathsf{adv}_F^{\mathtt{cau}}(\mathcal{A}) - 2^{-t}\right|$. Comme $\mathcal{B}$ fait au plus 2 requêtes et qu'il a presque le même temps de calcul que $\mathcal{A}$, la fin de la preuve est aisée. □

Nous pouvons maintenant établir une conséquence essentielle de ce résultat.

Corollaire 6.1 (PRF⇒cRE)**.** *Soit $F : \{0,1\}^n \times \mathcal{D}om \to \{0,1\}^t$ une famille de fonctions pseudo-aléatoires. Soit $\mathcal{A}$ un adversaire cRE contre F qui produit une distribution ayant une min-entropie d'au moins m et qui a une complexité en temps de T. Alors, il existe un adversaire PRF $\mathcal{B}$ tel que :*

$$\mathsf{adv}_F^{\mathtt{cre}}(\mathcal{A}) \leq \sqrt{2^t \cdot (2^{-m} + \mathsf{adv}_F^{\mathtt{prf}}(\mathcal{B}))},$$

où $\mathcal{B}$ a une complexité en temps en $\mathcal{O}(T)$ et fait au plus 2 requêtes.

Démonstration. La preuve est la conséquence directe du lemme 6.1 et du lemme 6.3. □

En pratique, pour imposer une sécurité cRE en 2^{-e+1}, on impose à la fois que $t + 2e \leq m$ et que $\mathsf{adv}_F^{\mathtt{prf}}(\mathcal{B}) \leq 2^{-(2e+t)}$. Il est intéressant de remarquer que cette dernière inégalité impose une condition sur la taille des clefs de F. En effet, plaçons nous dans le cas idéal où toutes les fonctions de F ont été choisies parfaitement aléatoirement dans l'ensemble des fonctions de $\{0,1\}^n \times \mathcal{D}om$ dans $\{0,1\}^t$. Considérons l'adversaire PRF $\mathcal{A}$ suivant : $\mathcal{A}$ choisit au hasard $(x_1, x_2) \in \{0,1\}^b$, envoie ce couple au challenger qui lui répond par (y_1, y_2). Il choisit alors une clef K et teste si $F(K, x_1)$ est égale à y_1 et si $F(K, x_2)$ est égale à y_2. Si c'est vrai, il retourne '1' à la fin de l'attaque, sinon il retourne '0'. Sa complexité temporelle est de $2T_F$ (où T_F désigne le temps de calcul de F), il fait deux requêtes et son avantage en PRF est égal à $2^{-n}(1 - 2^{-2t})$ qui vaut à peu près 2^{-n}. Même dans le cas idéal, on trouve un adversaire à faible temps de calcul dont l'avantage est approximativement égal à 2^{-n}, donc le meilleur adversaire contre une PRF non idéale à un avantage supérieur à 2^{-n}. Pour cette raison, on suppose que l'adversaire $\mathcal{B}$ du théorème précédent à un avantage $\mathsf{adv}_F^{\mathtt{prf}}(\mathcal{B}) \geq 2^{-n}$. Pour obtenir une sécurité cRE en 2^{-e+1}, il faut donc que $2^{-n} \leq 2^{-(2e+t)}$, c'est-à-dire que $n \geq 2e + t$. Cette inégalité a de lourdes conséquences sur les schémas que l'on peut analyser en pratique, comme nous le verrons dans la suite.

Cas sans préfixe

Nous allons procéder comme précédemment, pour établir les résultats équivalents aux précédents, dans le cas sans préfixe. Encore une fois, ces résultats montrent tout leur intérêt lorsque nous les appliquons au mode cascade.

Lemme 6.4. *Soit $F : \{0,1\}^n \times \mathcal{D}om \to \{0,1\}^t$ une famille de fonctions pseudo-aléatoires. Soit $\mathcal{A}$ un adversaire pf-cAU contre F qui produit une distribution ayant une min-entropie d'au moins m et qui a une complexité en temps de T. Alors, il existe un adversaire pf-PRF $\mathcal{B}$ tel que :*

$$\mathsf{adv}_F^{\mathtt{pf-cau}}(\mathcal{A}) \leq \mathsf{adv}_F^{\mathtt{pf-prf}}(\mathcal{B}) + \frac{1}{2^t},$$

où $\mathcal{B}$ a une complexité en temps en $\mathcal{O}(T)$ et fait au plus 2 requêtes.

Démonstration. La preuve est en tout point similaire au cas sans préfixe. Soit $\mathcal{B}$ l'adversaire PRF contre F suivant. Tout d'abord, $\mathcal{B}$ lance $\mathcal{A}$ qui lui envoie une distribution $\mathcal{D}_{\mathcal{A}}$ sur $\mathcal{D}om \times \mathcal{D}om$ de min-entropie au moins m. Il choisit alors (M_1, M_2) selon la distribution $\mathcal{D}_{\mathcal{A}}$ et fait suivre M_1 puis M_2 au challenger. Ce dernier répond par C_1 puis C_2 successivement. Si $C_1 == C_2$ alors $\mathcal{B}$ renvoie '1', sinon il renvoye '0'. On remarquera que comme $\mathcal{A}$ est sans préfixe, $\mathcal{B}$ l'est aussi.

Si le challenger génère C_1 et C_2 en utilisant un oracle aléatoire, alors la probabilité que $\mathcal{B}$ renvoie 1 est égale à $1/2^t$. Si le challenger utilise F_K, avec une clef K choisie uniformément dans $\{0,1\}^n$, alors la probabilité que $\mathcal{B}$ renvoie 1 est égale à $\mathsf{adv}_F^{\mathtt{pf-cau}}(\mathcal{A})$. L'avantage de $\mathcal{B}$ est donc égal à $\mathsf{adv}_F^{\mathtt{pf-prf}}(\mathcal{B}) = \left|\mathsf{adv}_F^{\mathtt{pf-cau}}(\mathcal{A}) - 2^{-t}\right|$. Comme $\mathcal{B}$ fait au plus 2 requêtes et qu'il a presque le même temps de calcul que $\mathcal{A}$, la fin de la preuve est aisée. □

Corollaire 6.2 (pf-PRF⇒pf-cRE)**.** *Soit $F : \{0,1\}^n \times \mathcal{D}om \to \{0,1\}^t$ une famille de fonctions pseudo-aléatoires. Soit $\mathcal{A}$ un adversaire pf-cRE contre F qui produit une distribution sans préfixe ayant une min-entropie d'au moins m et qui a une complexité en temps de T. Alors, il existe un adversaire pf-PRF $\mathcal{B}$ tel que :*

$$\mathsf{adv}_F^{\mathtt{pf-cre}}(\mathcal{A}) \leq \sqrt{2^t \cdot (3 \cdot 2^{-m} + \mathsf{adv}_F^{\mathtt{pf-prf}}(\mathcal{B}))},$$

où $\mathcal{B}$ a une complexité en temps en $\mathcal{O}(T)$ et fait au plus 2 requêtes.

Démonstration. La preuve est la conséquence directe du lemme 6.2 et du lemme 6.4. □

6.2 Étude des fonctions de hachage itérées et de CBC-MAC

Nous allons maintenant commencer à analyser des schémas pratiques d'extraction de clefs. Tout d'abord nous nous intéressons aux fonctions de hachage classiques telles que SHA-384 par exemple, d'une part car utiliser une fonction de hachage pour dériver des clefs est une méthode simple et *a priori* instinctive et d'autre part car c'est un résultat qui sera utile dans la suite. Ensuite, nous étudierons rapidement le cas de CBC-MAC et montrerons que lui aussi peut servir d'extracteur d'entropie.

6.2.1 Description du mode itéré

Ce que l'on appelle la construction en cascade est la construction qui est utilisée pour les fonctions de hachage itérées. Nous en donnons ici la description. Soit $H : \{0,1\}^n \times \{0,1\}^* \to \{0,1\}^n$ une fonction de hachage itérée et soit $h : \{0,1\}^n \times \{0,1\}^b \to \{0,1\}^n$ ce que l'on appelle communément la *fonction de compression* associée. La construction en cascade de h est la fonction $h^* : \{0,1\}^n \times \left(\{0,1\}^b\right)^* \to \{0,1\}^n$, définie par :

$$y_0 = IV,\ y_i = h(y_{i-1}, x_i) \text{ et } h^*(a, x) = y_n$$

où $x = (x_1, \ldots, x_n)$ est une chaîne de bits de taille $n \cdot b$ et $IV \in \{0,1\}^n$ est fixée. Pour construire H, on définit une manière de compléter les messages, ou *padding*, pour obtenir un message dont la taille est un multiple exact de b. En pratique cette manière de compléter les messages est une fonction de la taille $|x|$ du message x. Nous noterons dans la suite $\mathrm{pad}(|x|)$ la fonction induite par ce *padding* et nous utiliserons la notation $x_{pad} = x \| \mathrm{pad}(|x|)$. La fonction H est donc définie par $H(a, x) = h^*(a, x_{pad})$.

Soit t un entier compris entre 1 et n. Dans toute la suite, pour toute fonction F à valeurs dans $\{0,1\}^n$, on notera par $\overline{F}$ la fonction F dont les $n-t$ bits de poids faibles ont été tronqués, c'est-à-dire si $\mathsf{msb}_t(\cdot)$ désigne les t bits de poids fort, alors pour tout x dans l'ensemble de définition de

F, $\overline{F}(x) = \mathsf{msb}_t(F(x))$. Nous utiliserons cette notation essentiellement pour $\overline{h^*}$ et $\overline{H}$ (le lecteur pourra penser à SHA-384 qui est une fonction de hachage itérée pour laquelle $t = 384$ et $n = 512$).

6.2.2 Analyse

Dans cette partie, nous montrons que le mode cascade tronqué est un bon extracteur d'entropie calculatoire, dans la mesure où la fonction de compression est une PRF.

Lemme 6.5. *Soit $\mathcal{A}$ un adversaire pf-cRE contre $\overline{h^*}$ qui a un temps de calcul d'au plus T, produit une distribution de min-entropie au moins m et des messages de taille au plus ℓ blocs. Alors, il existe un adversaire PRF $\mathcal{A}'$ qui a un temps de calcul d'au plus $\mathcal{O}(T)$ et génère des messages au plus de taille ℓ blocs tel que :*

$$\mathsf{adv}^{\mathtt{pf\text{-}cre}}_{\overline{h^*}}(\mathcal{A}) \leq \sqrt{2^t \cdot (3 \cdot 2^{-m} + 2\ell \cdot \mathsf{adv}^{\mathtt{prf}}_h(\mathcal{A}'))}.$$

Ce lemme est une conséquence directe du corollaire 6.2 et du lemme suivant tiré de [BCK96a].

Lemme 6.6 (BKC)**.** *Si D est un adversaire PRF sans préfixe contre h^* qui fait au plus q requêtes, chacune constituée d'au plus ℓ blocs, alors il existe un adversaire $\mathcal{A}$ contre h tel que*

$$\mathsf{adv}^{\mathtt{pf\text{-}prf}}_{h^*}(D) \leq q\ell \cdot \mathsf{adv}^{\mathtt{prf}}_h(\mathcal{A})$$

et $\mathcal{A}$ fait au plus q requêtes et a approximativement la même complexité temporelle que D.

Comme cela a été expliqué dans la sous-section 6.1.4, l'avantage du meilleur attaquant PRF contre h est supérieur à $1/2^n$, où n désigne la taille de la clef. Or la construction cascade impose que la taille de la clef soit égale à la sortie de h. Si on ne tronquait pas la sortie de la fonction cascade, on aurait donc $t = n$ et comme $2^n \cdot \mathsf{adv}^{\mathtt{prf}}_h(\mathcal{A}) > 1$ pour le meilleur adversaire $\mathcal{A}$, cette borne ne présente aucun intérêt. Nous ne sommes donc pas en mesure d'analyser le cas où $n = t$ et sommes contraints à tronquer la sortie de la construction cascade pour se restreindre au cas où t est suffisament petit.

6.2.3 Le cas du mode CBC

Le mode CBC est un des primitives les plus utilisées pour assurer l'intégrité des échanges. Nous allons montrer qu'il peut aussi être utilisé pour faire de l'extraction de clefs pour peu que l'on en tronque la sortie. Le mode CBC a déjà été présenté au chapitre 3, dans la sous-section 4.3.1, mais nous rappelons en rappelons tout de même la description.

Soient E un algorithme de chiffrement par bloc de $\{0,1\}^n$ et K une clef pour E. Soit $M = (m_1, \ldots, m_\ell)$ un message constitué de ℓ blocs de n bits. Le CBC de M est $CBC_{E(K,\cdot)}(M) = T_{\ell+1}$ où $T_1 = 0^n$ et $T_{i+1} = E(K, m_i \oplus T_i)$ pour tout $1 \leq i \leq \ell$.

Nous rappelons maintenant un résultat prouvé dans [BPR05] qui majore l'avantage d'un attaquant contre un adversaire pf-PRF contre CBC (on rappelle que le pf dans pf-PRF signifie que les requêtes générées par l'adversaire forment un ensemble sans préfixe). Ce résultat est valable lorsque la permutation utilisée pour calculer le CBC est choisie de façon parfaitement aléatoire parmi toutes les permutations sur n bits.

Théorème 6.1 (BPR)**.** *Soit $\mathcal{A}$ un adversaire PRF sans préfixe contre CBC avec des blocs de n bits, $\mathcal{A}$ est autorisé à faire au plus $q \geq 2$ requêtes d'au plus ℓ blocs et a un temps de calcul d'au plus T. On a alors :*

$$\mathsf{adv}^{\mathtt{pf\text{-}prf}}_{CBC}(\mathcal{A}) \leq \frac{\ell q^2}{2^n}\left(12 + \frac{64\ell^3}{2^n}\right).$$

De ce théorème et du corollaire 6.2, on déduit facilement le résultat suivant :

Lemme 6.7. *Soit E un algorithme de chiffrement par bloc de $\{0,1\}^n$. Soit $\mathcal{A}$ un adversaire pf-cRE contre $\overline{CBC}$ utilisé avec E, adversaire qui a un temps de calcul d'au plus T, produit une distribution de min-entropie au moins m et des messages de taille au plus ℓ blocs. Alors, il existe un adversaire PRP $\mathcal{A}'$ qui a un temps de calcul d'au plus $\mathcal{O}(T+2\ell)$ et génère des messages au plus de taille ℓ blocs tel que :*

$$\mathsf{adv}^{\mathtt{pf\text{-}cre}}_{\overline{CBC}}(\mathcal{A}) \leq \sqrt{2^t \cdot \left(3 \cdot 2^{-m} + 2\ell \cdot \mathsf{adv}^{\mathtt{prp}}_E(\mathcal{A}') + \frac{4\ell}{2^n}\left(12 + \frac{64\ell^3}{2^n}\right)\right)}.$$

6.3 Étude de HMAC

Nous allons maintenant nous intéresser à deux constructions basées sur Hmac. La première est la même que celle analysée dans [DGH+04a] et est exactement la construction utilisée dans ike et donc dans IPsec. Ensuite, nous nous concentrons sur une constrution motivée par l'extraction de clef utilisée dans le protocole tls. Nous avons modifié la fonction d'extraction de clefs de tls pour obtenir une construction très similaire mais qui s'appuie uniquement sur Hmac, donc sur une fonction très répandue. L'analyse gagne donc en généralité. Par ailleurs, nous verrons dans la section suivante que les preuves établies ci-après s'adaptent très facilement au cas de tls.

6.3.1 Description de Nmac

Commençons par introduire les fonctions Nmac et Hmac, briques de bases de la fonction d'extraction de clefs.

Nmac est une famille de fonctions de hachage définie sur $\{0,1\}^n \times \{0,1\}^n \times \{0,1\}^*$ et à valeurs dans $\{0,1\}^c$. Elle est construite à partir d'une fonction de hachage itérée (éventuellement tronquée) Hash, définie sur $\{0,1\}^n \times \{0,1\}^*$ et à valeurs dans $\{0,1\}^c$. Si $(k_1,k_2) \in (\{0,1\}^n)^2$ est un couple de clefs et si $x \in \{0,1\}^*$ appartient à l'ensemble de définition de Hash alors la définition de Nmac est :

$$\mathsf{Nmac}^{\mathsf{Hash}}(k_1,k_2,x) \quad = \quad \mathsf{Hash}\left(k_2, \mathsf{Hash}(k_1,x)\right). \tag{6.4}$$

Si la famille de fonctions de hachage itérée Hash n'est pas tronquée, on a alors $\mathsf{Hash} = H$ et $c = n$, dans le cas contraire, on a $\mathsf{Hash} = \overline{H}$ et $c = t$. Suivant le cas, l'équation de définition (6.4) se ramène à une des deux équations suivantes :

$$\begin{aligned}
\mathsf{Nmac}^{H}(k_1,k_2,x) &= h^*\left(k_2, h^*\left(k_1, x_{pad}\right)_{pad}\right) \\
&= h\left(k_2, h^*(k_1,x_{pad})_{pad}\right), \\
\mathsf{Nmac}^{\overline{H}}(k_1,k_2,x) &= \overline{h^*}\left(k_2, \overline{h^*}\left(k_1, x_{pad}\right)_{pad}\right) \\
&= \overline{h}\left(k_2, \overline{h^*}(k_1,x_{pad})_{pad}\right).
\end{aligned}$$

Dorénavant, nous supposerons que pour tout x tel que $|x| = n$, on a $|x_{pad}| = b$ (ce qui est le cas en pratique). Cette hypothèse explique que dans la dernière équation ci-dessus, le second appel à h^* se résume à un appel à h.

La construction Nmac présente un avantage majeur, celle de conserver la propriété de PRF de la fonction de compression, contrairement à la construction cascade qui n'est qu'une pf-PRF

lorsque la fonction de compression est une PRF. En d'autres termes, quand NMAC est utilisé avec une fonction de hachage itérée classique h, si cette fonction de compression h est une bonne PRF, alors NMAC construit à partir de h est également une bonne PRF. Ce résultat a été établi par Bellare [Bel06] :

Lemme 6.8. *Soit $h\colon \{0,1\}^n \times \{0,1\}^b \to \{0,1\}^n$ une famille de fonctions. Soit $\mathcal{A}_{\mathsf{Nmac}^H}$ un adversaire PRF contre Nmac^H qui fait au plus q requêtes, chacune constituée d'au plus ℓ blocs, et qui a une complexité temporelle de T. Il existe deux adversaires PRF $\mathcal{A}_1$ et $\mathcal{A}_2$ contre h tels que l'avantage $\mathsf{adv}^{\mathtt{prf}}_{\mathsf{Nmac}^H}\left(\mathcal{A}_{\mathsf{Nmac}^H}\right)$ est majoré par :*

$$\mathsf{adv}^{\mathtt{prf}}_{h}\left(\mathcal{A}_1\right) + \binom{q}{2}\left[2\ell \cdot \mathsf{adv}^{\mathtt{prf}}_{h}\left(\mathcal{A}_2\right) + \frac{1}{2^n}\right].$$

Par ailleurs, $\mathcal{A}_1$ a une complexité temporelle d'au plus T et fait au plus q requêtes alors que $\mathcal{A}_2$ a une complexité temporelle d'au plus $\mathcal{O}(\ell \cdot T_h)$ et fait au plus 2 requêtes, où T_h désigne le temps pour une exécution de h.

Le résultat de Bellare ne s'applique pas directement à NMAC lorsque celui-ci est construit avec $\overline{H}$: la sortie de $\overline{H}$ est beaucoup plus petite que la sortie de H et à cause de cela la preuve de Bellare ne fonctionne plus et doit être adaptée. Nous avons obtenu le résultat suivant :

Lemme 6.9. *Soit $h\colon \{0,1\}^n \times \{0,1\}^b \to \{0,1\}^n$ une famille de fonctions. Soit $\mathcal{A}_{\mathsf{Nmac}^{\overline{H}}}$ un adversaire pf-PRF contre $\mathsf{Nmac}^{\overline{H}}$ qui a une complexité temporelle T et fait au plus q requêtes, chacune constituée d'au plus ℓ blocs. Il existe 2 adversaires PRF $\mathcal{A}_1$ et $\mathcal{A}_2$ contre h tels que $\mathsf{adv}^{\mathtt{pf-prf}}_{\mathsf{Nmac}^{\overline{H}}}\left(\mathcal{A}_{\mathsf{Nmac}^{\overline{H}}}\right)$ est majoré par :*

$$\mathsf{adv}^{\mathtt{prf}}_{h}\left(\mathcal{A}_1\right) + \binom{q}{2}\left[2\ell \cdot \mathsf{adv}^{\mathtt{prf}}_{h}\left(\mathcal{A}_2\right) + \frac{1}{2^t}\right].$$

où $\mathcal{A}_1$ fait q requêtes et a une complexité temporelle T, $\mathcal{A}_2$ fait 2 requêtes et a une complexité temporelle $\mathcal{O}(\ell \cdot T_h)$, où T_h désigne le temps de calcul de h.

Ce lemme peut être établi avec une preuve similaire à celle qui concerne NMAC lorsqu'il est construit avec H, preuve qui peut être trouvée dans [Bel06], à la nuance que les tests présents dans la preuve originelle sont maintenant effectués sur les t bits de poids forts de la sortie de H (et non sur la sortie entière comme dans la preuve de Bellare) et que l'adversaire est contraint à faire des requêtes sans préfixe.

6.3.2 Description de Hmac

HMAC est une famille de fonctions de hachage définie sur $\{0,1\}^* \times \{0,1\}^*$ et à valeurs dans $\{0,1\}^c$. Elle est construite à partir d'une fonction de hachage itérée (éventuellement tronquée) Hash, définie sur $\{0,1\}^n \times \{0,1\}^*$ et à valeurs dans $\{0,1\}^c$. Soient *ipad* et *opad* deux chaînes de bits de taille b et IV une chaîne de bits de taille n. Si la clef k est une chaîne de bits de taille b, alors $\mathsf{Hmac}^{\mathsf{Hash}}_{IV}(ipad, opad; k, x)$ est égal à :

$$\mathsf{Hash}\Big(IV, \big[k \oplus opad\big] \big\| \mathsf{Hash}\big(IV, [k \oplus ipad] \| x\big)\Big). \tag{6.5}$$

Les chaînes de bits *opad*, *ipad* et IV sont des constantes définies dans la norme de la famille de fonctions HMAC [BCK97]. Dans la suite nous noterons IV en indice d'HMAC et nous mettrons *ipad* et *opad* dans les parenthèses.

Si k est une chaîne de bits de taille b, alors l'équation de définition (6.5) peut être reformulée en utilisant NMAC, et alors, selon que $\mathsf{Hash} = H$ ou $\mathsf{Hash} = \overline{H}$, on obtient :

$$\mathsf{Hmac}^{H}_{IV}(ipad, opad; k, x) = \mathsf{Nmac}^{H}\left(h(IV, k \oplus ipad), h(IV, k \oplus opad), x\right),$$
$$\mathsf{Hmac}^{\overline{H}}_{IV}(ipad, opad; k, x) = \mathsf{Nmac}^{\overline{H}}\left(h(IV, k \oplus ipad), h(IV, k \oplus opad), x\right).$$

On remarquera que, en toute rigueur, les deux équations ci-dessus ne sont pas vraies, car le *padding* dans HMAC et dans NMAC n'est pas exactement le même : dans HMAC un bloc de clef est concaténé au message, ce qui change la taille de l'entrée de la fonction de hachage et donc change le *padding* qui lui est associé. Cependant, dans un soucis de simplification des notations, nous omettons cette particularité puisqu'elle n'altère pas les résultats.

Si k n'est pas une chaîne de bits de taille b, alors elle subit un traitement préalable afin de la transformer en une chaîne de bits de taille b, chaîne de bits que l'on utilisera alors comme définie dans l'équation (6.5). Ce traitement préalable dépend de la taille k par rapport à b :

- si $|k| \leq b$ alors on se contente d'ajouter à k suffisamment de '0' pour obtenir une chaîne de taille b. On ressort donc la chaîne de bits $k \| 0^{b-|k|}$;
- si $|k| \geq b$, alors la norme de HMAC [BCK97] impose que dans un premier temps l'on hache k en utilisant Hash afin d'obtenir une chaîne de bits de taille c. Puisqu'en pratique $c \leq b$, on est donc ramené au cas précédent, c'est-à-dire que dans un second temps on ajoute $b - c$ '0' pour obtenir une chaîne de bits de taille b. On ressort donc la chaîne de bits $\mathsf{Hash}(k) \| 0^{b-c}$.

Bellare [Bel06] a prouvé que HMAC est une fonction pseudo-aléatoire dans le cas où la fonction de compression est une fonction pseudo-aléatoire et où elle résiste aux attaques à clefs liées lorsque la clef est mise en entrée. Avant de présenter le résultat, nous introduisons la notion d'attaque à clefs liées.

À partir de la famille de fonctions $h\colon \{0,1\}^n \times \{0,1\}^b \to \{0,1\}^n$ nous définissons une famille de fonctions $\widehat{h}\colon \{0,1\}^b \times \{0,1\}^n \to \{0,1\}^n$ définie par $\widehat{h}(x, y) = h(y, x)$.

Une attaque par clefs liées sur la famille de fonctions $\widehat{h}\colon \{0,1\}^b \times \{0,1\}^n \to \{0,1\}^n$ est paramétrée par un ensemble $\Phi \sqsubset \mathcal{F}_{(b,b)}$ de fonctions de dérivation de clefs (où $\mathcal{F}_{(b,b)}$ désigne l'ensemble des fonctions de $\{0,1\}^b$ dans $\{0,1\}^b$). Dans le jeu des attaques à clefs liées (ou RKA pour *related-key attacks*) un challenger utilise un bit aléatoire c et une clef aléatoire K. Si $c = 1$, le challenger choisit une fonction G au hasard dans l'ensemble des fonctions de $\{0,1\}^b \times \{0,1\}^n$ dans $\{0,1\}^n$ et utilise $G(K, \cdot)$. Si $c = 0$, il utilise $\widehat{h}(K, \cdot)$. Le but de l'adversaire RKA est de deviner la valeur de c. Il peut faire des requêtes au challenger de la forme $(\phi, x) \in \Phi \times \{0,1\}^n$ et le challenger renvoie $G(\phi(K), x)$ si $b = c$, $\widehat{h}(\phi(K), x)$ sinon. L'avantage RKA de l'adversaire est défini par :

$$\mathsf{adv}^{\mathtt{rka}}_{\widehat{h},\Phi}(\mathcal{A}) = \left|\Pr\left[\mathcal{A}^{\widehat{h}} \Rightarrow 1\right] - \Pr\left[\mathcal{A}^{G} \Rightarrow 1\right]\right|.$$

Pour toutes les chaînes de bits $str \in \{0,1\}^b$ soit $\Delta_{str}\colon \{0,1\}^b \to \{0,1\}^b$ la fonction définie par $\Delta_{str}(K) = K \oplus str$.

Nous pouvons maintenant donner le théorème établi par Bellare :

Théorème 6.2. *Soit h une famille de fonctions de $\{0,1\}^n \times \{0,1\}^b$ dans $\{0,1\}^n$. Soit ipad et opad deux chaînes de b bits et soit $\Phi = \{\Delta_{ipad}, \Delta_{opad}\}$. Si $\mathcal{A}$ est un adversaire PRF contre Hmac^H qui a une complexité temporelle d'au plus T, génère des requêtes constituées d'au plus ℓ blocs, alors il existe une adversaire RKA $\mathcal{A}_1$ contre $\widehat{h}$ et deux adversaires $\mathcal{A}_2$ et $\mathcal{A}_3$ tels que $\mathsf{adv}^{\mathtt{cRE}}_{\mathsf{Hmac}^H}(\mathcal{A})$ est majoré par :*

$$\mathsf{adv}^{\mathtt{rka}}_{\widehat{h}}(\mathcal{A}_1) + \mathsf{adv}^{\mathtt{prf}}_{h}(\mathcal{A}_2) + (q-1)(q\ell - q/2) \cdot \mathsf{adv}^{\mathtt{prf}}_{h}(\mathcal{A}_3) + q(q-1)/2^{b+1}$$

où $\mathcal{A}_3$ fait au plus 2 requêtes et a une complexité en temps de $\mathcal{O}(\ell \cdot T_h)$, $\mathcal{A}_2$ fait q requêtes et a une complexité en temps T et $\mathcal{A}_1$ fait au plus 2 requêtes et a une complexité temporelle de T. La notation T_h désigne le temps de calcul de h.

6.3.3 Analyse de la première construction

Dans cette sous-section, nous étudions la construction de dérivation de clefs utilisée dans IKE. Cette construction est également mise en avant par Krawczyk [Kra] qui souhaite la normaliser. L'étude de cette construction présente donc un double intérêt. Elle a déjà fait l'objet d'une analyse dans [DGH+04a] et nous allons donner un nouvel éclairage sur cette construction la considérant d'un point de vue calculatoire.

Plus précisément la construction est la suivante. Soit pmk une chaîne de bits de taille s que dans la suite nous appelerons secret initial et soit mk le secret maître généré par cette construction. La chaîne de bits $rand$ est choisie de façon uniforme dans $\{0,1\}^b$ et mk est calculé de la façon suivante :

$$mk = \overline{\mathsf{Hmac}^{\mathsf{Hash}}}_{IV}(ipad, opad; rand, pmk).$$

Cela signifie que pour générer mk, après le calcul de HMAC, on tronque le résultat pour ne garder que les t bits de poids faible. Nous étudions uniquement le cas où la fonction de hachage utilisé n'est pas une fonction de hachage tronquée. L'autre cas peut être déduit des résultats présentés dans la sous-section suivante. Le théorème que nous présentons est une conséquence directe du théorème 6.2 et du corollaire 6.1.

Théorème 6.3. *Soit h une famille de fonctions de $\{0,1\}^n \times \{0,1\}^b$ dans $\{0,1\}^n$. Soit ipad et opad deux chaînes de b bits et soit $\Phi = \{\Delta_{ipad}, \Delta_{opad}\}$. Si $\mathcal{A}$ est un adversaire cRE contre la construction précédente, adversaire qui a une complexité temporelle d'au plus T, génère une distribution de probabilité sur $\{0,1\}^{b\ell}$ avec $\ell \geq 2$ de min-entropie plus grande que m, alors il existe une adversaire RKA $\mathcal{A}_1$ contre $\widehat{h}$ et deux adversaires prf $\mathcal{A}_2$ et $\mathcal{A}_3$ contre h tels que $\mathsf{adv}^{\mathtt{prf}}_{\mathit{Hmac}^H}(\mathcal{A})$ est majoré par :*

$$\sqrt{2^t\left(2^{-m} + \mathsf{adv}^{\mathtt{rka}}_{\widehat{h}}(\mathcal{A}_1) + \mathsf{adv}^{\mathtt{prf}}_{h}(\mathcal{A}_2) + (2\ell - 1)\cdot \mathsf{adv}^{\mathtt{prf}}_{h}(\mathcal{A}_3) + \frac{1}{2^b}\right)}$$

où $\mathcal{A}_3$ fait au plus 2 requêtes et a une complexité en temps de $\mathcal{O}(\ell \cdot T_h)$, $\mathcal{A}_2$ fait 2 requêtes et a une complexité en temps T et $\mathcal{A}_1$ fait au plus 2 requêtes et a une complexité temporelle de T. La notation T_h désigne le temps de calcul de h.

Ce théorème permer de justifier l'utilisation de cette construction avec SHA-512 pour extraire une clef de 256 bits. C'est exactement ce qui est conseillé par Krawczyk [Kra], qui propose ensuite d'utiliser les 256 bits créés comme clef lors de la phase de génération de clef. Pour cette phase, il propose d'utiliser une fonction pseudo-aléatoire dont la clef fait 256 bits (par exemple une fonction basée sur la construction précédente, mais avec SHA-256) et ainsi de produire autant de bits de clefs que nécessaires. Krawczyk explique qu'il vaut mieux utiliser SHA-512 et en tronquer la sortie, plutôt qu'utiliser directement SHA-256. Notre résultat permet de prouver la construction de Krawczyk dans le cas de SHA-512 et ne prouve rien dans le cas de SHA-256, nous ne pouvons donc que partager son point de vue, et, comme lui, encourager, par précaution, à utiliser SHA-512 puis à tronquer la sortie.

6.3.4 Analyse de la seconde construction

Dans cette sous-section, nous étudions une construction de dérivation de clefs particulière. Soit *pmk* une chaîne de bits de taille s que dans la suite nous appelerons secret initial, soit *label* une chaîne de bits quelconque, éventuellement générée par l'adversaire, et soit mk le secret maître généré par cette construction. Les trois chaînes de bits IV, *ipad* et *opad* sont considérées comme choisies au hasard et mk est calculé de la façon suivante :

$$mk = \mathsf{Hmac}_{IV}^{\mathsf{Hash}}(ipad, opad; pmk, label).$$

Comme cela a déjà été précisé, en pratique, IV, *ipad* et *opad* ne sont pas choisis au hasard à chaque utilisation mais imposées dans la norme d'HMAC [BCK97] pour *ipad* et *opad* et dans celui de la fonction de hachage concernée pour IV (comme celui de SHA-1 [SHA95] par exemple). Cependant, dans la suite nous supposerons qu'elles peuvent varier et qu'elles sont choisies au hasard avant chaque utilisation de HMAC. En pratique, ces variables ont été choisies au hasard une fois pour toute quand la norme a été spécifié. La validité et les conséquences pratiques de cette hypothèse seront discutées en détail dans la partie 6.4.4.

Nous montrons maintenant que cette construction est un bon extracteur d'entropie calculatoire, c'est-à-dire que le quadruplet $(IV, ipad, opad; mk)$ est indistinguable d'un quadruplet parfaitement aléatoire. Comme la définition de HMAC dépend de la taille de *pmk*, notre preuve est aussi dépendente de la taille de *pmk* : notre méthode de preuve n'est pas exactement la même suivant la taille de *pmk* et les résultats diffèrent également. Pour cette raison nous présentons les résultats de façon séparée.

Quand la clef est plus courte que le bloc

L'étude de ce cas est motivée par l'utilisation pratique des échanges de clefs Diffie-Hellman sur courbe elliptique. Dans ce cas le secret initial généré est habituellement long de 512 bits, c'est-à-dire plus petit que la longueur du bloc. Nous montrons directement que HMAC est un extracteur d'entropie, qu'il soit instancié avec H ou avec $\overline{H}$. Les bornes obtenues dans les deux cas sont identiques et les deux preuves sont identiques également, à une nuance près : dans le premier cas la preuve repose sur le résultat de Bellare qui établit que NMAC est une PRF lorsqu'il est instancié avec H, dans le second cas la preuve repose sur le lemme 6.9 qui établit que NMAC est une PRF lorsqu'il est instancié avec $\overline{H}$.

HMAC construit à partir de H. Tout d'abord nous montrons que, pour une clef plus petite que la taille du bloc, HMAC est un bon extracteur d'entropie calculatoire quand il est construit à partir de H. Nous insistons de nouveau sur le fait que dans le théorème suivant, *ipad* et *opad* sont choisies au hasard. Au contraire, la chaîne de bits IV n'a pas besoin d'être choisie au hasard ici, nous énoncerons donc le résultat et ferons la preuve comme si elle était fixée. Nous supposons également que $h(k, \cdot)$ est une PRF quand k est la clef et que $h(IV, \cdot \oplus k)$ est une PRF quand k est la clef.

Théorème 6.4. *Soit IV une chaîne de bits de taille n et soit h une famille de fonctions de $\{0,1\}^n \times \{0,1\}^b$ dans $\{0,1\}^n$, où la clef est la première entrée de la fonction, sur n bits. Soit h' la famille de fonctions définie par $h'_{IV}(pad, \cdot) = h(IV, \cdot \oplus pad)$ où la clef est pad, chaîne de b bits. Si $\mathcal{A}$ est un adversaire cRE dont la complexité temporelle est égale à T et qui génère des labels constitués d'au plus ℓ blocs et une distribution sur $\{0,1\}^b$ de min-entropie au moins m, alors il existe un adversaire PRF $\mathcal{A}_1$ contre h' et deux adversaires PRF $\mathcal{A}_2$ et $\mathcal{A}_3$ contre h tels*

que $\mathsf{adv}^{\mathtt{cre}}_{\mathit{Hmac}^H}(\mathcal{A})$ *est majoré par :*

$$\frac{\sqrt{2^{2n}\left(2^{-m}+2\cdot\mathsf{adv}^{\mathtt{prf}}_{h'}(\mathcal{A}_1)\right)}}{2}+\frac{1}{2^n}+\mathsf{adv}^{\mathtt{prf}}_{h}(\mathcal{A}_2)+2\ell\cdot\mathsf{adv}^{\mathtt{prf}}_{h}(\mathcal{A}_3)$$

où $\mathcal{A}_1$ *fait deux requêtes et a une complexité temporelle égale à* $T+2T_h$, $\mathcal{A}_2$ *fait 1 requête et a une complexité temporelle égale à* T *et* $\mathcal{A}_3$ *fait au plus 2 requêtes et a une complexité temporelle* $\mathcal{O}(\ell\cdot T_h)$ *et* T_h *désigne le temps d'une évaluation de* h*.*

On fait remarquer que si $\mathsf{adv}^{\mathtt{prf}}_{h'}(\mathcal{A}_1)$ était plus grand que 2^{-2n}, nous aurions $2^{2n}\cdot\mathsf{adv}^{\mathtt{prf}}_{h'}(\mathcal{A}_1)\geq 1$ et la borne supérieure ne nous serait d'aucune utilité. La clef de la fonction h' égale à $h(IV,\cdot\oplus pad)$ est pad et fait donc b bits. D'après ce qui a été expliqué dans la sous-section 6.1.4, le meilleur attaquant $\mathcal{A}_1$ contre h' a un avantage supérieur à $1/2^b$, la borne du théorème n'est donc utile que si $b\geq 2n$. C'est le cas pour SHA-1, pour lequel $b=512$ et $n=160$, mais pas pour SHA-256, ni SHA-512, pour lesquels $b=2n$ exactement.

Par ailleurs, à la place de considérer que la clef de h' est pad, on aurait préféré considérer que cette clef est IV, car cela aurait conduit à faire l'hypothèse que h est une PRF, ce qui est plus classique. Cependant comme IV ne fait que n bits, on sait que le meilleur adversaire PRF contre h a un avantage meilleur que 2^{-n} et donc la borne prouvée par le théorème n'est d'aucune utilité. Supposer que h est une PRF n'est donc pas suffisant, alors que, puisque $ipad$ et $opad$ sont grands, le niveau de sécurité de h' peut être suffisant.

L'idée principale de la preuve est de montrer que deux chaînes de bits $k_1=h(IV,ipad\oplus k)$ et $k_2=h(IV,opad\oplus k)$ sont pseudo-aléatoires (et donc notamment indépendantes) puis de les utiliser comme clefs pour NMAC qui est une PRF et dont la sortie est indistinguable d'une chaîne de bits parfaitement aléatoire. Nous faisons remarquer que, contrairement à [BCK96a] dans lequel il est *supposé* que k_1 et k_2 sont calculatoirement indépendantes, ici nous le prouvons en utilisant la famille de fonctions de hachage suivante :

$$(F(\cdot))_{(ipad,opad)}=(h(IV,\cdot\oplus ipad)\|h(IV,\cdot\oplus opad))_{(ipad,opad)}$$

qui est une PRF et donc qui est cRE.

Démonstration. Avant de considérer la preuve du théorème, nous ramenons la sécurité en PRF de la famille de fonctions de hachage

$$(F(\cdot))_{(ipad,opad)}=(h(IV,\cdot\oplus ipad)\|h(IV,\cdot\oplus opad))_{(ipad,opad)}$$

à celle de h' en montrant que, pour tout adversaire PRF $\mathcal{A}_F$ contre F de complexité T et qui fait 2 requêtes, il existe un adversaire PRF $\mathcal{A}_{h'}$ contre h' qui fait 2 requêtes, a une complexité temporelle de $T+2T_h$ et a un avantage $\mathsf{adv}^{\mathtt{prf}}_{h'}(\mathcal{A}_{h'})$ tel que $\mathsf{adv}^{\mathtt{prf}}_{F}(\mathcal{A}_F)=2\mathsf{adv}^{\mathtt{prf}}_{h'}(\mathcal{A}_{h'})$. Voici comment on construit $\mathcal{A}_{h'}$. Au début du jeu, il choisit un bit b et une clef pad parfaitement au hasard puis lance $\mathcal{A}_F$. À chaque requête x_i de $\mathcal{A}_F$, pour $1\leq i\leq 2$, $\mathcal{A}_{h'}$ fait également suivre x_i au challenger qui répond par y_i. Si $b=0$ alors $\mathcal{A}_{h'}$ choisit au hasard y'_i et renvoie $y'_i\|y_i$ à $\mathcal{A}_F$. Si $b=1$ alors $A_{h'}$ calcule $y'_i=F(pad,x_i)$ et envoie $y_i\|y'_i$ à $\mathcal{A}_F$ (on remarquera que y'_i et y_i ne sont pas concaténés dans le même ordre selon les cas). À la fin, $\mathcal{A}_F$ envoie un bit b' que $\mathcal{A}_{h'}$ fait suivre au challenger.

Dans les calculs qui suivent, on notera par $\mathcal{F}=\mathcal{F}_{b,n}$ l'ensemble de fonction de $\{0,1\}^b$ dans $\{0,1\}^n$. L'avantage de $\mathcal{A}_{h'}$ $\mathsf{adv}^{\mathtt{prf}}_{h'}(\mathcal{A}_{h'})$ est égal à :

$$
\begin{aligned}
&= \left|\Pr[pad' \xleftarrow{\$} \{0,1\}^b : \mathcal{A}_{h'}^{h'(pad',\cdot)} \Rightarrow 1] - \Pr[f \xleftarrow{\$} \mathcal{F} : \mathcal{A}_{h'}^{f} \Rightarrow 1]\right| \\
&= \left|\Pr[b=0]\left(\Pr\left[\begin{array}{c} pad \xleftarrow{\$} \{0,1\}^b \\ f' \xleftarrow{\$} \mathcal{F} \end{array} : \mathcal{A}_F^{h'(pad,\cdot)\|f'(\cdot)} \Rightarrow 1\right] - \Pr\left[\begin{array}{c} f \xleftarrow{\$} \mathcal{F} \\ f' \xleftarrow{\$} \mathcal{F} \end{array} : \mathcal{A}_F^{f\|f'} \Rightarrow 1\right]\right)\right. \\
&+ \Pr[b=1]\left(\Pr\left[\begin{array}{c} pad \xleftarrow{\$} \{0,1\}^b \\ pad' \xleftarrow{\$} \{0,1\}^b \end{array} : \mathcal{A}_F^{h'(pad,\cdot)\|h'(pad',\cdot)} \Rightarrow 1\right]\right. \\
&\left.\left. - \Pr\left[\begin{array}{c} pad' \xleftarrow{\$} \{0,1\}^b \\ f \xleftarrow{\$} \mathcal{F} \end{array} : \mathcal{A}_F^{h'(pad',\cdot)\|f(\cdot)} \Rightarrow 1\right]\right)\right| \\
&= \frac{1}{2}\left|\Pr\left[\begin{array}{c} pad \xleftarrow{\$} \{0,1\}^b \\ pad' \xleftarrow{\$} \{0,1\}^b \end{array} : \mathcal{A}_F^{h'(pad,\cdot)\|h'(pad',\cdot)} \Rightarrow 1\right] - \Pr\left[\begin{array}{c} f \xleftarrow{\$} \mathcal{F} \\ f' \xleftarrow{\$} \mathcal{F} \end{array} : \mathcal{A}_F^{f\|f'} \Rightarrow 1\right]\right| \\
&= \frac{1}{2}\left|\Pr\left[\begin{array}{c} pad \xleftarrow{\$} \{0,1\}^b \\ pad' \xleftarrow{\$} \{0,1\}^b \end{array} : \mathcal{A}_F^{F(pad,pad',\cdot)} \Rightarrow 1\right] - \Pr\left[\begin{array}{c} f \xleftarrow{\$} \mathcal{F} \\ f' \xleftarrow{\$} \mathcal{F} \end{array} : \mathcal{A}_F^{f\|f'} \Rightarrow 1\right]\right| \\
&= \frac{\mathsf{adv}_F^{\mathtt{prf}}(\mathcal{A}_F)}{2}.
\end{aligned}
$$

La dernière égalité est vraie car faire face à deux fonctions aléatoires concaténées est équivalent à faire face à une grande fonction aléatoire.

Pour démontrer le théorème, considérons maintenant la suite de jeux suivante :

Game 0 : Ce jeu correspond à l'attaque lorsqu'on exécute la vraie extraction.

1. $\mathcal{A}$ envoie $(\mathcal{D}, label)$
2. $pmk \xleftarrow{\mathcal{D}} \{0,1\}^s$, $opad \xleftarrow{\$} \{0,1\}^b$, $ipad \xleftarrow{\$} \{0,1\}^b$
3. $(k_1, k_2) = (h(IV, pmk \oplus ipad), h(IV, pmk \oplus opad))$
4. $k = \mathsf{Nmac}^H(k_1, k_2, label)$, envoyer $(IV, ipad, k)$ à $\mathcal{A}$
5. $\mathcal{A}$ envoie b'.

Game 1 : Dans ce jeu, nous choisissons k_1 et k_2 parfaitement au hasard dans $\{0,1\}^n$.

Game 2 : Dans ce jeu, nous choisissons k parfaitement au hasard dans $\{0,1\}^n$.

Tout d'abord, distinguer les jeux 0 et 1 équivaut à être capable de distinguer la distribution de $(ipad, opad, F(ipad, opad; pmk))$ de la distribution de $(ipad, opad, U_{2n})$. La distance entre les jeux 0 et 1 peut donc être majorée en utilisant le corollaire 6.1 pour F combiné au petit résultat établi plus haut. Il existe donc un adversaire $\mathcal{A}_1$ comme décrit dans l'énoncé du théorème tel que la distance entre les deux jeux est majorée par

$$\sqrt{2^{2n} \cdot (2^{-m} + 2 \cdot \mathsf{adv}_{h'}^{\mathtt{prf}}(\mathcal{A}_1))}.$$

Ensuite, distinguer les jeux 1 et 2 équivaut à être capable de distinguer Nmac d'une fonction parfaitement aléatoire. Il existe donc un adversaire PRF $\mathcal{A}'$ contre Nmac qui fait au plus 1 requête et a une complexité temporelle T telle que la distance entre les jeux 1 et 2 est majorée par $\mathsf{adv}_{\mathsf{Nmac}^H}^{\mathtt{prf}}(\mathcal{A}')$. Le lemme 6.8 prouvé par Bellare implique que ce dernier avantage est plus

petit que $\mathsf{adv}_h^{\mathtt{prf}}(\mathcal{A}_2) + 2\ell \cdot \mathsf{adv}_h^{\mathtt{prf}}(\mathcal{A}_3) + 1/2^n$ pour un certain $\mathcal{A}_2$ et un certain $\mathcal{A}_3$ comme décrits dans l'énoncé du théorème. □

HMAC construit à partir de $\overline{H}$. Dans une seconde partie, nous montrons que, toujours pour une clef plus petite que la taille du bloc, HMAC construit avec $\overline{H}$ est un bon extracteur d'entropie calculatoire.

Théorème 6.5. *Soit IV une chaîne de n bits et soit h une famille de fonctions de $\{0,1\}^n \times \{0,1\}^b$ dans $\{0,1\}^n$, où la clef est la première entrée sur n bits. Soit h' la fonction de hachage définie par $h'_{IV}(pad, \cdot) = h(IV, \cdot \oplus pad)$ où la clef est pad. Si $\mathcal{A}$ est un adversaire cRE qui a une complexité temporelle d'au plus T, génère des labels d'au plus ℓ blocs et une distribution sur $\{0,1\}^b$, avec une min-entropie supérieure à m, alors il existe un adversaire PRF $\mathcal{A}_1$ contre h' et 2 adversaires PRF $\mathcal{A}_2$ et $\mathcal{A}_3$ contre h tels que $\mathsf{adv}^{\mathtt{cre}}_{\mathsf{Hmac}^H}(\mathcal{A})$ est majoré par :*

$$\frac{\sqrt{2^{2n}\left(2^{-m} + 2 \cdot \mathsf{adv}^{\mathtt{prf}}_{h'}(\mathcal{A}_1)\right)}}{2} + \frac{1}{2^t} + \mathsf{adv}^{\mathtt{prf}}_h(\mathcal{A}_2) + 2\ell \cdot \mathsf{adv}^{\mathtt{prf}}_h(\mathcal{A}_3)$$

où $\mathcal{A}_1$ fait 2 requêtes et a une complexité temporelle $T + 2T_h$, $\mathcal{A}_2$ fait 1 requête et a une complexité temporelle T et $\mathcal{A}_3$ fait au plus 2 requêtes et a une complexité temporelle $\mathcal{O}(\ell \cdot T_h)$, où T_h désigne le temps de calcul de h.

La preuve de ce théorème est relativement similaire à celle du théorème 6.4, à la différence que l'on n'utilise plus le lemme 6.8 de Bellare mais cette fois le lemme 6.9.

Démonstration. Considérons la suite de jeux suivante :

Game 0 : Ce jeu correspond à l'attaque lorsque l'on exécute la vraie extraction.

1. $\mathcal{A}$ envoie $(\mathcal{D}, label)$
2. $pmk \overset{\mathcal{D}}{\leftarrow} \{0,1\}^s$, $IV \overset{\$}{\leftarrow} \{0,1\}^n$, $ipad \overset{\$}{\leftarrow} \{0,1\}^b$
3. $(k_1, k_2) = (h(pmk \oplus ipad, IV), h(pmk \oplus opad, IV))$
4. $k = \mathsf{Nmac}^{\overline{H}}(k_1, k_2, label)$, envoyer $(IV, ipad, k)$ à $\mathcal{A}$
5. $\mathcal{A}$ envoie b'

Game 1 : Dans ce jeu, nous choisissons k_1 et k_2 parfaitement au hasard dans $\{0,1\}^n$.

Game 2 : Dans ce jeu, nous choisissons k parfaitement au hasard dans $\{0,1\}^n$.

Tout d'abord, de même que dans la preuve du thèorème précédent, la distance entre les jeux 0 et 1 peut être majorée en utilisant le corollaire 6.1 : cette distance est donc majorée par

$$\sqrt{2^{2n} \cdot (2^{-m} + 2 \cdot \mathsf{adv}^{\mathtt{prf}}_{h'}(\mathcal{A}_1))}.$$

Ensuite, il existe un adversaire PRF $\mathcal{A}'$ contre NMAC (quand il est instancié avec $\overline{H}$) qui fait au plus 1 requête et a une complexité temporelle T telle que la distance entre les jeux 1 et 2 est majorée par $\mathsf{adv}^{\mathtt{prf}}_{\mathsf{Nmac}^{\overline{H}}}(\mathcal{A}')$. Puisque $\mathcal{A}'$ fait uniquement une requête, il est évidemment sans préfixe et donc son avantage est égal à $\mathsf{adv}^{\mathtt{pf-prf}}_{\mathsf{Nmac}^{\overline{H}}}(\mathcal{A}')$. Ceci est majoré par $\mathsf{adv}^{\mathtt{prf}}_h(\mathcal{A}_3) + 2\ell \cdot \mathsf{adv}^{\mathtt{prf}}_h(\mathcal{A}_4) + \frac{1}{2^t}$, comme nous l'avons prouvé dans le lemme 6.9. □

Quand la clef est plus longue que le bloc

Comme expliqué précédemment, si la clef est plus grande que la taille du bloc, alors elle est d'abord hachée, puis on lui concatène des '0' afin d'obtenir une chaîne de bits de taille b. Ce cas est rarement étudié lors des analyses de sécurité de HMAC et nécessite une étude spécifique pour appréhender l'impact du hachage de la clef. Cependant cette étude a un intérêt pratique puisqu'un échange de clefs Diffie-Hellman sur $\mathbb{Z}_p$ (où p est un nombre premier) génère en pratique un secret initial d'au moins 1024 bits, ce qui est plus grand que les 512 bits de clefs de HMAC-SHA-1. Dans cette partie, nous nous concentrerons sur HMAC uniquement quand il est utilisé avec une fonction de hachage tronquée $\overline{H}$. Nous donnerons d'abord le résultat de sécurité puis les lemmes intermédiaires utiles à la preuve.

Hacher le secret initial a deux conséquences principales sur notre preuve. Tout d'abord, la sortie du hachage est une chaîne de bits de taille t et on doit prouver que cette chaîne de bits a toujours beaucoup d'entropie. Si ce n'était pas le cas, alors, comme l'ajout de '0' n'augmente pas l'entropie, la clef de HMAC aurait peu d'entropie et une recherche exhaustive de cette clef permettrait de deviner les quelques valeurs possibles du secret-maître. Plus précisément, nous montrons que, lorsque la fonction de hachage utilisée est une fonction de hachage itérée tronquée, la sortie du hachage initial est une chaîne de bits calculatoirement indistinguable d'une chaîne de bits parfaitement aléatoire.

L'autre conséquence du hachage est que la clef de HMAC n'est plus uniformément distribuée puisque ses $b-t$ bits de poids faible sont égaux à '0'. Nous devons donc montrer que même dans ses circonstances HMAC est toujours une bonne fonction pseudo-aléatoire, ce qui garantie que la sortie de HMAC est calculatoirement indistinguable d'une chaîne de bits parfaitement aléatoire. Dans ce but, nous allons considérer les attaques à clefs liées contre h, alors que l'entrée et la clef sont inversées.

À partir de la famille de fonctions $h\colon \{0,1\}^n \times \{0,1\}^b \to \{0,1\}^n$ nous définissons une famille de fonctions $\widehat{h}\colon \{0,1\}^t \times \{0,1\}^n \to \{0,1\}^n$ définie par $\widehat{h}(x,y) = h(y, x\|0^{b-t})$.

Nous rappelons la définition des attaques à clefs liées contre $\widehat{h}$. Une attaque par clefs liées sur une famille de fonctions $\widehat{h}\colon \{0,1\}^t \times \{0,1\}^n \to \{0,1\}^n$ est paramétrée par un ensemble $\Phi \sqsubset \mathcal{F}_{(t,t)}$ de fonctions de dérivation de clefs (où $\mathcal{F}_{(t,t)}$ désigne l'ensemble des fonctions de $\{0,1\}^t$ dans $\{0,1\}^t$). Dans le jeu des attaques à clefs liées (ou RKA pour *related-key attacks*) un challenger utilise un bit aléatoire b et une clef aléatoire K. Si $b = 1$, le challenger choisit une fonction G au hasard dans l'ensemble des fonctions de $\{0,1\}^t \times \{0,1\}^n$ dans $\{0,1\}^n$ et utilise $G(K,\cdot)$. Si $b = 0$, il utilise $\widehat{h}(K,\cdot)$. Le but de l'adversaire RKA est de deviner la valeur de b. Il peut faire des requêtes au challenger de la forme $(\phi, x) \in \Phi \times \{0,1\}^n$ et le challenger renvoie $G(\phi(K), x)$ si $b = 1$, $\widehat{h}(\phi(K), x)$ sinon. L'avantage RKA de l'adversaire est défini par :

$$\mathsf{adv}^{\mathtt{rka}}_{\widehat{h},\Phi}(\mathcal{A}) = \left|\Pr\left[\mathcal{A}^{\widehat{h}} \Rightarrow 1\right] - \Pr\left[\mathcal{A}^{G} \Rightarrow 1\right]\right|.$$

Pour toutes les chaînes de bits $str \in \{0,1\}^t$ soit $\Delta_{str}\colon \{0,1\}^t \to \{0,1\}^t$ la fonction définie par $\Delta_{str}(K) = K \oplus str$.

Pour le théorème suivant, IV est choisie au hasard, mais *ipad* et *opad* n'ont pas besoin d'être choisies au hasard et nous énoncerons le résultat et la preuve comme si elles étaient fixées.

Théorème 6.6. *Soit h une famille de fonctions de $\{0,1\}^n \times \{0,1\}^b$ dans $\{0,1\}^n$. Soit ipad et opad deux chaînes de b bits et soit $\Phi = \{\Delta_{ipad}, \Delta_{opad}\}$. Si $\mathcal{A}$ est un adversaire pf-cRE qui a une complexité temporelle d'au plus T, génère des labels constitués d'au plus ℓ blocs et une distribution de probabilité sur $\{0,1\}^{sb}$ avec $s \geq 2$ de min-entropie plus grande que m, alors il*

existe une adversaire RKA $\mathcal{A}_2$ contre $\widehat{h}$ et trois adversaires $\mathcal{A}_1$, $\mathcal{A}_3$ et $\mathcal{A}_4$ tels que $\mathsf{adv}^{\mathtt{pf-cre}}_{\mathit{Hmac}^{\overline{H}}}(\mathcal{A})$ *est majoré par :*

$$\sqrt{2^t\left(3\cdot 2^{-m}+2s\cdot \mathsf{adv}^{\mathtt{prf}}_{h}(\mathcal{A}_1)\right)}+\mathsf{adv}^{\mathtt{rka}}_{\widehat{h}}(\mathcal{A}_2)$$
$$+\mathsf{adv}^{\mathtt{prf}}_{h}(\mathcal{A}_3)+2\ell\cdot\mathsf{adv}^{\mathtt{prf}}_{h}(\mathcal{A}_4)+\tfrac{1}{2^t}$$

où $\mathcal{A}_1$ et $\mathcal{A}_2$ font au plus 2 requêtes et ont une complexité en temps de T, $\mathcal{A}_3$ fait 1 requête et a une complexité en temps T et $\mathcal{A}_4$ fait au plus 2 requêtes et a une complexité temporelle de $\mathcal{O}(\ell\cdot T_h)$.

Pour prouver ce théorème, nous appliquons d'abord les résultats d'extraction sur la fonction cascade établis dans la section précédente. De cette manière nous prouvons que la sortie du hachage préalable est une chaîne de t bits qui semble parfaitement aléatoire. Comme par ailleurs $\widehat{h}$ est une PRF résistante aux attaques à clefs liées, ceci implique que $h(IV, ipad \oplus k\|0^{b-t})\|h(IV, opad \oplus k\|0^{b-t})$ semble uniformément distribuée. Par conséquent, nous utilisons NMAC avec deux clefs qui semblent parfaitement aléatoires et puisque NMAC est une PRF, sa sortie est indistinguable d'une chaîne de bits parfaitement aléatoires.

On remarquera que dans la preuve qui suit, IV est choisi au hasard au début du jeu. Par contre, on n'utilise pas le fait que *ipad* et *opad* sont choisis au hasard et ceux-ci pourraient être choisis autrement.

Démonstration. Considérons la séquence de jeux suivante.

Game 0 : Ce jeux correspond à l'attaque quand la vraie extraction de clefs est exécutée.

1. $\mathcal{A}$ envoie $(\mathcal{D}, label)$
2. $pmk \xleftarrow{\mathcal{D}} \{0,1\}^s$, $IV \xleftarrow{\$} \{0,1\}^n$, $K = \overline{H}(IV, pmk)$
3. $(k_1, k_2) = \left(\widehat{h}(K \oplus ipad, IV), \widehat{h}(K \oplus opad, IV)\right)$
4. $k = \mathsf{Nmac}^{\overline{H}}(k_1, k_2, labels)$, envoyer (IV, k) à $\mathcal{A}$
5. $\mathcal{A}$ envoie b'

Game 1 : Dans ce jeu, nous choisissons K de façon uniforme dans $\{0,1\}^t$.

Game 2 : Dans ce jeu, nous choisissons k_1 et k_2 de façon uniforme dans $\{0,1\}^n$.

Game 3 : Dans ce jeu, nous choisissons k de façon uniforme dans $\{0,1\}^n$. Cela correspond à l'attaque quand l'extraction de clefs est effectuée par une fonction aléatoire.

Puisque $\mathcal{A}$ est sans préfixe, la distance entre les jeux 0 et 1 peut être bornée en utilisant le lemme 6.5 : il existe un adversaire $\mathcal{A}_1$ comme décrit dans le théorème tel que la distance entre les deux jeux est majorée par $\sqrt{2^t\cdot(2^{-m}+2s\cdot\mathsf{adv}^{\mathtt{prf}}_{h}(\mathcal{A}_1))}$. Il existe $\mathcal{A}_2$ comme décrit dans le théorème tel que la distance entre les jeux 1 et 2 est majorée par $\mathsf{adv}^{\mathtt{rka}}_{\widehat{h},\Phi}(\mathcal{A}_2)$. Puisque $\mathcal{A}$ est sans préfixe, il existe un adversaire $\mathcal{A}'$ qui fait au plus 1 requête, constituée d'au plus ℓ blocs, et qui a une complexité temporelle de T et tel que la distance entre les jeux 2 et 3 est majorée par $\mathsf{adv}^{\mathtt{pf-prf}}_{\mathsf{Nmac}^{\overline{H}}}(\mathcal{A}')$. D'après le lemme 6.9, il existe $\mathcal{A}_3$ et $\mathcal{A}_4$ tels que cet avantage est plus petit que $\mathsf{adv}^{\mathtt{prf}}_{h}(\mathcal{A}_3)+2\ell\cdot\mathsf{adv}^{\mathtt{prf}}_{h}(\mathcal{A}_4)+\frac{1}{2^t}$ (où $\mathcal{A}_3$ et $\mathcal{A}_4$ sont tels qu'ils sont décrits dans l'énoncé du lemme). □

6.4 Application à l'analyse de TLS

Dans cette partie, nous appliquons les méthodes et les résultats des parties précédentes à la nouvelle version en cours d'élaboration de TLS v.1.2 [DR08]. Nous donnons des preuves de sécurité pour la fonction d'extraction de clefs décrite dans la norme, fonction qui est très similaire à celle présente dans les versions précédentes de TLS.

Par ailleurs, les nouvelles versions de la norme de TLS promeuvent l'utilisation d'au moins SHA-256, voire d'une fonction de hachage plus forte. Dans ce papier, nous nous concentrons sur SHA-384 et donnons des résultats de sécurité s'adressant au cas spécifique d'une fonction de hachage itérée tronquée.

6.4.1 Description de la fonction de dérivation de clefs

Dans TLS l'extraction de clefs est effectuée de la manière suivante. Tout d'abord le client et le serveur échangent deux chaînes publiquent de 256 bits, $rand_s$ et $rand_c$, avec dans chacune 224 bits parfaitement aléatoires et 32 bits fixés. Ensuite le client et le serveur échange le secret initial.

Dans l'échange de clefs RSA, le client génère une chaîne aléatoire de 368 bits, concaténée à l'encodage sur 16 bits de la dernière version du protocole supportée, et chiffre le tout en utilisant la clef publique RSA du serveur. C'est cette chaîne de 384 bits qui constitue le secret initial. C'est une valeur sur 384 bits, mais on souligne qu'il y a seulement 368 bits d'entropie, puisque les 16 bits de poids forts sont fixés.

Dans l'échange de clefs Diffie-Hellman, le client et le serveur utilisent un groupe G dans lequel l'hypothèse DDH est vraie et réalisent un échange Diffie-Hellman pour obtenir un élément commun de G. Sous l'hypothèse DDH, cet élément paraît uniformément distribué dans le groupe G. Sa représentation binaire est le secret initial. On fait remarquer que cette représentation binaire n'est pas une chaîne de bits uniformément distribuée.

Dans les deux cas, nous noterons par pmk le secret initial. Ensuite, ce que l'on appelle le secret maître, noté mk, est créé. C'est une chaîne de 384 bits exactement. Durant cette étape, l'entropie de pmk est extraite en faisant appel à une fonction appelée HPRF :

$$mk = \mathsf{Hprf}(pmk, \text{“}\texttt{master␣secret}\text{”}\|rand_c\|rand_s),$$

où “`master␣secret`” est un label.

La fonction HPRF peut être n'importe quelle fonction pour peu qu'elle ait été autorisée au cours de l'échange, cependant la norme en propose une par défaut et nous nous concentrerons dans la suite sur celle-ci. Cette fonction est très similaire à celle utilisée dans la version précédente de TLS, TLS 1.1. Cette fonction est construite à partir de la concaténation et l'itération de HMAC. Pour des raisons de simplicité, nous allons d'abord introduire une fonction intermédiaire appelée NPRF, construite à partir de concaténations de NMAC. Soit Hash une fonction de hachage, comme SHA-384 par exemple. Par défaut, NPRF est définie par

$$\mathsf{Nprf}^{\mathsf{Hash}}(k_1, k_2, x) = \mathsf{Nmac}^{\mathsf{Hash}}(k_1, k_2, W(1)\|x)\Big\| \ldots \Big\|\mathsf{Nmac}^{\mathsf{Hash}}(k_1, k_2, W(v)\|x),$$
$$(\forall\, 1 \leq i \leq v) \quad W(i) = \mathsf{Nmac}^{\mathsf{Hash}}(k_1, k_2, W(i-1)) \text{ et } W(0) = x,$$

où v désigne le nombre de fois que l'on concatène NMAC. Ce nombre dépend de la taille de la sortie de la fonction de hachage. Puisque le secret maître doit faire 384 bits, si la fonction de sortie est sur 160 bits, comme SHA-1, alors il faut concaténer 3 fois la fonction de sortie et jeter les 96 bits de poids faible excédentaires. Par contre si la fonction est SHA-384, alors $v = 1$ suffit.

Exactement de la même manière que l'on dérive HMAC de NMAC, la fonction HPRF dérive de la fonction NPRF. C'est-à-dire que l'on définit trois chaînes de bits IV, *ipad* et *opad* et par défaut $\mathsf{Hprf}^{\mathsf{Hash}}_{ipad,opad}(IV;k,x)$ est égale à :

$$\mathsf{Hprf}^{\mathsf{Hash}}_{ipad,opad}(IV;k,x) = \mathsf{Nprf}^{\mathsf{Hash}}\left(h(IV,k\oplus ipad), h(IV,k\oplus opad), x\right).$$

Cette similarité de construction nous permet d'établir un résultat similaire à ceux présenter ci-dessus, et ce grâce à une preuve identique : puisque HPRF est la composée de NPRF, qui est une bonne PRF, avec un extracteur d'entropie calculatoire, la sortie de HPRF paraît uniformément distribuée. On remarquera tout de même que dans la suite, nous avons uniquement besoin de choisir au hasard IV, alors que *ipad* et *opad* restent fixes.

Puisque NPRF est la concaténation et la composition de plusieurs NMAC, sa resistance en PRF se ramène facilement à la résistance en PRF de NMAC. Le nombre v de concaténations dépend de la taille de sortie de la fonction de hachage et influe sur la sécurité en PRF de NPRF. Le théorème suivant établit la sécurité de NPRF en tant que PRF.

Théorème 6.7. *Soit $u \geq 1$, $t \geq 1$ et soit $h : \{0,1\}^n \times \{0,1\}^b \to \{0,1\}^n$ une famille de fonctions. Soit $\mathcal{A}$ un adversaire en PRF contre $\mathsf{Nprf}^{\overline{H}}$ construite avec v concaténations de $\mathsf{Nmac}^{\overline{H}}$. L'adversaire $\mathcal{A}$ fait au plus q requêtes, chacune constituées d'au plus u blocs, et a une complexité temporelle d'au plus T. Il existe un adversaire $\mathcal{A}'$ contre* NMAC *tel que :*

$$\mathsf{adv}^{\mathtt{prf}}_{\mathsf{Nprf}^{\overline{H}}}(\mathcal{A}) \leq \mathsf{adv}^{\mathtt{prf}}_{\mathsf{Nmac}^{\overline{H}}}(\mathcal{A}') + qv^2/2^n.$$

Par ailleurs, $\mathcal{A}'$ a une complexité temporelle d'au plus $T + \mathcal{O}(qv)$ et fait au plus $2qv$ requêtes constituées chacune d'au plus u blocs.

Démonstration. Cette démonstration est aisée : l'attaquant $\mathcal{A}'$ contre NMAC lance l'attaquant $\mathcal{A}$ contre NPRF et à chacune des q requêtes x_i de $\mathcal{A}$, il pose au challenger les $2v$ requêtes nécessaires pour pouvoir calculer la sortie de NPRF sur x_i ; à la fin du jeu il fait suivre la réponse renvoyée par $\mathcal{A}$. Si le challenger utilise NMAC pour répondre aux requêtes, alors $\mathcal{A}'$ simule parfaitement NPRF et la probabilité que $\mathcal{A}'$ renvoie 1 dans ce cas est égale à la probabilité que $\mathcal{A}$ renvoie 1 lorsqu'il fait face à NPRF. Lorsque le challenger utilise une fonction parfaitement aléatoire pour répondre aux requêtes, alors $\mathcal{A}'$ simule parfaitement un oracle aléatoire, sauf si, suite à requête x de la part de $\mathcal{A}$, il existe deux indices $i \neq j$ tels que $W(i) = W(j)$, ce qui arrive avec probabilité inférieure à $v^2/2^n$ et ce, pour chacune des q requêtes faites par $\mathcal{A}$. Dans ce cas, la probabilité que $\mathcal{A}'$ réponde 1 est égale à la probabilité que $\mathcal{A}$ réponde 1, sachant qu'il n'y a pas eu de collision, plus la probabilité qu'il réponde 1, sachant qu'il y a eu une collision. Cette dernière probabilité est inférieure à la probabilité qu'il y ait une collision, soit $q \cdot v^2/2^n$. L'avantage de $\mathcal{A}$ est donc inférieur à l'avantage de $\mathcal{A}'$ auquel il faut ajouter le terme $qv^2/2^n$. □

Nous allons maintenant nous attacher à montrer que le secret maître généré dans TLSest indistinguable d'une chaîne de bits parfaitement aléatoire. On rappelle que ce secret maître mk fait 384 bits et est dérivé du secret initial pmk en calculant :

$$\begin{aligned} mk &= \mathsf{Hprf}^{\mathsf{Hash}}_{ipad,opad}(IV; pmk, \text{“}\mathtt{master\_secret}\text{”}\|rand_c\|rand_s) \\ &= \mathsf{Nprf}^{\mathsf{Hash}}\left(h(IV,pmk\oplus ipad), h(IV,pmk\oplus opad), \text{“}\mathtt{master\_secret}\text{”}\|rand_c\|rand_s\right), \end{aligned}$$

où “master_secret” est un label et où IV est considérée choisie aléatoirement avant chaque nouveau calcul de mk.

6.4.2 Résultats de sécurité

Dans cette section, nous adaptons les théorèmes des sections précédentes au cas de TLS. Tout d'abord nous donnons les résultats de sécurité pour une longue clef, c'est-à-dire pour une clef constituée de s blocs avec $s \geq 2$. On souligne la similarité du théorème suivant avec le théorème 6.6. Les deux théorèmes sont démontrés de la même manière, NPRF et NMAC jouant exactement le même rôle, à l'exception qu'à la fin de la preuve du théorème suivant la sécurité en PRF contre NPRF est réduite à celle de NMAC.

Théorème 6.8. *Soit h une famille de fonctions de $\{0,1\}^n \times \{0,1\}^b$ dans $\{0,1\}^n$. Soit ipad et opad deux chaînes de b bits et soit $\Phi = \{\Delta_{ipad}, \Delta_{opad}\}$. Soit $\mathcal{A}$ un adversaire pf-cRE contre* HPRF*, qui a une complexité temporelle d'au plus T, qui génère des labels constitués d'au plus ℓ blocs et une distribution sur $\{0,1\}^{sb}$, avec $s \geq 2$ blocs, et de min-entropie supérieure à m. Supposons que* HPRF *est la concatenation de v appels à* HMAC*. Alors il existe une adversaire RKA $\mathcal{A}_2$ contre $\widehat{h}$ et trois adversaire PRF $\mathcal{A}_1$, $\mathcal{A}_3$, $\mathcal{A}_4$ tels que $\mathsf{adv}^{\mathtt{pf-cre}}_{\mathit{Hmac}^{\overline{H}}}(\mathcal{A})$ est majoré par :*

$$\sqrt{2^t\left(3\cdot 2^{-m} + 2s\cdot \mathsf{adv}^{\mathtt{prf}}_h(\mathcal{A}_1)\right)} + \mathsf{adv}^{\mathtt{rka}}_{\widehat{h}}(\mathcal{A}_2)$$
$$+\mathsf{adv}^{\mathtt{prf}}_h(\mathcal{A}_3) + 4v^2\ell\cdot \mathsf{adv}^{\mathtt{prf}}_h(\mathcal{A}_4) + \frac{2v^2}{2^t} + \frac{v^2}{2^n}$$

où $\mathcal{A}_1$ et $\mathcal{A}_2$ font au plus 2 requêtes et ont une complexité temporelle T, $\mathcal{A}_3$ fait $2v$ requêtes et a une complexité temporelle de T et $\mathcal{A}_4$ fait au plus 2 requêtes et a une complexité temporelle $\mathcal{O}(\ell \cdot T_h)$.

Démonstration. Comme expliqué ci-dessus, cette démonstration suit les mêmes étapes que la démonstration du théorème 6.6, à la différence qu'à la fin de la preuve la sécurité en PRF contre NPRF est réduite à celle de NMAC, qui est elle-même ramenée à la sécurité contre h. □

Nous allons maintenant nous intéresser à la sécurité de la construction de dérivation de clefs de TLS pour $s = 1$. Quand la clef n'est pas plus grande que le bloc, de façon similaire à 6.5, nous pouvons établir un résultat de sécurité dans lequel intervient le terme $\sqrt{2^{2n}(2^{-m}+\varepsilon)}$ pour un certain ε plus petit que 2^{2n}. Ce terme est petit pour SHA-1, mais il est plus grand que 1 pour SHA-384, puisque $b = 2n$ et que $m \leq b$. C'est pourquoi nous adaptons les hypothèses de sécurité et supposons que $\widehat{h}$ est une PRF résistante contre les attaques à clefs liées quand la clef est une chaîne de bits de min-entropie au moins m ($m = n$ pour les attaques à clefs liées classiques). Plus précisément, le but et les capacités d'un attaquant en clefs liés de min-entropie m, à la différence qu'au début du jeu, l'adversaire génère une distribution efficacement échantillonable de min-entropie au moins m, l'envoie au challenger et ce dernier choisi une clef selon cette distribution. Nous dirons dans la suite que $\widehat{h}$ est résistant contre les m-attaques à clefs liées et nous noterons par $\mathsf{adv}^{\mathtt{m-rka}}_{\widehat{h},\Phi}(\mathcal{A})$ l'avantage de l'adversaire m-RKA contre $\widehat{h}$.

Cette hypothèse, peu classique, ne peut pas être réduite à la sécurité de $\widehat{h}$ contre les attaques à clefs liées. En effet, l'hypothèse de PRF suppose que la clef est uniformément distribuée et une bonne PRF quand la clef est uniformément distribuée n'est pas forcément une bonne PRF dès qu'elle ne l'est plus, même pour une clef à forte min-entropie. Considérons à titre de contre-exemple la fonction suivante : soit f la famille de fonctions de $\{0,1\}^n \times \{0,1\}^n$ dans $\{0,1\}^n$ définie par $f_K(x) = K \oplus x$. Si K est choisi au hasard dans $\{0,1\}^n$, la famille de fonction est un famille de fonctions aléatoires parfaites contre les adversaires limités à faire une seule requêtes. Par contre, si $K = K'\|0$ où K' est choisie uniformément dans $\{0,1\}^{n-1}$, K est une chaîne de

n bits de min-entropie $n-1$ et f n'est plus sûre du tout contre la même classe d'adversaires, puisque le bit de poids faible de la sortie de f peut être deviné avec bonne probabilité.

Théorème 6.9. *Soient ipad et opad deux chaînes de b bits, soit $\Phi = \{\Delta_{ipad}, \Delta_{opad}\}$ et soit h une famille de fonctions de $\{0,1\}^n \times \{0,1\}^b$ dans $\{0,1\}^n$ qui est résistante contre les m-attaques à clefs liées. Soit $\mathcal{A}$ un adversaire cRE contre* Hprf *qui a une complexité temporelle T, génère des labels constitués d'au plus ℓ blocs et une clef d'au plus 1 bloc et une min-entropie d'au moins m. Supposons que* Hprf *est la concaténation de v* Hmac*. Alors il existe un adversaire m-RKA $\mathcal{A}_0$ et deux adversaires PRF $\mathcal{A}_1$ et $\mathcal{A}_2$ tels que l'avantage $\mathsf{adv}^{\mathtt{pf-cre}}_{\mathit{Hprf}^{\overline{H}}}(\mathcal{A})$ est borné par :*

$$\mathsf{adv}^{\mathtt{m-rka}}_{\widehat{h},\Phi}(\mathcal{A}_0) + \mathsf{adv}^{\mathtt{prf}}_{h}(\mathcal{A}_1) + 4v^2\ell \cdot \mathsf{adv}^{\mathtt{prf}}_{h}(\mathcal{A}_2) + \frac{2v^2}{2^t} + \frac{v^2}{2^n}$$

où $\mathcal{A}_0$ fait au plus 2 requêtes et a une complexité temporelle d'au plus T, $\mathcal{A}_1$ fait au plus $2v$ requêtes et a une complexité temporelle d'au plus T et où $\mathcal{A}_2$ fait au plus 2 requêtes et a une complexité temporelle de $\mathcal{O}(\ell \cdot T_h)$, où T_h désigne le temps de calcul de h.

6.4.3 Paramètres pratiques

La norme de TLS impose que le secret-maître soit long de 384 bits. Par conséquent, si on utilise SHA-384 comme fonction sous-jacente, $v = 1$, $n = 512$ et $t = 384$. Le label et les deux *nonces*, une fois concaténés, forment une chaîne de 616 bits, ce qui est plus petit que la taille du bloc de SHA-384, qui vaut 1024 bits, c'est pourquoi en pratique $\ell = 1$.

Soit h la fonction de compression de SHA-384. Supposons que le meilleur adversaire PRF contre h en temps T soit l'adversaire qui effectue une attaque exhaustive. L'avantage de ce dernier est donc égal à $(T/T_h)/2^n$. De même supposons que le meilleur attaquant m-RKA connu contre h avec une complexité temporelle égale à T et avec $\Phi = \{\Delta_{ipad}, \Delta_{opad}\}$ soit celui qui réalise une recherche exhaustive en commençant par les éléments les plus probables. Son avantage est plus petit que $(T/T_h)/2^m$.

Nous examinons dans ce contexte la sécurité pratique de la dérivation de la clef quand le secret maître est plus petit que la taille du bloc et quand il est plus long. Pour une clef constituée de $s = 2$ blocs, l'avantage pf-cRE d'un adversaire de temps T est plus petite que $2^{-383} + 7(T/T_h) \cdot 2^{-512} + \sqrt{7(T/T_h) \cdot 2^{-128}}$ si $m \geq 512$. Cela implique une sécurité de 62 bits si $m \geq 512$.

Pour une petite clef, l'avantage cRE d'un adversaire en temps T est plus petit que $2^{-383} + (T/T_h)\left(6 \cdot 2^{-512} + 2^{-m}\right)$. Cela implique une sécurité d'au moins 124 bits si $m \geq 128$.

Dans le cas de RSA, le secret maître est long de 384 bits, ce qui est plus petit que la taille du bloc qui est égale à 1024 bits. Puisque sa min-entropie est égale à 368 bits, sa sécurité est d'au moins 124 bits.

Dans le cas Diffie-Hellman, si l'hypothèse DDH est vraie, alors l'élément Diffie-Hellman résultant de l'échange est indistinguable pour un adversaire d'un élément aléatoire dans le groupe. Par conséquent, dans ce cas, si l'échange de clefs est effectuée sur un sous groupe G de $\mathbb{Z}_p^\star$, où p est un nombre premier de 1024 bits, un sous groupe de 256 bits suffit pour assurer une sécurité de 124 bits. Si p est strictement plus grand que 1024 bits, alors la taille de l'élément devient plus grande que la taille du bloc et G doit faire au moins 512 bits pour que les preuves garantissent une sécurité supérieure à 62 bits.

6.4.4 Quand l'IV n'est plus aléatoire

Nos preuves de sécurité reposent sur le fait que l'*IV* est choisie au hasard avant chaque extraction d'un secret maître (et pour Hmac, il faut également qu'*ipad* et *opad* soit choisies

au hasard). Cependant l'IV, *ipad* et *opad* sont fixées une fois pour toutes dans les normes de HMAC [BCK97] et celui des fonctions de hachage [SHA95, Riv92] et ne peuvent varier. Par conséquent, il peut sembler que nos preuves n'aient pas d'intérêt pratique. Heureusement, ce n'est pas le cas.

En effet, notre définition de l'extracteur d'aléa calculatoire permet à l'adversaire de faire uniquement une requête pour deviner b. Cependant, on pourrait autoriser l'adversaire à faire au plus q requêtes avec la même IV. Dans ce cas, un argument hybride permet d'établir que l'avantage de l'adversaire est majorée par q fois l'avantage d'un adversaire contre le cas à 1 requête.

Montrons le. Soient Ext un extracteur d'entropie calcultoire et $\mathcal{A}$ un adversaire cRE contre Ext ayant droit à q requêtes avec le même IV. Soit $\mathcal{A}_i$, pour $1 \leq i \leq q$, l'adversaire cRE ayant droit à 1 requête suivant : $\mathcal{A}_i$ lance $\mathcal{A}$, fait suivre au challenger la distribution $\mathcal{D}_{\mathcal{A}}$ que ce dernier lui envoie, le challenger lui répond par (IV, Y) ; il choisit alors $X_1, \ldots, X_{i-1}$ selon $\mathcal{D}_{\mathcal{A}}$ de façon indépendante et calcule $Y_j = \mathsf{Ext}(IV, X_j)$, il choisit $Y_{i+1}, \ldots, Y_q$ de façon uniforme dans $\mathcal{D}om$ et fixe $Y_i = Y$; il envoie alors $(IV, Y_1, \ldots, Y_q)$ à $\mathcal{A}$ qui répond par un bit b' ; $\mathcal{A}_i$ finit en faisant suivre le bit b' au challenger. L'argument hybride nous dit que la probabilité de succès de $\mathcal{A}'$ est inférieure à la somme des probabilités de succès des $\mathcal{A}_i$, ce qui est inférieur à q fois la probabilité de succès du meilleur $\mathcal{A}_i$, c'est ce qu'il fallait démontrer.

Cela implique que si IV, *ipad* et *opad* ont été générées aléatoirement quand les normes ont été écrites, alors l'avantage de tout adversaire cRE contre la fonction d'extraction de clefs de TLS ou contre HMAC croît linéairement avec le nombre d'extractions de secret maître faites. D'ailleurs, une hypothèse similaire a déjà été faite par Barak *et al.* [BST03] avec les mêmes conséquences pour les bornes de sécurité.

On peut remarquer que la démonstration précédente fonctionne car IV est rendue publique par le challenger et donc l'adversaire $\mathcal{A}_i$ peut simuler le choix de $Y_1, \ldots, Y_{i-1}$. On peut donc établir un résultat similaire dans le cas particulier du *leftover hash lemma*, c'est ce qui a été prouvé par Shoup dans son livre [Sho05] (voir théorème 6.22.), ou pour n'importe quel autre extracteur d'entropie fort.

Au contraire, on ne peut pas établir ce résultat pour les extracteurs d'entropie qui ne sont pas forts, ni pour les familles de fonctions pseudo-aléatoires car elles ne dévoilent pas leur clef. On connaît même des familles de fonctions pseudo-aléatoires qui sont très sûres contre des adversaires faisant exactement 1 requête, et qui sont faibles contre des adversaires pouvant faire 2 requêtes (par exemple la famille de fonctions qui à tout x associe $K \oplus x$ où K est la clef choisie).

Quatrième partie

Construction de fonctions de hachage

7

Fonctions de hachage construites à partir de permutations

Sommaire

Les fonctions de hachage ont été évoquées à plusieurs reprises tout au long de ce mémoire, parfois de façon indirecte : au chapitre 1, nous avons fait appel à un oracle aléatoire, oracle aléatoire qui n'est autre qu'une modélisation idéale d'une fonction de hachage, au chapitre 2 nous avons évoqué l'idée d'utiliser des fonctions de hachage comme fonctions pseudo-aléatoires car c'est une hypothèse de plus en plus considérée, au chapitre 4 nous avons étudié le mode cascade et Hmac comme extracteur d'entropie calculatoire. Comme on le voit, elles sont une primitive très utilisée pour l'établissement d'un canal sécurisé et particulièrement lorsqu'il s'agit de dérivation de clefs. Leur niveau sécurité a donc un impact immédiat sur le niveau de sécurité de l'ensemble et pour cette raison, il est intéressant d'étudier leur solidité. C'est ce que nous faisons dans les deux chapitres suivants. Plus précisément nous analysons certains modes opératoires utilisés pour construire des fonctions de hachage.

Il est d'autant plus utile d'examiner la sécurité des fonctions de hachage que dans un passé récent, elles ont été sujettes à de nombreuses attaques en collision [WLF+05, WYY05a, WY05, WYY05b, BCJ+05, Kli06, JP07], révélant des faiblesses dans des fonctions de hachage en lesquelles on avait confiance, comme MD5 ou SHA-1. Pour cette raison, plusieurs nouvelles fonctions de hachage ont été proposées telles que SMASH [Knu05] ou Radiogatún [BDPA, BDPA08]. La plupart des constructions existantes utilisent des algorithmes de chiffrement par bloc et ce pour deux raisons : d'une part parce que l'on sait en construire qui soient efficaces et sûrs, d'autre part car les modes opératoires à base d'algorithmes de chiffrement par blocs que l'on a conçus sont prouvés sûrs.

En effet, plusieurs modes pour les fonctions de compression ont été proposés au début des années 1990, dont notamment le célèbre mode de Davies-Meyer utilisé pour toute la famille MDx, et ont été analysés en détails [BRS02, PGV94]. Plusieurs offrent de très bons niveaux de sécurité. Cependant, en ces temps de remise en question des fonctions existantes, d'autres voies de recherche sont tout de même explorées et de nouveaux modes opératoires sont étudiés. L'une d'entre elle est de construire des fonctions de hachage avec des algorithmes de chiffrement par bloc dont la clef est fixée. Le but originel de ces constructions était d'augmenter l'efficacité : les fonctions de compression existantes nécessitent un calcul de clef et une évaluation de l'algorithme de chiffrement à chaque itération, alors qu'en fixant la clef, on économise le calcul de clef qui est souvent une étape coûteuse. C'est pour cette raison que Knudsen a proposé SMASH [Knu05] en 2005. Cependant, Pramstaller *et al.* [PRR05] ont cassé SMASH en collision. Knudsen a alors proposé deux modifications [Knu05] pour éviter cette attaque. Deux ans plus tard, Lamberger *et al.* [LPRR07] ont cassé en seconde préimage la version originale de SMASH, mais leur attaque ne s'applique pas aux deux nouvelles versions. Ces dernières sont restées inviolés depuis 2005, cependant aucune preuve ne vient étayer leur sécurité.

À Eurocrypt 2005, Black *et al.* [BCS05] ont étudié s'il était possible de construire une fonction de hachage sûre dans laquelle on effectuait qu'un seul calcul de clef, tout au début. Ils ont montré qu'un attaquant tout puissant était capable de trouver une collision sur la fonction de hachage en effectuant $\mathcal{O}(n)$ chiffrements, ce qui implique que l'on ne peut pas espérer *prouver* une telle construction dans les modèles habituels de sécurité. Ce résultat d'impossibilité a ouvert la voie à de nouvelles constructions, dans lesquelles on effectue plusieurs chiffrements à chaque itération (par plusieurs il faut comprendre strictement plus qu'un) et ce avec des clefs fixées mais différentes. Dans la suite, on dit que de telles fonctions de hachage sont *basées sur k permutations*, où k représente le nombre d'appels au chiffrement par itération, pour signifier que la clef est fixée.

À Eurocrypt 2008, Rogaway et Steinberger [RS08b] ont étendu ce résultat au cas à plusieurs permutations. Ils ont exhibé des attaques en collision et en préimage, donnant ainsi des bornes supérieures sur la sécurité. Par exemple, pour des constructions à base de 2 permutations, ils ont donné une attaque en collision et une en préimage qui nécessitent toutes deux $\mathcal{O}(2^{n/2})$ chiffrements. La meilleure construction à base de 2 permutations ne pourra donc pas être garantie sûre en collision et en préimage si $\mathcal{O}(2^{n/2})$ chiffrements sont effectués. De ce résultat il découle une question évidente : existe-t-il des constructions qui atteignent les bornes supérieures prouvées par Rogaway et Steinberger ? Et sinon, peut-on encore améliorer ces bornes ?

C'est pour répondre en partie à cette question que deux papiers viennent de paraître à Icalp [SS08] et Crypto [RS08a] cette année. Ils examinent le cas à 3 permutations et proposent des constructions. Dans ce chapitre, nous étudions la même problématique, mais examinons le cas à 2 permutations. Dans un premier temps, nous montrons, en donnant une attaque, que la construction SMASH proposée par Knudsen, malgré les améliorations proposées, peut être cassée en collision. Cependant, SMASH peut être améliorée en ajoutant une permutation dans la fonction de compression. La fonction de hachage ainsi créée est maintenant basée sur deux permu-

tations. Nous avons découvert que cette construction avait déjà été proposée dans la thèse [Tho05] de Thomsen, alors qu'il est sous la direction de Knudsen, mais elle n'avait pas été analysée. Nous montrons ici qu'elle peut être prouvée sûre en collision et en préimage. Cependant, comme nous allons le voir, si cette construction est optimale en préimage, la preuve de résistance aux collisions n'atteint pas la bornes établie par Rogaway et Steinberger. Nous faisons remarquer que les résultats évoqués dans ce chapitre ont donné lieu à une publication [FSZ08].

7.1 Construction et modèle de sécurité

7.1.1 Remarques sur le modèle de sécurité

Dans ce chapitre, nous nous plaçons dans le cadre du modèle du chiffrement idéal, modèle décrit en introduction dans la sous-section 2.4.1. On rappelle que dans ce modèle l'algorithme de chiffrement par bloc est modélisé par une famille de permutations parfaitement aléatoires et que l'adversaire a accès à deux oracles E et E^{-1}, ce que l'on note $\mathcal{A}^{E,E^{-1}}$. Il peut poser au plus Q requêtes aux oracles : à chacune des requêtes (K_i, X_i) qu'il fait à E il reçoit en réponse $Y_i = E(K_i, X_i)$ et à chacune des requêtes (K_i, Y_i) qu'il fait à E^{-1} il reçoit en réponse $X_i = E^{-1}(K_i, Y_i)$.

Dans les constructions à base de deux permutations que nous présentons dans ce chapitre, les clefs k_1 et k_2 choisies pour la construction sont publiques et données à l'adversaire. Puisque les autres permutations $E(k, \cdot)$ pour $k \neq k_1, k_2$ sont indépendantes de $E(k_1, \cdot)$ et $E(k_2, \cdot)$, nous supposons sans perte de généralité que l'adversaire ne fait pas de requêtes de la forme (k, x) à E ou de requêtes de la forme (k, y) à E^{-1} pour $k \neq k_1, k_2$. Par soucis de simplicité, nous utiliserons les notations $\pi_1 = E(k_1, \cdot)$ et $\pi_2 = E(k_2, \cdot)$ et donnerons simplement accès à π_1, π_1^{-1}, π_2 et π_2^{-1}. On remarquera que dans ce cadre, on ne repose pas sur toute la puissance du modèle du chiffrement idéal, nous supposons seulement que π_1 et π_2 sont choisies indépendamment et de façon parfaitement aléatoire parmi l'ensemble de toutes les permutations. Nous n'utilisons pas le fait que pour tout $k \neq k_1, k_2$, $E(k, \cdot)$ est une permutation choisie au hasard aléatoirement dans l'ensemble de toutes les permutations.

Dans la suite nous allons successivement casser la résistance aux collisions de SMASH et prouver la résistance aux collisions de notre mode opératoire. La définition de la résistance aux collisions a été donnée en introduction, sous-section 2.5.1, mais rappelons la succinctement. Si H est une fonction de hachage, le but d'un adversaire en collision contre H est de trouver deux messages différents M et M' tels que $H(M) = H(M')$ (les fonctions de hachage que nous étudions dans ce chapitre sont paramétrées par les oracles E et E^{-1}). La probabilité que l'adversaire réussisse à casser la résistance aux collisions de H est notée $\mathsf{adv}_H^{\mathsf{Coll}}(\mathcal{A})$ et $\mathsf{adv}_H^{\mathsf{Coll}}(Q)$ désigne le maximum des probabilités de succès $\mathsf{adv}_H^{\mathsf{Coll}}(\mathcal{A})$, où ce maximum est pris sur tous les adversaires $\mathcal{A}$ qui font au plus Q requêtes.

Nous étudierons également dans ce chapitre, la résistance aux préimages de notre proposition. Cette définition est également donnée en introduction, sous-section 2.5.3, mais nous la rappelons brièvement. Soit Y un élément choisi par le challenger (voir l'introduction pour plus de détails, la manière dont Y est choisi n'influe pas sur les résultats ici). Le but d'un adversaire en préimage est de trouver M tel que $H(M) = Y$. La probabilité que l'adversaire $\mathcal{A}$ réussisse à casser la résistance aux préimages de H est notée $\mathsf{adv}_H^{\mathsf{pre}}(\mathcal{A})$ et $\mathsf{adv}_H^{\mathsf{pre}}(Q)$ désigne le maximum des probabilités de succès $\mathsf{adv}_H^{\mathsf{pre}}(\mathcal{A})$, où ce maximum est pris sur tous les adversaires $\mathcal{A}$ qui font au plus Q requêtes.

Nous supposerons dans ce chapitre qu'un adversaire ne pose jamais une requête pour laquelle il connaît déjà la réponse, c'est-à-dire qu'il ne fait jamais deux fois la même requête et que s'il fait la requête (k, x) à E, lequel répond par y, alors il ne fait pas la requête (k, y) à E^{-1}, et vice

et versa. Par ailleurs, nous supposons que quand un adversaire renvoie (M, M') (ou M seul) à la fin du jeu, il a déjà calculé $H(M)$ et $H(M')$, c'est-à-dire qu'il a déjà fait toutes les requêtes nécessaires pour calculer $H(M)$ et $H(M')$. Ces hypothèses simplifient les preuves et ne sont pas très restrictives, puisqu'il est facile de transformer un adversaire quelconque $\mathcal{A}$ en un adversaire $\mathcal{A}'$ qui respecte ces règles et qui a le même avantage. Il suffit que $\mathcal{A}'$ lance $\mathcal{A}$, conserve en mémoire les requêtes faites par $\mathcal{A}$ et à chaque fois que celui-ci fait une nouvelle requête, il vérifie que cette requête n'est pas déjà dans la liste : si elle est dans la liste, il peut répondre à $\mathcal{A}$, sinon il fait suivre la requête au challenger et fait suivre la réponse du challenger à $\mathcal{A}$. Tout à la fin il fait si nécessaire toutes les requêtes nécessaire pour calculer $H(\mathcal{M})$ et $H(\mathcal{M}')$ puis fait suivre M et M' au challenger. On voit facilement que $\mathcal{A}$ et $\mathcal{A}'$ ont la même probabilité de succès.

Enfin, faisons une dernière remarque sur la méthode employée pour calculer la complexité d'une attaque. On en trouve habituellement deux dans la littérature. D'une part, on peut définir cette complexité comme étant égale à la complexité temporelle de l'adversaire qui mène l'attaque. C'est bien sûr à cette complexité que l'on s'intéresse en pratique et c'est la complexité que Knudsen considère dans son papier à propos de SMASH [Knu05]. C'est la complexité que nous considèrerons pour nos attaques pratiques sur SMASH. Nous appellerons cette complexité la *complexité temporelle*. D'autre part, la complexité des attaques peut être mesurée par le nombre de requêtes faites aux oracles. Cette complexité est très souvent utilisée dans les preuves [PGV94, BRS02], ou au contraire pour prouver qu'il est impossible d'établir une preuve [BCS05, RS08b], car elle est nettement plus simple à évaluer ou à majorer. De plus, elle minore la complexité temporelle : si on prouve qu'un adversaire doit faire au moins Q requêtes pour casser la résistance aux collisions, cela signifie qu'il devra faire au moins Q étapes de calculs, mais parfois nettement plus. Nous utiliserons donc cette complexité dans nos preuves et l'appellerons la *complexité en requêtes*.

7.1.2 Constructions de SMASH et sa généralisation

Dans cette sous-section, nous introduisons successivement le schéma originel de SMASH, puis les modifications proposées par Knudsen et enfin la construction que nous proposons, qui est une généralisation du schéma de SMASH. Toutes ces versions de SMASH sont fondées sur le mode cascade, aussi nous ne décrivons que les fonctions de compression dans chaque cas.

Smash

Nous abordons donc tout d'abord la version originelle de SMASH. Soit $\pi = E(k, \cdot)$ une permutation aléatoire, $IV \in \{0,1\}^n$ une chaîne de bits fixée, $\theta \neq 0, 1$ un élément fixé de $GF(2^n)$, le corps fini à 2^n éléments, et soit $smash\colon \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ la fonction définie par :

$$smash^\pi(h, x) = \pi(h \oplus x) \oplus h \oplus \theta \cdot x,$$

où $\cdot$ désigne la multiplication dans $GF(2^n)$.

Étant donné (IV, π, θ), les hachés du message $x = (x_1, \ldots, x_\ell) \in \{0,1\}^{n \cdot \ell}$ est $SMASH(x) = h_{\ell+1}$ où $h_0 = \pi(IV) \oplus IV = smash^\pi(IV, 0^n)$, $h_k = smash^\pi(h_{k-1}, x_k)$, pour tout $1 \leq k \leq \ell$, et $h_{\ell+1} = \pi(h_\ell) \oplus h_\ell = smash^\pi(h_\ell, 0^n)$.

Versions améliorées de SMASH

Après l'attaque de [PRR05], Knudsen a proposé deux moyens de modifier le schéma [PRR05, Knu05], à savoir : "Une [possibilité] est d'utiliser une permutation différente pour chaque itération.

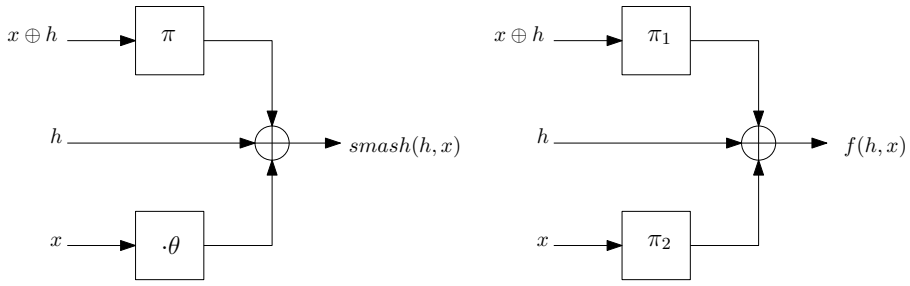


FIG. 7.1 – La fonction de compression de SMASH et notre fonction de compression basée sur 2 permutations.

Une autre [possibilité] est d'utiliser une fonction de compression sûre (...) tous les t blocs de message, pour disons $t = 8$ ou $t = 16$".

La modification qui consiste à utiliser une permutation différente pour chaque itération, nous l'appelerons la *première modification* et la modification, qui consiste à utiliser une fonction de compression sûre (telle $\pi(h) \oplus h$ par exemple) tous les traitements de t blocs de message, nous l'appellerons la *seconde modification*.

Notre généralisation

Les attaques présentées ici, de même que celle de [PRR05], utilisent les propriétés particulières de la multiplication par θ pour attaquer la fonction de hachage. Cette multiplication est une source de faiblesse, c'est pourquoi c'est elle que nous nous sommes attachés à modifier. On peut remarquer que la multiplication par $\theta \neq 0$ est une permutation sur $\{0,1\}^n$, choisir θ au hasard dans $GF(2^n)$ privé de 0, c'est choisir une permutation au hasard parmi $2^n - 1$ possibles, permutations qui ont toutes en commun des propriétés structurelles particulières. Plutôt que de choisir comme permutation la multiplication par un certain θ, nous proposons de choisir cette permutation au hasard parmi l'ensemble des permutations possibles. Il sera beaucoup plus difficile pour l'adversaire d'exploiter une propriété particulière à toutes les permutations possibles. En ce sens nous proposons une généralisation du schéma de Knudsen. Comme nous l'avons déjà précisé, nous avons découvert que cette généralisation avait déjà été proposée indépendamment par [Tho05] mais n'a pas été analysée.

Nous faisons remarquer que choisir, dans le modèle du chiffrement idéal, une permutation au hasard parmi toutes les permutations possibles, revient, en pratique, à utiliser un algorithme de chiffrement par bloc dont la clef a été choisie au hasard.

Nous modifions la fonction de compression proposée par Knudsen, mais également la fonction de hachage globale, puisque suite à la transformation de la fonction de compression, particulariser les premières et dernières itérations ne semble plus nécessaire. La description de la fonction que nous proposons est donc la suivante.

Soient $\pi_1 = E(k_1, \cdot)$ et $\pi_2 = E(k_2, \cdot)$ deux permutations aléatoires, $IV \in \{0,1\}^n$ une chaîne de bits fixée et $f\colon \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ la fonction de compression définie par :

$$f(h, x) = \pi_1(x) \oplus \pi_2(x \oplus h) \oplus h.$$

Étant donné (IV, π_1, π_2), le haché du message $x = (x_1, \ldots, x_\ell) \in \{0,1\}^{n \cdot \ell}$ est $H(x) = h_\ell$ où $h_0 = IV$, $h_k = f(h_{k-1}, x_k)$ pour tout $1 \leq k \leq \ell$.

FORMAT. Les constructions introduites ci-dessus imposent que la taille du message soit un multiple de la taille du bloc. Pour étendre cette construction à des messages de taille arbitraire, on

peut appliquer une fonction injective simple au message en lui imposant un format particulier. On peut par exemple utiliser la mise en forme classique, également proposée pour SMASH : on ajoute un '1' suivi d'autant de '0' que nécessaire pour obtenir une taille multiple de la taille du bloc. Dans SMASH, on doit aussi rajouter ce que l'on appelle le renforcement de Merkle-Damgård. Ce dernier consiste à rajouter à la fin du message la taille de ce message. La longueur du message est codé sur un nombre fixé de bits, le plus souvent 64 bits. De ce fait, on ajoute donc un '1' suivi d'autant de '0' que nécessaire pour obtenir une taille multiple de la taille du bloc moins 64 bits, puis l'encodage de la taille du message sur 64 bits. Le message M après mise en forme devient donc $x = M \,\|\, 10^s \,\|\, |M|$ où s est tel que la taille de x est un multiple de n, la taille du bloc.

Nous conseillons également d'ajouter le renforcement de Merkle-Damgård à notre construction, puisque, même si la preuve de sécurité que nous sommes capables d'établir ne le nécessite pas, la meilleure attaque connue contre le schéma sans renforcement est strictement plus efficace que la meilleure attaque connue contre la construction avec le renforcement.

7.2 Une attaque en collision contre toutes les versions précédentes de SMASH

Dans cette section, nous allons présenter une attaque en collision contre SMASH dont la complexité temporelle est en $\mathcal{O}(n2^{n/3})$. Puisqu'elle génère deux messages, qui provoquent une collision, et qui ne sont constitués que de 2 blocs, elle peut être montée contre la version originelle de SMASH, mais également contre les versions améliorées, dans la mesure où $t \geq 3$ (nous rappelons que t désigne le nombre d'itérations où l'on utilise la fonction de compression de SMASH avant d'utiliser une fonction de compression alternative sûre). Cette attaque repose fortement sur la linéarité de la multiplication par θ. Ensuite, nous présentons une variante de cette attaque, qui permet de diminuer le temps de calcul mais qui génère des messages plus longs. Pour cette raison, elle est valable et très efficace contre la première modification mais contre la deuxième modification son impact est plus limité. Contre la première modification, elle génère deux messages longs de $2^{\sqrt{n}-1}$ blocs et a une complexité temporelle de $\sqrt{n}2^{2\sqrt{n}}$. Cela signifie que pour $n = 256$, il y a une attaque en $c \cdot 2^{36}$ où c est une petite constante.

7.2.1 Attaque générique

Nous décrivons maintenant une attaque en collision contre toutes les versions précédentes de SMASH, notamment les deux versions améliorées, avec une complexité temporelle de $\mathcal{O}(n2^{n/3})$. L'attaque génère deux messages de deux blocs.

On fait remarquer que l'attaque générique présentée dans [BCS05] par Black *et al.* s'applique aussi à SMASH avec la première modification et trouve une collision avec une complexité en requêtes d'au plus $\mathcal{O}(2(n+1))$, mais avec une complexité temporelle plus grande que 2^n. Par conséquent, cette attaque ne contredit pas le niveau de sécurité espéré par Knudsen [Knu05], à savoir une sécurité pratique de $\mathcal{O}(2^{n/2})$. L'attaque présentée dans [PRR05] par Pramstaller *et al.* est très efficace contre la version originelle de SMASH, mais comme ils le précisent dans leur article, elle ne s'applique pas aux deux modifications proposées par Knudsen.

Dans la suite, nous allons utiliser les notations déjà introduites dans la sous-section 7.1.2. Soient π et π' deux permutations utilisées respectivement lors de la première et de la seconde itération. Soient (α_1, β_1) et (α'_1, β'_1) 2 couples tels que $\pi(\alpha_1) = \beta_1$ et $\pi'(\alpha'_1) = \beta'_1$. On définit $\gamma_1 = \beta_1 \oplus \theta \cdot \alpha_1$, $\gamma'_1 = \beta'_1 \oplus \theta \cdot \alpha'_1$, $x_1 = \alpha_1 \oplus h_0$, $h_1 = smash^{\pi}(h_0, x_1) = \beta_1 \oplus \theta \cdot \alpha_1 \oplus (\theta + 1) \cdot h_0$,

et $x'_1 = \alpha'_1 \oplus h_1$. Par conséquent, pour $h_2 = smash^{\pi'}(h_1, x'_1)$, on obtient :

$$h_2 = \gamma'_1 \oplus (\theta+1)\cdot\gamma_1 \oplus (\theta+1)^2 \cdot h_0. \quad (7.1)$$

Soient (α_2, β_2) et (α'_2, β'_2) 2 autres couples tels que $\pi(\alpha_2) = \beta_2$ et $\pi'(\alpha'_2) = \beta'_2$. On définit de façon similaire à ci-dessus $\gamma_2 = \beta_2 \oplus \theta\cdot\alpha_2$, $\gamma'_2 = \beta'_2 \oplus \theta\cdot\alpha'_2$, $x_2 = \alpha_2 \oplus h_0$, $h'_1 = smash^{\pi}(h_0, x_2) = \beta_2 \oplus \theta\cdot\alpha_2 \oplus (\theta+1)\cdot h_0$ et $x'_2 = \alpha'_2 \oplus h'_1$. Pour $h'_2 = smash^{\pi'}(h'_1, x'_2)$, on obtient :

$$h'_2 = \gamma'_2 \oplus (\theta+1)\cdot\gamma_2 \oplus (\theta+1)^2 \cdot h_0.$$

Tout d'abord, on fait remarquer que si $h_2 = h'_2$, alors $SMASH(x_1, x'_1) = SMASH(x_2, x'_2)$, puisque $SMASH(x_1, x'_1) = h_2 \oplus \pi''(h_2)$ et $SMASH(x_2, x'_2) = h'_2 \oplus \pi''(h'_2)$, où π'' désigne la permutation utilisée à la troisième itération. Notre objectif est donc d'assurer que $h_2 = h'_2$. Or, d'après les équations précédentes, on a $h_2 = h'_2$ si et seulement si $\gamma'_1 \oplus (\theta+1)\cdot\gamma_1$ est égal à $\gamma'_2 \oplus (\theta+1)\cdot\gamma_2$, ce qui est équivalent à :

$$(\theta+1)\cdot\gamma_1 \oplus (\theta+1)\cdot\gamma_2 \oplus \gamma'_1 \oplus \gamma'_2 = 0. \quad (7.2)$$

L'attaque décrite ci-dessous est une conséquence directe de cette relation.

Faisons $2q$ requêtes à π pour générer 2 suites de q éléments $(\alpha_{1,i}, \beta_{1,i})$ et $(\alpha_{2,i}, \beta_{2,i})$ et $2q$ requêtes à π' pour générer 2 suites de q éléments $(\alpha'_{1,i}, \beta'_{1,i})$ et $(\alpha'_{2,i}, \beta'_{2,i})$. Nous calculons maintenant les $\gamma_{j,i} = \beta_{j,i} \oplus \theta\cdot\alpha_{j,i}$ et $\gamma'_{j,i} = \beta'_{j,i} \oplus \theta\cdot\alpha'_{j,i}$ associés, pour $j = 1, 2$ et $1 \le i \le q$.

Si $q = 2^{n/4}$, le paradoxe des anniversaires généralisé nous dit qu'avec forte probabilité il existe un quadruplet $(\gamma_{1,a}, \gamma'_{1,b}, \gamma_{2,c}, \gamma'_{2,d})$ tel que l'équation (7.2) soit vraie (on peut trouver une preuve du paradoxe des anniversaires généralisé dans l'annexe A). Cependant trouver un tel quadruplet nécessite une complexité temporelle de $\mathcal{O}(n2^{n/2})$. Pour $q = 2^{n/3}$, l'algorithme présenté dans [Wag02] permet de trouver un quadruplet qui vérifie l'équation (7.2) en temps $\mathcal{O}(n2^{n/3})$ et en espace $\mathcal{O}(2^{n/3})$. Par conséquent, en utilisant cet algorithme, on peut monter une attaque dont la complexité en requêtes est de $\mathcal{O}(2^{n/3})$ et la complexité temporelle est de $\mathcal{O}(n2^{n/3})$, ce qui est bien plus petit que la complexité temporelle de $\mathcal{O}(2^{n/2})$ espérée.

7.2.2 Amélioration de l'attaque

L'amélioration présentée dans cette sous-section provient directement de la généralisation de l'algorithme à 4 listes [Wag02] présenté précédemment. Plus il y a de listes, plus petite est la complexité temporelle. Le principal inconvénient de cette amélioration est qu'elle génère des messages plus longs et par conséquent ne peut pas être utilisée complètement contre la seconde modification.

Supposons qu'à la place de chercher des messages sur 2 blocs, on cherche des messages sur 3 blocs. On utilise les mêmes notations que ci-dessus et on introduit π'' la permutation utilisée dans la troisième itération, (α''_1, β''_1) et (α''_2, β''_2) deux couples tels que $\pi''(\alpha''_1) = \beta''_1$ et $\pi''(\alpha''_2) = \beta''_2$, $\gamma''_1 = \beta''_1 \oplus \theta\cdot\alpha''_1$ et $\gamma''_2 = \beta''_2 \oplus \theta\cdot\alpha''_2$. Si on définit $x''_1 = \alpha''_1 \oplus h_2$, $x''_2 = \alpha''_2 \oplus h'_2$, $h_3 = smash^{\pi''}(h_2, x''_1)$ et $h'_3 = smash^{\pi''}(h'_2, x''_2)$, de même que dans précédemment on a :

$$\begin{aligned} h_3 &= \gamma''_1 \oplus (\theta+1)\cdot\gamma'_1 \oplus (\theta+1)^2\cdot\gamma_1 \oplus (\theta+1)^3\cdot h_0 \\ h'_3 &= \gamma''_2 \oplus (\theta+1)\cdot\gamma'_2 \oplus (\theta+1)^2\cdot\gamma_2 \oplus (\theta+1)^3\cdot h_0. \end{aligned}$$

Par conséquent $h_3 = h'_3$ si et seulement si $(\theta+1)^2\cdot(\gamma_1 \oplus \gamma_2) \oplus (\theta+1)\cdot(\gamma'_1 \oplus \gamma'_2) \oplus (\gamma''_1 \oplus \gamma''_2) = 0$.

Cette équation nous conduit à monter une attaque qui génère 6 listes et qui essaie de trouver un élément dans chaque liste tel que le xor de tous ces éléments est égal à 0. Ceci peut être

généralisé à des messages de k blocs de long. On peut facilement montrer que $h_k = h'_k$ si et seulement si :

$$\bigoplus_{i=0}^{k}(\theta+1)^{k-i}\cdot(\gamma_1^{(i)}\oplus\gamma_2^{(i)})=0, \tag{7.3}$$

où les notations utilisées sont la généralisation directe des notations introduites précédemment.

L'algorithme présenté dans [Wag02] trouve un $2k$-uplet en temps $\mathcal{O}(nk/(1+\log_2(2k))\cdot 2^{n/(1+\log_2(2k))})$ et nécessite $2k$ listes de taille $\mathcal{O}(2^{n/(1+\log_2(2k))})$, par conséquent il faut faire $\mathcal{O}(nk/(1+\log_2(2k))\cdot 2^{n/(1+\log_2(2k))})$ requêtes pour générer toutes ces listes. La complexité de l'attaque est minimale pour $2k=2^{\sqrt{n}}$ et dans ce cas la complexité temporelle est égale à $\mathcal{O}(\sqrt{n}\cdot 2^{2\sqrt{n}})$.

Cette amélioration peut être utilisée pour toutes les valeurs de k contre la première modification et par conséquent cette version de SMASH peut être attaquée en temps $\mathcal{O}(\sqrt{n}\cdot 2^{2\sqrt{n}})$, en générant des messages de $2^{\sqrt{n}-1}$ blocs. Pour $n=256$, cela signifie que la complexité temporelle est de $c\cdot 2^{36}$ pour une petite constante c et que les messages font 2^{15} blocs de 256 bits, c'est-à-dire 1 Mo.

Par contre, l'amélioration ne peut être appliquée que pour $k \leq t-1$ contre la seconde modification de SMASH. Par conséquent contre cette modification, la meilleure attaque a une complexité temporelle de $\mathcal{O}(nt/(1+\log_2(t))\cdot 2^{n/(2+\log_2(t-1))})$, c'est-à-dire $\mathcal{O}(n2^{n/4})$ et $\mathcal{O}(n2^{n/5})$ pour $t=8$ et $t=16$ respectivement, qui sont les valeurs proposées par Knudsen [Knu05] (on rappelle que t désigne le nombre d'itérations où on utilise la fonction de compression classique de SMASH, avant d'utiliser une fonction de compression alternative). Pour $n=256$, cela donne une complexité temporelle approximativement égale à 2^{64} et 2^{52} respectivement.

7.3 Une attaque en seconde préimage et en préimage contre les versions précédentes de SMASH

Les idées développées au chapitre précédent permettent également de monter des attaques en seconde préimage et même en préimage contre les versions de SMASH proposées par Knudsen. Il est intéressant de noter que cette attaque en préimage est la première attaque en préimage contre SMASH, puisque les résultats de Pramstaller *et al.* [PRR05] et ceux de Lamberger *et al.* [LPRR07] concernaient respectivement la collision et la seconde préimage.

7.3.1 Attaque en seconde préimage

Présentons d'abord l'attaque en seconde préimage. Pour cela nous introduisons un message $x=(x_1,\cdot,x_\ell)$ constitué de $\ell\geq 3$ blocs de n bits. On suppose que x_ℓ contient déjà le renforcement de Merkle et Dåmgard. On note par $h_0,\ldots,h_\ell$ les valeurs de chainage intermédiaires calculées lors du hachage de x. On a donc $h_0=IV$, $h_k=f(h_{k-1},x_k)$ pour tout $1\leq k\leq \ell$ et $H(x)=h_\ell$. On va chercher à construire un message $x'=(x'_1,\cdot,x'_\ell)$ de même taille que x, tel que $x'_\ell=x_\ell$ et que si $h'_0,\ldots,h'_\ell$ désignent les valeurs de chainage intermédiaires calculées lors du hachage de x', on a alors $h'_{\ell-1}=h_{\ell-1}$. On aura bien alors $h'_\ell=h_\ell$.

Pour simplifier les notations on présente l'attaque contre la version originale de SMASH, comme on l'a vu à la section précédente, il est très simple de l'adapter aux deux autres versions de SMASH. Pour trouver ce message x', il est équivalent de trouver ℓ valeurs $\gamma^{(i)}=\beta^{(i)}\oplus\theta\cdot\alpha^{(i)}$ tels que

$$\bigoplus_{i=1}^{\ell-1}(\theta+1)^{\ell-1-i}\cdot\gamma^{(i)}=h_{\ell-1}\oplus(\theta+1)^{\ell-1}\cdot IV, \tag{7.4}$$

(cette équation est issue d'une généralisation de l'équation (7.1) où $h_0 = IV$). Puisque $h_{\ell-1}$, θ, et IV sont fixés, ceci peut se faire en utilisant l'algorithme de Wagner, ce qui donne une attaque de complexité temporelle :

$$\mathcal{O}\left(\frac{n(\ell-1)}{(1+\log_2(\ell-1))}\cdot 2^{n/(1+\log_2(\ell-1))}\right).$$

On remarquera que cette attaque n'est intéressante que si $\ell \geq 3$ et ne fonctionne contre la seconde version que lorsque $t \geq \ell - 1$. Si $t \leq \ell - 1$, alors on peut utiliser l'attaque suivante en préimage.

7.3.2 Attaque en préimage

Le renforcement de Merkle et Dåmgard rend plus difficile une attaque en préimage, car dans le cas de la seconde préimage on connaît un bloc de message x_ℓ bien formaté (c'est-à-dire intégrant le renforcement) tel que $h_\ell = f(h_{\ell-1}, x_\ell)$. Ce n'est plus le cas dans le cas de la préimage, aussi notre but dans un premier temps sera de recréer un tel bloc de message. Pour cela nous allons utiliser une propriété très particulière du formatage classique du renforcement de Merkle et Dåmgard. Avant de commencer à décrire l'attaque, nous attirons l'attention du lecteur sur le fait que, sans ce renforcement, l'attaque décrite à la section précédente fonctionnerait parfaitement en préimage puisque l'on aurait pas besoin de préoccuper du formatage du dernier bloc.

Soit h la valeur dont on veut trouver une préimage et soit ℓ un entier plus petit que $2^{64} - 1$. Le but est de trouver h' et m tel que $h = f(h', m)$ et tel que m puisse être le dernier bloc d'un message de ℓ blocs. On pourra alors adapter l'attaque en seconde préimage décrite à la section précédente, pour trouver un message x de taille ℓ tel que son dernier bloc soit m et tel que son avant dernière valeur de chainage est égale à h'.

Nous introduisons la variable $\alpha = m \oplus h'$, trouver h' et m est équivalent à trouver α et m puisque

$$h = f(h', m) = \pi(h' \oplus m) \oplus h' \oplus \theta \cdot m = \pi(\alpha) \oplus \alpha \oplus (\theta + 1) \cdot m.$$

Pour trouver m on s'appuie sur le fait que le formatage impose au minimum les 65 derniers bits de m (un '1' et ℓ codé sur 64 bits), mais que les autres bits peuvent être choisis librement : il existe donc 2^{n-65} messages m différents qui peuvent convenir. On fait donc 2^{65} requêtes à π et on calcule à chaque fois la valeur $m = (\pi(\alpha) \oplus \alpha \oplus h) \cdot (\theta + 1)^{-1}$. Avec forte probabilité, une de ces valeurs est bien formatée et donc on a trouvé m et h'. La complexité temporelle de cette attaque est de l'ordre de 2^{65}.

Comme on l'a précisé précédemment, on finit l'attaque de la même manière que dans l'attaque en seconde préimage. La complexité temporelle de cette étape est

$$\mathcal{O}\left(\frac{n(\ell-1)}{(1+\log_2(\ell-1))}\cdot 2^{n/(1+\log_2(\ell-1))}\right).$$

Puisque la valeur ℓ est libre, on le choisit de telle sorte que la complexité temporelle de l'algorithme de Wagner soit la plus petite possible. Par exemple pour la version originelle de SMASH, on prend $\ell = 2^{\sqrt{n}}$, et pour $n = 256$, la complexité globale de l'attaque est donc de l'ordre de 2^{65}. Cette attaque s'adapte facilement contre la première modification et la seconde modification de SMASH, pour peu que $t \geq 3$ et donc permet d'avoir une attaque en seconde préimage contre la seconde modification de SMASH si $t \geq 3$.

7.4 Résistance en collision de la généralisation

Nous examinons maintenant la résistance aux collisions de la version généralisée que nous proposons. Tout d'abord, nous prouvons qu'une attaque requiert au moins $\Omega(2^{n/4})$ requêtes pour réussir à trouver une collision avec bonne probabilité. Ensuite, nous donnons une attaque contre le schéma avec une complexité en requêtes de $\mathcal{O}(2^{3n/8})$, mais une complexité temporelle de $\mathcal{O}(2^{3n/4})$. Le plus souvent cette attaque génère deux messages de tailles différentes et par conséquent ne peut être utilisée lorsque l'on ajoute le renforcement de Merkle-Damgård lors de la mise en forme des messages. La meilleure attaque connue dans ce cas là est une attaque en paradoxe des anniversaires avec $\mathcal{O}(2^{n/2})$ requêtes et avec une complexité temporelle en $\mathcal{O}(n2^{n/2})$.

7.4.1 Preuve de sécurité

L'attaque présentée dans [BCS05] prouve en particulier que l'on ne peut pas espérer *prouver* la résistance aux collisions de SMASH si plus de $\mathcal{O}(n)$ requêtes ont été faites. Au contraire, on montre que si on remplace la multiplication par θ par une permutation forte (modélisée par un algorithme de chiffrement idéal), alors on peut prouver qu'il faut au minimum $\Omega(2^{n/4})$ requêtes pour casser la résistance aux collisions, et par conséquent qu'une telle attaque a une complexité temporelle plus grande que $2^{n/4}$. Cette preuve est valide que l'on utilise le renforcement de Merkle-Damgård ou pas.

Pour prouver ce résultat nous allons procéder en deux temps. Tout d'abord, nous allons montrer que la probabilité de trouver une collision contre la fonction de compression f est plus petite que $Q^4/(4 \cdot 2^n)$ où Q est le nombre de requêtes faites par l'adversaire. On en déduira le résultat sur la fonction de hachage en entier H.

Il est intéressant de faire remarquer que le deuxième résultat de [RS08b] propose une attaque en collision par paradoxe des anniversaires contre toute fonction de compression basée sur 2 permutations. Sa probabilité de succès 1/2 et sa complexité en requêtes est essentiellement en $2^{n/4}$. Cela signifie que notre fonction de compression basée sur 2 permutations a une résistance en collision qui est optimale.

Théorème 7.1. *Soit $\mathcal{A}$ un adversaire en collision contre f calculatoirement non borné, qui fait au plus Q requêtes. Son avantage est majoré par :*

$$\mathsf{adv}_f^{\mathsf{Coll}}(\mathcal{A}) \leq \frac{Q^4}{4 \cdot 2^n}.$$

Démonstration. Un adversaire en collision est autorisé à faire au plus Q requêtes à π_1, π_2, π_1^{-1} ou π_2^{-1}. Nous montrons que la probabilité que cette adversaire trouve une collision pour h est majorée par $Q^4/2^n$. Les permutations π_1, π_2 sont choisies aléatoirement.

Commençons par faire remarquer que trouver $(x, h) \neq (x', h')$ tels que $f(h, x) = f(h', x')$ est équivalent à trouver $(\alpha_1, \alpha_2) \neq (\alpha_1, \alpha_2)$ tels que $\alpha_1 \oplus \beta_1 \oplus \alpha_1' \oplus \beta_1' = \alpha_2 \oplus \beta_2 \oplus \alpha_2' \oplus \beta_2'$ (pour le voir il suffit de poser $\alpha_1 = x \oplus h$, $\alpha_2 = x$, $\alpha_1' = x' \oplus h'$, $\alpha_2' = x'$). C'est donc sur la probabilité de trouver 2 couples $(\alpha_1, \alpha_2) \neq (\alpha_1, \alpha_2)$ que nous allons nous concentrer.

On se place à la q^{e} requête, on suppose que l'adversaire a fait q_1 requêtes à π_1 ou à π_1^{-1} et q_2 requêtes à π_2 ou à π_2^{-1}, où $q = q_1 + q_2$, et enfin on suppose qu'il n'a pas encore trouvé de collision. Soient (α_1^i, β_1^i) les q_1 couples distincts tels que $\beta_1^i = \pi_1(\alpha_1^i)$ générés par les requêtes à π_1 ou à π_1^{-1}. De même, soient (α_2^j, β_2^j) les q_2 couples distincts tels que $\beta_2^j = \pi_2(\alpha_2^j)$ générés par les requêtes à π_2 ou à π_2^{-1}. On peut supposer sans perte de généralité que l'adversaire est sur le point de faire une requête à π_1 ou à π_1^{-1}, et supposons même qu'il est sur le point de faire une

requêtes α à π_1. Soient $\Delta_{i,j,k} = \alpha_1^i \oplus \beta_1^i \oplus \alpha_2^j \oplus \beta_2^j \oplus \alpha_2^k \oplus \beta_2^k \oplus \alpha$, puisqu'il n'y a pas encore de collision, il y a $q_1 q_2 (q_2 - 1)/2$ $\Delta_{i,j,k}$ distincts. Puisque $q_1 + q_2 = q$ et que la fonction $x \mapsto x(q-x)$ est majorée par $q^2/4$, on a $q_1 q_2 \leq q^2/4$. Par ailleurs, puisque $2^n - q \geq 2^{n-1}$, la probabilité qu'il existe un triplet (i, j, k) tel que $\beta = \pi_1(\alpha) = \Delta_{i,j,k}$ est inférieure à

$$\frac{q_1 q_2 (q_2 - 1)}{2(2^n - q_2)} \leq \frac{q^3}{4 \cdot 2^n}.$$

Par conséquent, à la q^{e} itération, la probabilité de succès est plus petite que $q^3/(4 \cdot 2^n)$ et à la fin de l'attaque la probabilité de succès est plus petite que $\sum_{q=1}^{Q} q^3/(4 \cdot 2^n) \leq Q^4/(4 \cdot 2^n)$. □

Théorème 7.2. *Soit $\mathcal{A}$ un adversaire en collision contre H calculatoirement non borné, qui fait au plus Q requêtes. Son avantage est majoré par :*

$$\mathsf{adv}_H^{\mathsf{Coll}}(\mathcal{A}) \leq \frac{Q^4}{2^n}.$$

Démonstration. Pour trouver une collision contre toute la fonction de hachage il faut soit trouver une collision sur la fonction de compression, soit trouver un antécédent à IV. Comme cela est prouvé à la section suivante, l'adversaire a une probabilité inférieure à $Q^2/2^n$ de trouver un antécédent à IV. D'après le théorème précédent, il a une probabilité inférieure à $Q^4/(4 \cdot 2^n)$ de trouver une collision sur f. En somme, sa probabilité de trouver une collision sur H est bien inférieure à $Q^4/2^n$. □

7.4.2 Attaques

Attaques en $\mathcal{O}(2^{n/2})$ requêtes

Nous présentons maintenant deux attaques en collision utilisant le paradoxe des anniversaires. Ces attaques génèrent deux messages de même taille et sont donc valables même quand on ajoute le renforcement de Merkle-Dåmgard. La première fait $\mathcal{O}(2^{n/2})$ requêtes à chacune des permutations, quand la seconde réduit le nombre de requêtes faites à la première permutation, elle n'en fait que n, au prix d'une augmentation du nombre de requêtes faites à la seconde permutation, elle en fait $\mathcal{O}(n2^{n/2})$. Cette dernière attaque est analogue à une attaque faite par Joux [Jou04]. Les attaques donnent une majoration de la résistance aux collisions du schéma quand on ajoute le renforcement de Merkle-Dåmgard et permettent d'énoncer le résultat suivant.

Résultat 7.1. *Pour $Q \geq 2^{n/2}$, il existe un adversaire en collision avec probabilité de succès plus grande que* $1/2$.

Plus précisément, nous allons prouver le théorème suivant en exhibant un attaquant. Puisque $\mathsf{adv}_H^{\mathtt{Coll}}(Q)$ est croissant avec Q, cela prouve le résultat.

Théorème 7.3. *Pour tout $Q \leq 2^{n/2}$,*

$$\mathsf{adv}_H^{\mathtt{Coll}}(Q) \geq \frac{Q(Q-1)}{4 \cdot 2^n}.$$

Démonstration. Soit $Q \leq 2^{n/2}$. Soit $\mathcal{A}$ l'adversaire suivant. Il choisit au hasard Q blocs de n bits $(a_i)_{1 \leq i \leq Q}$ et calcule $b_i = \pi_2(a_i \oplus h_0) \oplus \pi_1(a_i)$ en faisant Q requêtes à π_1 et π_2. S'il y a (i_0, j_0) tel que $i_0 \neq j_0$ et $b_{i_0} = b_{j_0}$ alors les messages a_{i_0} et a_{j_0} induisent une collision pour H. Nous allons minorer la probabilité que cela se produise.

Puisque $\pi_2'(\cdot) = \pi_2(\cdot \oplus h_0)$ est une permutation aléatoire, $f(IV, \cdot)$ est la somme de deux permutations aléatoires. Lucks [Luc00] a montré que pour tout adversaire B qui fait au plus Q requêtes, son avantage de distinguer le XOR de deux permutations aléatoires, d'une fonction parfaitement aléatoire est majoré par $Q^3/2^{2n-1}$. Soit $\mathcal{B}$ un adversaire en prf contre $f(IV, \cdot)$ qui procède de façon très similaire à $\mathcal{A}$: il choisit au hasard Q blocs de n bits $(a_i)_{1 \leq i \leq Q}$, les envoie à l'oracle et reçoit en retour les b_i. S'il y a (i_0, j_0) tel que $i_0 \neq j_0$ et $b_{i_0} = b_{j_0}$ alors il retourne 1. L'avantage en prf de $\mathcal{B}$ est :

$$\mathsf{adv}^{\mathtt{prf}}_{f(IV,\cdot)}(\mathcal{B}) = \left| \Pr[\mathcal{B}^{f(IV,\cdot)} \text{ trouve une collision}] - \Pr[\mathcal{B}^{O} \text{ trouve une collision}] \right|.$$

où O est une fonction aléatoire de $\{0,1\}^n$ dans $\{0,1\}^n$. La borne inférieure du paradoxe des anniversaires nous dit que $\Pr[\mathcal{B}^O \text{ trouve une collision}] \geq Q(Q-1)/2^{n+1}$. On remarquera que la probabilité que $\mathcal{A}$ trouve une collision est égale à la probabilité que $\mathcal{B}$ trouve une collision quand il fait face à $f(IV, \cdot)$. Par conséquent, on a

$$\begin{aligned} \Pr[\mathcal{A} \text{ trouve une collision}] &\geq \frac{Q(Q-1)}{2^{n+1}} - \frac{Q^3}{(2^{2n}-1)} \\ &\geq \frac{Q(Q-1)}{4 \cdot 2^n}. \end{aligned}$$

Cela prouve le résultat attendu. □

Nous venons de voir que $\mathcal{O}(2^{n/2})$ appels à π_1, et $\mathcal{O}(2^{n/2})$ appels à π_2 sont suffisants pour trouver une collision avec forte probabilité. Nous montrons maintenant comment diminuer fortement le nombre d'appels faits à π_2 à $\mathcal{O}(n)$, en augmentant en contrepartie le nombre d'appels faits à π_1 jusqu'à $\mathcal{O}(n2^{n/2})$. Cette attaque s'appuie sur une méthode présentée par Joux [Jou04].

Nous définissons d'abord deux suites aléatoires de $n/2$ éléments de n bits, $u_1^0, \ldots, u_{n/2}^0$ et $u_1^1, \ldots, u_{n/2}^1$ et générons les deux suites de $n/2$ éléments de n bits $v_1^0, \ldots, v_{n/2}^0$ et $v_1^1, \ldots, v_{n/2}^1$ telles que pour tout i, $1 \leq i \leq n/2$, $u_i^0 \neq u_i^1$ et $v_i^\sigma = \pi_2(u_i^\sigma)$, pour $\sigma \in \{0,1\}$. On crée ces suites en faisant n requêtes à π_2.

Ensuite, pour toute chaîne de bits $(\sigma_1, \ldots \sigma_{n/2}) \in \{0,1\}^{n/2}$, on calcule $H(u_1^{\sigma_1} \| \ldots \| u_{n/2}^{\sigma_{n/2}})$ en effectuant les calculs suivants : $h_0 = IV$ et $h_i = \pi_1(h_{i-1} \oplus u_i^{\sigma_i}) \oplus h_{i-1} \oplus v_i^{\sigma_i}$ pour tout $1 \leq i \leq n/2$ et $H(u_1^{\sigma_1} \| \ldots \| u_{n/2}^{\sigma_{n/2}}) = h_{n/2}$. Cela nécessite $n/2$ requêtes à π_1 pour chaque chaîne de bits $(\sigma_1, \ldots, \sigma_{n/2})$ et donc $n/2 \cdot 2^{n/2}$ requêtes à π_1 en tout (mais aucune requête à π_2). Si on applique le paradoxe des anniversaires à l'ensemble des $H(u_1^{\sigma_1} \| \ldots \| u_{n/2}^{\sigma_{n/2}})$, on a donc une collision avec forte probabilité en $\mathcal{O}(n \cdot 2^{n/2})$ appels à π_1 et $\mathcal{O}(n)$ à π_2.

7.4.3 Attaques en $\mathcal{O}(2^{3n/8})$ requêtes

On présente maintenant une attaque qui réussit avec probabilité presque 1 et qui nécessite $2 \cdot 2^{3n/8}$ requêtes. Cette attaque est meilleure que les attaques précédentes puisqu'elle a une complexité en requêtes de seulement $2 \cdot 2^{3n/8}$, mais elle a une complexité temporelle de $\mathcal{O}(n2^{3n/4})$. Par ailleurs, on ne contrôle pas la taille des deux messages générés durant l'attaque et très probablement ils n'auront pas la même taille, par conséquent, le renforcement de Merkle-Damgård permet d'éviter l'attaque. Notre analyse de cette attaque est entièrement heuristique. Cependant, nous avons testé l'attaque pour $n = 36$ et $n = 40$ et les fréquences de succès confirment l'analyse heuristique (voir le tableau de la figure 7.2 pour un résumé des résultats).

Proposition 7.1. *Pour $Q \geq 2^{3n/8+2}$, en supposant que l'analyse heuristique est vraie, il existe un adversaire en collision avec une bonne probabilité de succès.*

Algorithme 3 Attaque(IV, π_1, π_2)

Choisir α_0 au hasard, $\beta_0 \leftarrow IV \oplus \alpha_0$
$T \leftarrow \{IV\}$, $column \leftarrow \{IV\}$,
pour $(i \leftarrow 1, q)$ **faire**
    $\alpha_i \leftarrow \pi_1(\alpha_{i-1}) \oplus \alpha_{i-1}$, $\beta_i \leftarrow \pi_1(\beta_{i-1}) \oplus \beta_{i-1}$, $\Delta_{i,i} \leftarrow \alpha_i \oplus \beta_i$
    $column \leftarrow column \cup \{\Delta_{i,i}\}$, $T \leftarrow T \cup \{\Delta_{i,i}\}$
fin de pour
pour $(i \leftarrow 1, q)$ **faire**
    pour $(j \leftarrow 1, q)$ **faire**
        Trouver $\Delta_{k,k}$ dans *column* tel que $\Delta_{k,k} = \alpha_i \oplus \beta_j$
        si (il existe $\Delta_{k,k}$) **alors**
            pour $(\ell \leftarrow 1, \min(q-i, q-j))$ **faire**
                si ($\alpha_{i+\ell} \oplus \beta_{j+\ell}$ appartient à T) **alors** $collision \leftarrow \alpha_{i+\ell} \oplus \beta_{j+\ell}$ et sortir de la boucle
                sinon $T \leftarrow T \cup \{\alpha_{i+\ell} \oplus \beta_{j+\ell}\}$
                fin de si
            fin de pour
        fin de si
    fin de pour
fin de pour
Avec une recherche en profondeur d'abord, trouver les 2 messages x' et x qui mènent à *collision* dans T
renvoyer x et x'

Description de l'attaque. Dans la suite, tout d'abord nous expliquons comment nous faisons les requêtes, ensuite nous évaluons informellement l'espérance du nombre de messages dont nous sommes capables de calculer le haché. Pour un algorithme précis, voir l'algorithme 3. Le principe de l'attaque consiste à introduire l'arbre T de tous les messages pour lesquels on arrive à calculer le haché et, par une astuce, d'essayer de le faire grossir autant que possible, de telle sorte qu'il contienne assez vite $2^{n/2}$ nœuds. On espère alors que le paradoxe des anniversaires s'applique et permet de trouver une collision.

Soient α_0 et β_0 deux chaînes de n bits choisies aléatoirement dans $\{0,1\}^n$ de telle sorte que $\alpha_0 \oplus \beta_0 = IV$. Soit q un entier. On génère les suites $(\alpha_i)_{0\leq i\leq q}$ et $(\beta_i)_{0\leq i\leq q}$ telles que pour tout $1 \leq i \leq q$, $\alpha_i = \pi_1(\alpha_{i-1}) \oplus \alpha_{i-1}$, et $\beta_i = \pi_2(\beta_{i-1}) \oplus \beta_{i-1}$. Pour tout $0 \leq i, j \leq q$, on définit $\Delta_{i,j} = \alpha_i \oplus \beta_j$. On fait $Q = 2q$ requêtes à π_1 et π_2 et on génère approximativement $(q+1)^2$ $\Delta_{i,j}$ différents. On génère de cette manière les suites, car on a ainsi la propriété intéressante suivante : pour tout (i,j), $f(\Delta_{i,j}, \alpha_i) = \Delta_{i+1,j+1}$. Par conséquent, pour tout $1 \leq \ell \leq q$ le haché du message $\alpha_0\|\alpha_1\|\ldots\|\alpha_{\ell-1}$ est $\Delta_{\ell,\ell}$ et si $\Delta_{k,k} = \Delta_{i,j}$, pour tout $1 \leq \ell \leq q - \max(i,j)$, $1 \leq \ell \leq q - \max(i,j)$, le haché du message $M_{k,i,j,\ell} = \alpha_0\|\alpha_1\|\ldots\|\alpha_{k-1}\|\alpha_i\|\alpha_{i+1}\|\ldots\|\alpha_{i+\ell-1}$ est $\Delta_{i+\ell,j+\ell}$. Un tel triplet (k,i,j) est appelé *triplet de collision* et le message $M_{k,i,j,\ell}$ est une préimage de $\Delta_{i+\ell,j+\ell}$ pour H.

S'il y a beaucoup de triplets de collision (i,j,k) différents, alors on est capable de trouver une préimage pour beaucoup de valeurs $\Delta_{i+\ell,j+\ell}$. Introduisons l'arbre $T = (V, E)$ où

$$\begin{aligned} V &= \{\Delta_{a,a}, 0 \leq a \leq q\} \cup \{\Delta_{a,b} | \exists \text{ triplet de collision } (k,i,j) \text{ } t.q. \text{ } a - i = b - j \geq 0\} \\ E &= \{(\Delta_{a,b}, \Delta_{a+1,b+1}) \text{ } t.q. \text{ } 0 \leq a, b \leq q-1, \Delta_{a,b} \in T\}. \end{aligned}$$

On remarque que nous sommes capables de trouver une préimage pour tout $\Delta_{a,b} \in V$ et que T est un arbre si et seulement si il n'y a pas de collision (sinon nous serions capable de

n	Q	nombre d'expériences	taille de T	pourcentage de succès
36	2^{15}	10000	$2^{18} \leq \cdot \leq 2^{19}$	52%
40	$2^{16.5}$	1000	$2^{20} \leq \cdot \leq 2^{21}$	59%

FIG. 7.2 – Résultats expérimentaux

trouver un cycle dans T et il y aurait deux manières d'atteindre un certain $\Delta_{a,b} \in V$). Par conséquent, notre but est de faire grandir V jusqu'à ce qu'il ait environ $2^{n/2}$ sommets de telle sorte qu'une collision ait lieu. Dans la suite nous expliquons informellement pourquoi ceci se produit avec forte probabilité pour $q = 2^{3n/8+1}$. Cette analyse considère que les α_i et les β_j sont uniformément distribués, ce qui n'est bien sûr pas le cas. Cependant, nous espérons que cette analyse donne une bonne intuition de ce qui se passe.

Soit $t = |T|$ la taille de T, nous allons évaluer approximativement l'espérance de t. On remarque d'abord que T contient au moins les $q+1$ sommets $\Delta_{a,a}$ pour $0 \leq a \leq q$. Si (i,j,k) est un triplet de collision tel que $i \neq j$, alors tous les $\Delta_{i+\ell,j+\ell}$, avec $0 \leq \ell \leq q - \max(i,j)$, sont ajoutés à T. Par conséquent, si $\mathcal{S}et$ désigne $\{(i,j,k)\ t.q.\ i \neq j, i \neq k, j \neq k\}$, on a :

$$t \approx 1 + q + \sum_{(i,j,k) \in Set} \mathbb{1}_{\{\Delta_{k,k} = \Delta_{i,j}\}}(q - \max(i,j)),$$

où $\mathbb{1}$ désigne la fonction caractéristique. On a donc

$$\mathbb{E}(t) \approx 1 + q + \sum_{(i,j,k) \in Set} \Pr\left[\Delta_{k,k} = \Delta_{i,j}\right](q - \max(i,j))$$

Si les α_i et les β_j étaient uniformément distribués (ce qui *n'*est *pas* le cas) alors on aurait $\Pr\left[\Delta_{k,k} = \Delta_{i,j}\right] = 1/2^n$ et par conséquent on aurait

$$\mathbb{E}(t) \approx 1 + q + \frac{1}{2^n} \sum_{(i,j,k) \in Set} (q - \max(i,j)) = 1 + q + \frac{q(q+1)(q-1)(q-2)}{3 \cdot 2^n} \approx \frac{q^4}{3 \cdot 2^n}.$$

Nous pouvons conclure que pour $q = 2^{3n/8}$, nous pouvons espérer que T contienne plus de $2^{n/2}$ nœuds et, dans ce cas, espérer que le paradoxe des anniversaires s'applique de telle sorte que deux de ces nœuds induisent une collision. Comme on l'a déjà dit, si on trouve une collision alors l'attaque est finie, nous sommes capables de trouver deux messages qui provoquent une collision pour H. □

COMPLEXITÉ DE L'ATTAQUE ET RÉSULTATS EXPÉRIMENTAUX. L'algorithme précis est décrit dans l'algorithme 3. L'attaque nécessite $\mathcal{O}(2^{3n/8})$ requêtes à π_1 et π_2, $\mathcal{O}(2^{n/2})$ en espace (pour sauvegarder T) et $\mathcal{O}(n2^{3n/4})$ en temps (car on doit chercher *tous* les triplets de collision (i,j,k) avec $1 \leq i,j,k \leq 2^{3n/8}$, c'est-à-dire tous les triplets (i,j,k) tels que $\Delta_{k,k} = \alpha_i \oplus \beta_j$).

Nous avons fait des tests pour n égal à 36 et 40. Pour ça, nous avons utilisé l'algorithme de chiffrement par bloc RC5 avec deux clefs aléatoires et un IV aléatoire. Les résultats sont résumés dans la figure 7.2. Il apparaît que pour $Q = 2\sqrt{2} \cdot 2^{3n/8}$, dans *toutes* les expériences l'arbre T contient entre $2^{n/2}$ et $2^{n/2+1}$ nœuds et une collision est trouvée dans au moins la moitié des cas. Cela valide notre analyse heuristique de l'attaque.

REMARQUES. Nous avons étudié d'autres constructions de fonction de compression utilisant deux permutations et uniquement des xor. Certaines induisent des fonctions de hachage qui sont trivialement cassables, pour toutes les autres une variante de cette attaque peut être appliquée (parfois il faut adapter astucieusement l'attaque, en faisant des requêtes à π_1^{-1} ou π_2^{-1}).

7.5 Sécurité en préimage contre la construction généralisée

7.5.1 Preuve de sécurité

Dans cette section nous prouvons que la résistance en préimage de notre construction est prouvablement en $\mathcal{O}(2^{n/2})$ requêtes. On remarquera que dans le schéma de la fonction de compression, nous avons mis ce que l'on appelle habituellement un *feed-forward*, c'est-à-dire que la valeur de chaînage h en entrée est xorée à la sortie des deux permutations. Ôter ce dernier n'altère pas la résistance aux collisions, il a été mis uniquement pour éviter des attaques triviales en préimage contre la fonction de compression, attaques qui induieraient des attaques en préimage contre la fonction de hachage globale plus efficaces que $\mathcal{O}(2^{n/2})$. Grâce à ce *feed-forward*, nous pouvons prouver que la fonction de compression est résistante en préimage tant que $\mathcal{O}(2^{n/2})$ requêtes ont été faites, ce qui implique que la fonction de hachage entière l'est aussi puisque trouver une préimage pour la fonction de hachage entière nécessite de trouver une préimage pour la fonction de compression.

Proposition 7.2. *Soit $\mathcal{A}$ un adversaire calculatoirement non borné qui fait au plus Q requêtes. La probabilité qu'il réussisse à casser la résistance en préimage de f est majorée par $Q^2/2^n$.*

Démonstration. Soient α et α' deux requêtes faites respectivement à π_1 et π_2 et soient β et β' les réponses respectives (c'est-à-dire que l'on a $\beta = \pi_1(\alpha)$ et $\beta' = \pi_2(\alpha')$). Soit x la valeur dont on cherche une préimage. La probabilité $\Pr[x = \alpha \oplus \beta \oplus \alpha' \oplus \beta']$ est égale à $\sum_y \Pr[\beta = x \oplus \alpha \oplus \alpha' \oplus y] \Pr[\beta' = y]$ car β et β' sont indépendantes. Comme α, α' sont fixées par l'adversaire, cette dernière somme est égale à $1/2^n$. Le résultat est le même si c'est à π_1^{-1} que la requête β est faite ou si c'est à π_2^{-1} que la requête β' est faite. Par conséquent, si $\mathcal{A}$ fait q_1 requêtes à π_1 ou à π_1^{-1} et s'il fait q_2 requêtes à π_2 ou à π_2^{-1} (de telle sorte que $q_1 + q_2 = Q$) et s'il obtient les couples (α_i, β_i) et (α'_j, β'_j) respectivement, alors la probabilité qu'il existe un couple (i, j) tel que $x = \alpha_i \oplus \beta_i \oplus \alpha'_j \oplus \beta'_j$ est majorée par $q_1 q_2/2^n$ et donc par $Q^2/2^n$. □

7.5.2 Optimalité de la preuve et attaques

Dans [RS08b], Rogaway et Steinberger présentent une attaque en préimage génerique en $\mathcal{O}(n2^{n/2})$ contre toute fonction de hachage basée sur 2 permutations. Cela signifie, comme on l'a déjà mentionné, que notre construction atteint le meilleur niveau de sécurité contre la préimage que l'on peut espérer, à savoir $\mathcal{O}(2^{n/2})$ requêtes. En ce sens la construction est optimale.

Par ailleurs, l'attaque de [RS08b] contre la fonction de hachage en entier nécessite $\mathcal{O}(n2^{n/2})$ requêtes, mais la sécurité pratique exacte n'est pas établie en général. Cependant, dans notre cas, la sécurité pratique semble plus grande que 2^n. Ceci nous amène à nous demander quelle est l'attaque avec la plus petite complexité temporelle. Puisque trouver une préimage pour la fonction de compression nécessite $\mathcal{O}(n2^{n/2})$ en temps, l'attaque de Lai et Massey [LM92] peut être utilisée. Cette attaque est une attaque par le milieu non équilibrée : on calcule $2^{n/4}$ préimages sur la fonction de compression, on hache $2^{3n/4}$ messages et le paradoxe des anniversaires nous dit qu'avec forte probabilité il existe une préimage et un haché qui sont égaux, ce qui nous permet donc de trouver une préimage sur la fonction de hachage en entier. Calculer les préimages nécessite de faire $2^{n/4} \cdot 2^{n/2} = 2^{3n/4}$ requêtes, demande $\mathcal{O}(2^{3n/4})$ étapes de calcul et $\mathcal{O}(2^{n/4})$ en mémoire. Le calcul des hachés nécessite $2^{3n/4}$ requêtes, $\mathcal{O}(2^{3n/4})$ étapes de calcul et $\mathcal{O}(2^{3n/4})$ en mémoire. Trouver une collision entre les deux listes nécessite $\mathcal{O}(n2^{3n/4})$ en temps. Globalement, trouver une préimage nécessite donc $\mathcal{O}(2^{3n/4})$ requêtes, $\mathcal{O}(n2^{3n/4})$ en temps et $\mathcal{O}(2^{3n/4})$ en espace. La

complexité temporelle est supérieure à $\mathcal{O}(n2^{n/2})$ et trouver une attaque en préimage contre toute la fonction de hachage de complexité temporelle moins que $\mathcal{O}(2^{3n/4})$ est un problème ouvert.

7.6 Conclusion

Dans ce chapitre, nous avons présenté une attaque en collision contre SMASH avec une complexité temporelle en $\mathcal{O}(2^{n/3})$. Cette attaque s'applique non seulement à la version originelle de SMASH, qui avait déjà été cassée, mais également aux versions améliorées de SMASH qui elles n'avaient pas encore été cassées. Pour éviter l'attaque, nous proposons de remplacer la multiplication par θ par un algorithme de chiffrement par bloc dont la clef est fixée.

Nous prouvons que cette construction est résistante en collision jusqu'à au moins $\Omega(2^{n/4})$ requêtes. Sans le renforcement de Merkle-Damgård, la résistance aux collisions peut être cassée avec $\mathcal{O}(2^{3n/8})$ requêtes, alors que la meilleure attaque connue contre ce schéma quand le renforcement de Merkle-Damgård est utilisé est l'attaque par paradoxe des anniversaires qui nécessite $\mathcal{O}(2^{n/2})$ requêtes.

Enfin, nous prouvons que la résistance en préimage de ce schéma est en $\mathcal{O}(2^{n/2})$ requêtes, ce qui est le meilleur niveau de sécurité que l'on peut espérer obtenir, comme cela est prouvé dans [RS08b]. En ce sens, notre construction est optimale et répond partiellement à un problème ouvert : trouver une construction basée sur 2 permutations qui atteint les niveaux de sécurité trouvés dans [RS08b].

Notre travail ouvre également de nouveaux problèmes. Tout d'abord, réduire l'intervalle entre la borne de sécurité établie dans la preuve, à savoir $\Omega(2^{n/4})$, et la meilleure attaque en collision connue, à savoir $\mathcal{O}(2^{3n/8})$ sans le renforcement de Merkle-Damgård et $\mathcal{O}(2^{n/2})$ avec ce dernier. Le second problème ouvert consiste à trouver une fonction de compression basée sur 2 permutations pour laquelle la résistance en collision de la fonction de hachage entière est prouvablement optimale.

8

Attaque en seconde preimage contre des fonctions de hachage avec tramage

Sommaire

8.1 Introduction

Parallèlement aux récentes attaques en collision contre les fonctions de hachage les plus répandues, sont apparues des attaques contre le mode opératoire le plus célèbre pour construire

des fonctions de hachage : le mode cascade. Joux [Jou04] a ouvert la voie en 2004 en montrant que l'on pouvait construire des k-collisions avec le mode cascade en seulement $\mathcal{O}(k2^{n/2})$ requêtes à la fonction de compression. Cette attaque a encouragé les études du mode opératoire à la recherche d'autres faiblesses particulières. Kelsey et Schneier [KS05] puis Kelsey et Kohno [KK06] se sont demandés ce que l'on pouvait faire contre le mode cascade, à partir du moment où l'on autorisait l'adversaire à faire plus de $2^{n/2}$ requêtes à la fonction de compression. Kelsey et Schneier [KS05] ont montré que l'on pouvait trouver des secondes préimages, contre des messages longs, en nettement moins que $\mathcal{O}(2^n)$ requêtes à la fonction de compression.

Afin d'éviter cette attaque, de nouvelles variantes du mode cascade [BD06, Riv05] ont alors été proposées. Certaines consistent à ajouter une troisième entrée, appelée tramage, à la fonction de compression, et ce afin d'empêcher la construction des messages extensibles qui sont à l'origine de l'attaque de Kelsey et Schneier. Le tramage est un procédé venant des techniques de traitement d'image, il consiste à ajouter une perturbation pour améliorer la qualité de l'image obtenue. Ici on l'ajoute dans l'espoir d'améliorer la résistance aux secondes préimages. Le mode Haifa de Biham et Dunkelman [BD06] inclut tout simplement le numéro du bloc de message. En d'autres termes, le tramage est un compteur, ce qui peut être encodé sur 64 bits. Rivest [Riv05] propose une solution qui utilise pour tramage une suite sans carré qui peut être codée sur seulement 2 ou 16 bits.

Notre objectif dans cette partie est d'étudier la sécurité en seconde préimage de ces deux modes opératoires. Pour cela, nous développons une nouvelle attaque en seconde préimage qui s'applique contre le mode cascade, mais aussi contre le nouveau mode proposé par Rivest. Cette attaque est même plus large, puisqu'on peut l'adapter pour attaquer une construction de famille de fonctions de hachage universelle à sens unique proposée par Shoup. Enfin, si on peut encore attaquer la construction de Rivest, il n'en est pas de même de Haifa. Au contraire, nous prouvons qu'il n'existe pas d'attaque générique contre le mode opératoire Haifa. Tous ces résultats ont donné lieu à une publication

8.2 Modèle de sécurité

Dans ce chapitre, nous sommes amenés à considérer des constructions de fonction de hachage itérée H comme le mode cascade décrit en introduction dans la sous-section 2.5.4. Nous nous plaçons dans le cadre du modèle de l'oracle aléatoire : la fonction de compression h est modélisée par un oracle aléatoire. Dans ce modèle, l'adversaire a accès à h et peut lui poser au plus Q requêtes : à chaque requête (y, x), il renvoie y' tel que $y' = h(y, x)$.

Dans la suite, nous nous intéressons principalement à la résistance en seconde préimage. Cette notion est décrite en introduction à la sous-section 2.5.2, mais nous la rappelons succinctement. Le challenger choisit un message M et l'envoie à l'adversaire, qui renvoie un message M'. L'adversaire gagne si $H(M) = H(M')$ (les fonctions de hachage que nous étudions dans ce chapitre sont paramétrées par l'oracle aléatoire qui modélise la fonction de compression). La probabilité que l'adversaire réussisse à casser la résistance aux secondes préimages de H est notée $\mathsf{adv}^{\mathsf{sec}}_H(\mathcal{A})$ et $\mathsf{adv}^{\mathsf{sec}}_H(Q)$ désigne le maximum des probabilités de succès $\mathsf{adv}^{\mathsf{sec}}_H(\mathcal{A})$, où ce maximum est pris sur tous les adversaires $\mathcal{A}$ qui ont la même complexité temporelle et font au plus Q requêtes.

Nous supposons dans ce chapitre que l'adversaire ne pose jamais deux fois la même requête. Par ailleurs, nous supposons que quand un adversaire renvoie M' à la fin du jeu, il a déjà calculé $H(M)$ et $H(M')$, c'est-à-dire qu'il a déjà fait toutes les requêtes nécessaires pour calculer $H(M)$ et $H(M')$.

Nous utilisons les mêmes notations que celles introduites à la sous-section 2.5.4. Dans ce

chapitre, nous sommes amenés à considérer la taille en nombre de blocs d'un message M, notée $|M|_b = |M|/b$ où b est le nombre de bits dans un bloc et $|M|$ la taille de M en bits. Par ailleurs, dans ce chapitre, k est un entier tel que les messages pouvant être hachés font au plus $2^k - 1$ blocs de long. Les messages dont on veut trouver une seconde préimage seront choisis parfaitement au hasard dans $\{0,1\}^{(2^k-1)\cdot b}$.

8.3 Attaque connue en seconde préimage contre le mode cascade

Avant de présenter nos attaques en seconde préimage, nous allons présenter la meilleure attaque connue contre le mode cascade. Celle-ci a été présentée par Kelsey et Schneier [KS05]. Elle s'appuie sur la notion de message extensible et vise surtout à trouver une seconde préimage pour de longs messages.

8.3.1 Construction de message extensible

Un message extensible $\mathcal{M}$ est une famille de messages de taille différente mais ayant tous la même valeur de hachage intermédiaire avant le traitement du renforcement de Merkle-Damgård, c'est-à-dire qu'il existe une *valeur cible* y_T telle que pour tout $M \in \mathcal{M}$, $h^*(IV, M) = y_T$. Chacun de ses messages est appelé une instance de $\mathcal{M}$. La *portée* de $\mathcal{M}$ est l'ensemble des longueurs (en nombre de blocs) de ses instances, on la note $portee(\mathcal{M}) = \{|M|_b,\ M \in \mathcal{M}\}$.

Pour construire un message extensible, Kelsey et Schneier présentent dans leur article [KS05] une méthode générique qui nécessite $\mathcal{O}(k \cdot 2^{n/2} + 2^k)$ requêtes à la fonction de compression et qui crée un message extensible dont la portée est égale à $[\![k,\ 2^k + k - 1]\!]$, où k est un paramètre à fixer. Leur méthode s'appuie en partie sur une attaque présentée par Joux [Jou04].

Voici comment elle fonctionne. Soit P un bloc de message quelconque et $y_0 = IV$ la valeur initiale. Il s'agit de créer k couples de messages qui collisionnent, et ce en k étapes. À la i^{e} étape on crée un message M_i^1 de taille $2^{i-1} + 1$ et un message M_i^0 de taille 1 tels que $h^*(y_{i-1}, M_i^1) = h^*(y_{i-1}, M_i^0) = y_i$. Voici comment on trouve ces deux messages. Soit $y_{tmp} = h^*(y_{i-1}, P^{2^{i-1}})$, où $P^{2^{i-1}}$ désigne P concaténé 2^{i-1} fois. On choisit $2^{n/2}$ blocs de message m_j différents. On génère alors la liste des $h(y_{tmp}, m_j)$ et celle des $h(y_{i-1}, m_j)$ et on cherche une collision entre les deux listes, c'est-à-dire un couple (u, v) tel que $h(y_{i-1}, m_u) = h(y_{tmp}, m_v)$. On pose alors $y_i = h(y_{i-1}, m_u)$, $M_i^0 = m_u$ et $M_i^1 = P^{2^{i-1}} \| m_v$ et on réitère l'opération au rang $i + 1$. Le message extensible est :

$$\mathcal{M} = \left\{ M_1^{\sigma_0} \| \dots \| M_k^{\sigma_{k-1}} \ \ t.q.\ (\sigma_j)_{0 \le j \le k-1} \in \{0,1\}^k \right\}.$$

Pour obtenir l'instance de taille r de $\mathcal{M}$, avec $k \le r \le 2^k - 1 + k$, on définit $\sigma_{k-1} \dots \sigma_0$ l'écriture en binaire sur k bits du nombre $r - k \in [\![0,\ 2^k - 1]\!]$ (où σ_0 est le bit de poids faible et σ_{k-1} est le bit de poids fort), et on génère l'instance $M_1^{\sigma_0} \| \dots \| M_k^{\sigma_{k-1}}$.

La génération d'un couple de messages à la i^{e} étape nécessite $\mathcal{O}(2 \cdot 2^{n/2} + 2^{i-1})$ requêtes, pour i variant entre 1 et k. La complexité globale pour générer $\mathcal{M}$ est donc bien égale à $\mathcal{O}(k \cdot 2^{n/2} + 2^k)$.

8.3.2 Attaque de Kelsey et Schneier

La contruction de message extensible présentée à la sous-section précédente permet de monter une attaque en seconde préimage contre le mode cascade. Soit M le message envoyé par le challenger, on suppose que $|M|_b = 2^k - 1$. Si $M = x_1 \dots x_{2^k-1}$, on note $y_0 = IV$ et $y_i = h(y_{i-1}, x_i)$ pour $1 \le i \le 2^k - 1$. Dans une première étape, l'attaquant génère un message extensible $\mathcal{M}$ de

portée $[\![k, 2^k + k - 1]\!]$. Soit y_T la valeur cible de $\mathcal{M}$. Dans un deuxième temps, on choisit un ensemble de 2^{n-k} blocs de message m_j distincts et on calcule les $h(y_T, m_j)$. Avec forte probabilité il existe un couple (u, v) tel que $h(y_T, m_u) = y_v$, avec $v \geq k + 1$. On dit que l'on connecte le message extensible à une des valeurs de chaînage intermédiaires de M. On génère alors l'instance de $\mathcal{M}$ de taille $v - 1$, que l'on note N. Le message $M' = N\|m_u\|x_{v+1}\| \dots \|x_{2^k-1}$ est une seconde préimage de M (puisque $h^*(IV, M) = h^*(IV, M')$ et $|M|_b = |M'|_b$, on a bien $H(M) = H(M')$ malgré le renforcement de Merkle-Damgård) et l'attaque est donc couronnée de succès.

Cette attaque nécessite au plus $\mathcal{O}(k \cdot 2^{n/2} + 2^k)$ requêtes pour générer le message extensible, puis $\mathcal{O}(2^{n-k})$ requêtes pour raccorder le message extensible aux valeurs intermédiaires de chaînage de M. La complexité globale est en $\mathcal{O}(k \cdot 2^{n/2} + 2^k + 2^{n-k})$. Lorsque $n \geq 2k$, le terme prédominant de cette complexité est le terme 2^{n-k}. C'est le cas pour SHA–1 car $n = 160$ et les messages font au plus $2^k = 2^{55}$ blocs de long. Pour SHA–1, la complexité de l'attaque de Kelsey et Schneier est donc d'environ 2^{105}.

8.3.3 Preuve d'optimalité

Nous allons maintenant montrer que dans le cas où $n \geq 2k$, cette attaque est optimale, c'est-à-dire qu'il n'existe pas d'attaque qui soit asymptotiquement plus efficace.

Théorème 8.1 (Sécurité en seconde préimage du mode cascade)**.** *Soit H une fonction de hachage itérée suivant le mode cascade et $\mathcal{A}$ un adversaire en seconde préimage contre H qui fait au plus Q requêtes à h. Soit $2^k - 1$ la taille du message généré par le challenger. On a alors :*

$$\Pr\left[\mathcal{A} \Rightarrow M' \colon H(M) = H(M')\right] \leq \frac{2^k \cdot Q}{2^n}.$$

Démonstration. Nous simulons l'exécution d'un adversaire $\mathcal{A}$ et enregistrons toutes les requêtes faites par $\mathcal{A}$ à l'oracle aléatoire : ces requêtes forment un ensemble de triplets (y, x, y') tels que $y' = h(y, x)$. Comme expliqué dans le modèle de sécurité de la section 8.2, on suppose que $\mathcal{A}$ évalue $H(M)$, de telle sorte que $\mathcal{A}$ fait les requêtes correspondantes à l'oracle à un moment ou à un autre. On notera $M_{pad} = (x_i)_{1 \leq i \leq m}$ où $m = |M_{pad}|_b$ et $(y_{i-1}, x_i, y_i)_{1 \leq i \leq m}$ ces requêtes . En particulier, $H(M) = y_m$. Puisque h est une fonction aléatoire, les y_i générés sont parfaitement aléatoires.

Si $\mathcal{A}$ réussit à trouver une seconde préimage, alors $\mathcal{A}$ produit en particulier une collision. Or il est bien connu que pour le mode cascade avec *renforcement*, une collision sur la fonction de hachage implique une collision sur la fonction de compression [Mer89, Dam89]. Remontrons-le.

Soit M' un message tel que $H(M) = H(M')$ et soient $M'_{pad} = (x'_i)_{1 \leq i \leq m'}$ où $m' = |M'_{pad}|_b$, $y'_0 = IV$, et, pour tout $1 \leq i \leq m'$, $y'_i = h(y'_{i-1}, x'_i)$. On a donc $y_m = y'_{m'}$. Si $|M| \neq |M'|$, alors nécessairement $x_m \neq x'_{m'}$ et comme $h(y_{m-1}, x_m) = h(y'_{m'-1}, x'_{m'})$, on a trouvé une collision sur h. Au contraire si $|M| = |M'|$, on introduit i_0 tel que :

- $i_0 = m + 1$, si $x_m \neq x'_m$
- $i_0 = \min\left\{1 \leq i \leq m \text{ t.q. } \forall\, i \geq j \geq m,\ (y_{j-1}, m_j) = (y'_{j-1}, m'_j)\right\}$ sinon.

Si $i_0 = 1$ alors $M = M'$, ce qui est exclu, donc $2 \leq i_0 \leq m + 1$. Par définition de i_0, on a $(y_{i_0-2}, x_{i_0-1}) \neq (y'_{i_0-2}, x'_{i_0-1})$ et $h(y_{i_0-2}, x_{i_0-1}) = y_{i_0-1} = y'_{i_0-1} = h(y'_{i_0-2}, x'_{i_0-1})$, on a donc trouvé une collision sur h.

Ceci implique que si $\mathcal{A}$ renvoie M' et gagne, alors il existe i_1 tel que $(y_{i_1-1}, x_{i_1}) \neq (y'_{i_1-1}, x'_{i_1})$ et $h(y_{i_1-1}, x_{i_1}) = h(y'_{i_1-1}, x'_{i_1})$. Notons E cet événement et prouvons que la probabilité qu'il se produise est plus petite que $Q \cdot m/2^n = Q \cdot 2^k/2^n$.

À chaque requête (y, x) de $\mathcal{A}$ différente de tous les (y_i, x_i), la probabilité que $h(y, x)$ soit égale à une des y_i est inférieure à $m/2^n$ car il y a au plus m valeurs y_i différents. Puisque $\mathcal{A}$ envoie Q requêtes à l'oracle durant son exécution, on obtient la borne annoncée. □

Ce théorème dit qu'il faut faire au moins $\mathcal{O}(2^{n-k})$ requêtes pour obtenir une seconde préimage avec forte probabilité. Or si $n \geq 2k$, l'attaque de Kelsey et Schneier a une complexité en $\mathcal{O}(2^{n-k})$, elle est donc bien optimale.

8.4 Mode cascade avec tramage

L'idée générale du hachage tramé est de perturber le processus du hachage en ajoutant une entrée à la fonction de compression et en utilisant pour cette entrée les éléments successifs d'une suite de tramage fixée. Ceci donne à l'attaquant moins de contrôle sur l'entrée de la fonction de compression et rend le hachage d'un bloc de message dépendant de sa position dans le message global. En particulier, le but du tramage est d'empêcher les attaques basées sur des messages extensibles.

Puisque la suite de tramage $\mathbf{z}$ doit être au moins aussi longue que le nombre maximal de blocs de tout message qui peut être haché, il est raisonnable de considérer comme candidats pour $\mathbf{z}$ des suites de taille au moins $2^k - 1$. Soit $\mathcal{A}$ un alphabet fini et soit $\mathbf{z}$ la suite de tramage qui est un mot de $\mathcal{A}$ de taille finie. On désigne par $\mathbf{z}[i]$ le i^{e} élément de $\mathbf{z}$.

Le mode cascade avec tramage requiert une fonction de compression qui accepte trois entrées $h\colon \{0,1\}^n \times \{0,1\}^b \times \mathcal{A} \to \{0,1\}^n$ et est obtenu en itérant cette fonction de compression. Si le message M est tel que $M_{pad} = x_1 \cdots x_m$, alors on pose $y_0 = IV$, pour tout $1 \leq i \leq m$, $y_i = h\left(y_{i-1}, x_i, \mathbf{z}\left[i\right]\right)$ et $H(M) = y_m$. Il est possible de transformer facilement une fonction de hachage classique, telle SHA–1, en une fonction de hachage tramée. Pour cela, il faut réduire la bande passante accordée au message. Supposons que les lettres de $\mathcal{A}$ puissent toutes se coder sur t bits, et supposons que $h\colon \{0,1\}^n \times \{0,1\}^b \to \{0,1\}^n$ soit une fonction de compression classique, on pose alors $h'\colon \{0,1\}^n \times \{0,1\}^{b-t} \times \mathcal{A} \to \{0,1\}^n$ et $h'(y, x, \mathbf{z}[i]) = h(y, x \| \mathbf{z}[i])$. On ôte donc t bits à la bande passante accordée au message.

8.4.1 Mots et suites

Nous commençons par introduire les notions liées aux suites de tramage, notions qui nous serons utiles par la suite.

Notations et Terminologie

Soit ω un mot sur un alphabet fini $\mathcal{A}$. L'opérande $\cdot$ désigne la concaténation. Si ω peut être écrit comme $\omega = x.y.z$ (où x,y ou z peuvent être vides), on dit que x est un *préfixe* de ω et que y est un *facteur* (ou sous-mot) de ω. Un mot fini ω est un *carré* (respectivement un *carré abélien*) si il peut être écrit sous la forme $\omega = x \cdot x'$ où $x' = x$ (respectivement où x' est le fruit d'un réarrangement des lettres de x). Un mot est dit *sans carré* (respectivement *sans carré abélien*) si aucun de ses facteurs est un carré (resp. un carré abélien). On remarquera que des mots sans carré abélien sont aussi sans carré. Enfin, on note $\omega[i]$ la i^{e} lettre d'un mot ω.

Complexité de suite

Pour une taille donnée, le nombre de facteurs différents présents dans une suite finie donne une notion intuitive de la *complexité* de cette suite : la suite est plus complexe (ou plus riche)

si elle possède un grand nombre de facteurs différents. On désigne par $Fact_{\mathbf{z}}(\ell)$ le nombre de facteurs de taille ℓ de la suite $\mathbf{z}$:

$$Fact_{\mathbf{z}}(\ell) = \#\left\{\omega \text{ t.q. } |\omega| = \ell \text{ et } \omega \text{ facteur de } \mathbf{z}\right\}.$$

8.4.2 Propositions de Rivest

Nous présentons maintenant la proposition concrête principale de Rivest et une variante qu'il a également introduite. Pour comprendre les propositions de Rivest, il faut d'abord que nous introduisions une suite sans carré abélien appelée suite de Keränen.

Une suite infinie sans carré abélien

En 1992, Keränen [Ker92] a trouvé un mot infini sans carré abélien sur 4 lettres (il n'y a pas de mots sans carré abélien sur trois lettres). Dans ce papier, nous appelons ce mot infini abélien sans carré la suite de Keränen et nous notons par $\mathbf{k}$ la suite composée de ses 2^k premiers éléments. On ne rentre pas dans les détails de sa construction, leur description précise peut être trouvée dans [Ker92, Ker03, Riv05]. Il est cependant intéressant de remarquer que cette suite a une structure très régulière, structure que l'on peut exploiter pour prouver que $\mathbf{k}$ a une complexité inhabituellement basse.

Lemme 8.1. *Pour $\ell \leq 85$, on a :*

$$Fact_{\mathbf{k}}(\ell) \leq 8 \cdot \ell + 332.$$

En calculant le nombre exact de facteurs de $\mathbf{k}$, il est possible d'observer que cette borne est fine (par exemple il y a exactement 732 facteurs de taille 50). Ce résultat sera essentiel dans notre attaque.

Keränen-DMD

Rivest a suggéré d'utiliser directement la suite de Keränen comme tramage. L'alphabet utilisé pour le tramage est l'alphabet $\mathcal{A} = \{a, b, c, d\}$ qui peut être codé sur deux bits. Le nombre de bits de donnée en entrée de la fonction de compression est donc réduit de seulement 2 bits, la perte d'efficacité du hachage liée à l'ajout du tramage est donc faible. Il est possible de générer la suite de Keränen en ligne, un symbole à la fois, en temps amorti constant et avec une complexité en espace logarithmique. On fera référence à cette proposition sous le nom Keränen-DMD (pour *Keränen-Dithered Merkle-Damgård*).

La proposition concrête de Rivest

Pour accélérer la génération de la suite de tramage, Rivest a proposé un schéma légèrement modifié dans lequel les lettres utilisées pour le tramage sont codées sur 16 bits. Si le message M est tel que M_{pad} est constitué de m blocs, alors pour $1 \leq i < m$ le i^{e} élément du tramage est de la forme :

$$\left(0, \mathbf{k}\left[\left\lfloor i/2^{13} \right\rfloor\right], i \mod 2^{13}\right) \in \{0,1\} \times \mathcal{A} \times \{0,1\}^{13}$$

L'idée est d'incrémenter le compteur pour chaque symbole de tramage et de passer à la prochaine lettre dans la suite de Keränen seulement lorsque le compteur repasse à 0. Cette suite de tramage dite « diluée » peut être générée près de 2^{13} fois plus vite que la suite de Keränen.

On impose que le dernier symbole de la suite de tramage ait une forme différente (on rappelle que b désigne le nombre de bits d'un bloc de message) :

$$(1, |M| \mod b) \in \{0,1\} \times \{0,1\}^{15}$$

Dans la suite, on fera référence à cette proposition, qui est la proposition concrète de Rivest, par l'anagramme DMD-CP (pour *Dithered Merkle-Damgård – Concrete Proposal*).

La motivation originelle de cette proposition est de réduire le coût de calcul du tramage, mais cela a également comme conséquence involontaire d'augmenter la complexité de la suite de tramage générée, par rapport à la suite de Keränen. Cependant il est possible de prouver que cette augmentation reste assez faible :

Lemme 8.2. *Soit* $\mathbf{c}$ *la suite obtenue par dilution de* $\mathbf{k}$ *par un compteur sur 13 bits. Alors pour tout* $0 \leq \ell < 2^{13}$*, on a :*

$$Fact_{\mathbf{c}}(\ell) = 8 \cdot \ell + 32760.$$

La dilution ne génère pas une suite de complexité exponentiellement plus grande que celle de Keränen : elle est toujours linéaire en ℓ, même si le terme constant est plus grand à cause du compteur.

8.5 Attaque contre le mode cascade sans tramage

Nous allons maintenant décrire une nouvelle attaque pour trouver une seconde préimage sur un mode itéré. Elle s'appuie fortement sur la structure d'*arbre de collision* (aussi appelée *diamant*) introduite par Kelsey et Kohno [KK06]. Avant de présenter l'attaque qui l'utilise contre le mode cascade avec tramage, on décrit une version plus simple de cette attaque contre le mode cascade sans tramage.

8.5.1 La structure d'arbre de collision

Un arbre de collision de taille ℓ est une multicollision qui a la forme d'un arbre binaire complet de hauteur ℓ, avec 2^ℓ feuilles. Ses nœuds sont étiquetés par des valeurs de chainage sur n bits et ses arêtes sont étiquetées par des blocs de message sur b bits, de telle sorte que si (y, y') est une arête étiqueté par $x \in \{0,1\}^b$ alors $h(y, x) = y'$. Par conséquent, de n'importe laquelle des 2^ℓ feuilles, il part un chemin étiqueté par ℓ blocs de message et qui mène à la même valeur cible y_T, étiquette de la racine de l'arbre.

Un arbre de collision de profondeur ℓ peut être construit avec une complexité en temps et en espace de $2^{\frac{n}{2}+\frac{\ell}{2}+2}$. Tout d'abord, choisir 2^ℓ valeurs distinctes y_i pour les feuilles de l'arbre et $2^{n/2-\ell/2+1}$ blocs de message m_j. Pour chacun des couples (i, j), calculer $h(y_i, m_j)$ et trouver tous les quadruplets (u, u', v, v') tels que $u \neq v$ et $h(y_u, m_{u'}) = h(y_v, m_{v'})$. Des expériences menées par Kelsey et Konho (voir [KK06] pour plus de détails) montrent qu'avec grande probabilité il existe $2^{\ell-1}$ quadruplets $(u_i, u'_i, v_i, v'_i)_{1 \leq i \leq 2^{\ell-1}}$ tels que, pour tout $i \neq j$: $u_i \neq v_i \neq u_j \neq v_j$. À partir de ces quadruplets, nous pouvons construire le premier étage de notre arbre binaire de collision : pour chacun des quadruplets $(u_i, u'_i, v_i, v'_i)_{1 \leq i \leq 2^{\ell-1}}$ on crée une arête étiquetée par $m_{u'_i}$ entre y_{u_i} et $y'_i = h(y_{u_i}, m_{u'_i})$ et on crée une arête étiquetée par $m_{v'_i}$ entre y_{v_i} et y'_i. On a $2^{\ell-1}$ valeurs y'_i, avec forte probabilité toutes distinctes, et on reitère l'algorithme précédent jusqu'à avoir construit les ℓ étages de l'arbre de collisions.

La construction du i^{e} étage nécessite 2^i fois $\mathcal{O}(2^{n/2-i/2+1})$ requêtes à h, donc au total $\mathcal{O}(2^{n/2+i/2+1})$ requêtes. La construction d'un arbre de collision de taille ℓ exige donc $\mathcal{O}(2^{n/2+\ell/2})$ requêtes à h.

8.5.2 Attaque en seconde préimage sur le mode cascade

Soit M un message cible constitué de 2^k blocs. L'idée principale de notre attaque est que connecter un message à un arbre de collision peut être fait en moins de 2^n appels à la fonction de compression. Par ailleurs, connecter la racine de l'arbre à une des 2^k valeurs de hachage intermédiaires calculées durant le calcul de $H(M)$ prend seulement 2^{n-k} appels à la fonction de compression. L'attaque fonctionne en 4 étapes comme décrit dans la figure 8.1.

1. Génération de l'arbre de collision : créer un arbre de collision de profondeur ℓ avec des feuilles de valeur deux à deux distinctes ; soit y_T la valeur de la racine. On remarque que ceci peut être fait une seule fois, et peut être réutilisé pour calculer des secondes préimages pour de multiples messages.
2. Connecter y_T à une valeur de hachage intermédiaire du message M. Ceci peut être fait en générant des blocs de message aléatoires B, jusqu'à ce que $h(y_T, B) = y_{i_0}$ pour un certain i_0, $\ell + 1 \leq i_0 < |M|_b$. Soit B_0 un bloc de message satisfaisant cette condition.
3. Générer un préfixe quelconque P constitué de $i_0 - \ell - 1$ blocs et dont le haché est une des valeurs étiquetant une des feuilles de l'arbre de collision. Soit $y = h^*(IV, P)$ cette valeur et soit T la chaîne de ℓ blocs traversant l'arbre de collision de y jusqu'à y_T.
4. Former le message $M' = P\|T\|B_0\|x_{i_0+1} \ldots x_{2^k}$.

FIG. 8.1 – Résumé de l'attaque contre le mode cascade classique.

Les messages M' et M sont de même taille et ont la même valeur de hachage intermédiaire avant le renforcement, par conséquent ils produisent la même valeur de hachage malgré le renforcement de Merkel et Damgård.

Un arbre de collision de profondeur ℓ peut être construit avec une complexité en temps et en espace de $2^{\frac{n}{2}+\frac{\ell}{2}+2}$. La seconde étape de l'attaque peut être effectuée avec 2^{n-k} appels à la fonction de compression, et la troisième avec $2^{n-\ell}$ appels. La complexité temporelle totale de l'attaque est donc : $2^{\frac{n}{2}+\frac{\ell}{2}+2} + 2^{n-k} + 2^{n-\ell}$. Cette expression est minimale pour $\ell = (n-2)/3$, et dans ce cas, le coût total devient approximativement $5 \cdot 2^{2n/3} + 2^{n-k}$.

8.5.3 Comparaison avec Kelsey et Schneier

Contre le mode cascade originel, l'attaque de [KS05] est plus efficace que la nôtre (sur SHA-1, ils peuvent trouver une seconde préimage d'un message de taille 2^{55} en 2^{105}, alors que nous avons besoin de 2^{109} appels à la fonction de compression pour obtenir le même résultat).

Cependant, notre technique donne à l'adversaire plus de contrôle sur la seconde préimage, puisqu'elle permet typiquement de choisir librement tous les blocs de P sauf 1 (P a été introduit dans la figure 8.1), soit à peu près la première moitié du message. Par exemple, on peut choisir de répliquer la plupart du message cible, ce qui mène à une seconde préimage qui diffère du message original par seulement $\ell + 2$ blocs.

La différence apparente principale entre les deux techniques est que l'attaque de Kelsey et Schneier s'appuie sur les *messages extensibles*. Leur attaque construit un message extensible en temps $k \cdot 2^{n/2+1}$. Notre attaque peut aussi être vue comme une nouvelle technique plus flexible pour construire des messages extensibles, en choisissant un préfixe de la bonne taille et en le connectant à l'arbre de collision. Ceci peut être fait en temps $2^{n/2+k/2+2} + 2^{n-k}$. Bien qu'elle soit

plus coûteuse, cette nouvelle technique peut être adaptée pour fonctionner quand un tramage est ajouté en entrée, comme nous allons le montrer dans la suite.

8.6 Attaque en seconde préimage contre le mode cascade avec tramage

Dans cette section, nous présentons la première attaque en seconde préimage connue contre le mode cascade tramé de Rivest. Dans la sous-section 8.6.2, nous adaptons l'attaque de la section 8.5 à Keränen-DMD et on obtient une préimage en temps $(k + 40.5) \cdot 2^{n-k+3}$. On applique ensuite l'attaque étendue à DMD-CP, et on obtient alors une seconde préimage en approximativement 2^{n-k+15} évaluations de la fonction de la compression.

8.6.1 Adaptation de l'attaque contre le mode cascade avec tramage

On se place dans un cadre général en supposant que l'algorithme de hachage utilise une suite de tramage quelconque $\mathbf{z}$. Quand on construit l'arbre de collision, on doit choisir quels symboles de tramage utiliser. Une solution simple est d'utiliser le même symbole de tramage pour toutes les arêtes situées à la même profondeur dans l'arbre. Nous allons également avoir besoin d'une lettre supplémentaire pour connecter l'arbre de collision au message M. Ainsi, afin de construire un arbre de collision de profondeur ℓ, nous devons fixer un mot ω de taille $\ell + 1$, utiliser $\omega[i]$ comme symbole de tramage à la profondeur i et utiliser la dernière lettre de ω pour réaliser la connexion.

La séquence de tramage rend le haché d'un bloc dépendant de sa position dans le message. Par conséquent, l'arbre de collision ne peut être connecté à sa cible qu'à certaines positions, à savoir aux positions où ω et $\mathbf{z}$ coincident. L'ensemble des positions dans le message où cela est possible est donc donné par :

$$Range(\omega) = \Big\{ i \in \mathbb{N} \;\Big|\; \big(\ell + 1 \leq i\big) \wedge \big(\mathbf{z}[i-\ell] \ldots \mathbf{z}[i] = \omega\big) \Big\}.$$

En d'autre termes, lors de la seconde étape de l'attaque, on trouve un bloc B_0 pour connecter l'arbre de collision au message, c'est-à-dire tel que $h(y_T, B_0, \omega[\ell + 1]) = y_{i_0}$. Si $i_0 \in Range(\omega)$, il va être possible de construire la seconde préimage. Sinon, il faut recommencer et trouver un autre bloc B_0. On voit que plus ω est présent dans $\mathbf{z}$, c'est-à-dire plus $Range(\omega)$ est de grande taille, plus on a de chance que i_0 soit dans $Range(\omega)$ et moins on sera obligé de recommencer. Idéalement, ω devrait donc être le facteur de longueur $\ell + 1$ qui apparaît le plus fréquemment dans $\mathbf{z}$. Quelle est la probabilité que ce facteur le plus fréquent ω apparaisse à une position aléatoire dans $\mathbf{z}$? Bien que cette probabilité soit fortement dépendente de la suite considérée, il est possible de donner une borne générique : dans le pire des cas, tous les facteurs de taille $\ell + 1$ apparaissent dans $\mathbf{z}$ avec la même fréquence. Dans ce cas, la probabilité qu'un facteur de taille $\ell + 1$ choisi au hasard dans $\mathbf{z}$ soit le mot ω est égale à $1/Fact_{\mathbf{z}}(\ell + 1)$. En moyenne, on sera donc amené à recommencer la deuxième étape au plus $Fact_{\mathbf{z}}(\ell + 1)$, avant de trouver un $i_0 \in Range(\omega)$.

La dernière étape de l'attaque est similaire à l'attaque originelle. L'attaque est résumée dans la figure 8.2 ci-après.

La complexité des premières et dernières étapes ne change pas. Par contre celle de la deuxième étape varie. Le coût de la recherche d'une seconde préimage pour une suite donnée $\mathbf{z}$, dans le pire des cas où tous les facteurs apparaissent avec exactement la même fréquence, est donné par :

$$2^{\frac{n}{2}+\frac{\ell}{2}+2} + Fact_{\mathbf{z}}(\ell + 1) \cdot 2^{n-k} + 2^{n-\ell}.$$

1. Choisir le facteur ω le plus fréquent de $\mathbf{z}$, avec $|\omega| = \ell + 1$.
2. Construire un arbre de collision de profondeur ℓ en utilisant ω comme symboles de tramage pour chaque chemin reliant une feuille à la racine. Soit y_T l'étiquette de la racine de l'arbre.
3. trouver un bloc B_0 connectant y_T à n'importe que y_i (disons y_{i_0}) en utilisant $\omega[\ell+1]$ comme lettre de tramage. Répéter cette étape jusqu'à ce que $i_0 \in Range(\omega)$.
4. Mener les dernières étapes de l'attaque comme décrit dans la figure 8.1.

FIG. 8.2 – Résumé de l'attaque quand on utilise une suite de tramage $\mathbf{z}$.

On remarque que la propriété principale de $\mathbf{z}$ influençant le coût de cette deuxième étape est sa complexité, alors que son absence de répétition influe le coût de l'attaque de Kelsey et Schneier.

8.6.2 Application aux fonctions proposées par Rivest

On va appliquer l'attaque précédente au cas particulier des fonctions proposées par Rivest, contre lesquelles elle se révèle particulièrement efficace.

Cryptanalyse de Keränen-DMD

Le coût de l'attaque étendue contre Keränen-DMD dépend de la complexité de la suite $\mathbf{k}$. Le lemme 8.1 nous dit que $Fact_{\mathbf{k}}(\ell) \leq 8 \cdot \ell + 332$ pour $\ell \leq 85$. Malgré la forte absence de répétitions, la suite $\mathbf{k}$ offre un faible niveau de sécurité contre notre attaque puisque en évaluant le coût de notre attaque sur Keranen-DMD on obtient :

$$2^{\frac{n}{2}+\frac{\ell}{2}+2} + (8 \cdot \ell + 340) \cdot 2^{n-k} + 2^{n-\ell}.$$

Si n est du même ordre de grandeur que $3k$, alors le premier terme de cette somme est du même ordre de grandeur que les deux autres et si $n \gg 3k$ alors il devient négligeable. Nous utilisons cette approximation plusieurs fois dans la suite. En choisissant $\ell = k - 3$, le coût total de l'attaque est approximativement de $(k + 40.5) \cdot 2^{n-k+3}$ ce qui est beaucoup plus petit que 2^n malgré le tramage. Pour SHA–1 par exemple, la complexité est d'environ 2^{115}.

Finissons cette application par une remarque. Avec $\ell = 50$, et supposant que tous les facteurs ont la même fréquence d'apparition dans $\mathbf{k}$, la probabilité qu'un bloc, connectant l'arbre de collision au message, tombe dans $Range(\omega)$, pour n'importe quel ω, est de $2^{-9.5}$. On observe que dans les premiers 2^{30} symboles de $\mathbf{k}$ il y a des facteurs de taille 50 qui n'apparaissent que 910237 fois alors que d'autres apparaissent 2941800 fois. En choisissant un de ces derniers et en supposant que sa fréquence d'apparition est la même tout au long de $\mathbf{k}$, la probabilité d'apparition devient $2^{-8.5}$. Approximer la complexité de l'attaque en utilisant $Fact(\ell+1)$ conduit à une bonne estimation du coût réel de l'attaque, c'est pourquoi on s'est contenté de la borne issue du cas où tous les messages sont équiprobables.

Cryptanalyse de DMD-CP

Nous appliquons maintenant l'attaque contre la proposition concrête de Rivest. Comme expliqué dans la section 8.4.2, on rappelle que la suite utilisée pour cette proposition est basée sur la suite de Keränen, mais que l'on change de symbole seulement quand un compteurs sur 13

bits repasse à 0. On rappelle que le lemme 8.2 nous dit que la complexité de cette suite $\mathbf{c}$ est $Fact_{\mathbf{c}}(\ell) = 8 \cdot \ell + 32760$ pour tout $0 \leq \ell < 2^{13}$. Le coût de l'attaque vaut par conséquent :

$$2^{\frac{n}{2}+\frac{\ell}{2}+2} + (8 \cdot \ell + 32768) \cdot 2^{n-k} + 2^{n-\ell}.$$

Encore une fois, si n est plus grand qu'environ $3k$, la meilleure valeur pour ℓ est $k-3$ et la complexité de l'attaque est donc approximativement : $(k + 4094) \cdot 2^{n-k+3} \simeq 2^{n-k+15}$. Dans le cas de SHA-1, une seconde préimage peut être calculée en temps 2^{120}.

8.7 Attaque contre une famille de fonctions de hachage universelle à sens unique

Dans cette section, nous montrons que notre attaque est suffisamment générique pour être appliquée contre des fonctions possédant des propriétés de sécurité différentes, à savoir des familles de fonctions de hachage universelles à sens unique (ou UOWHF pour *Universal One-Way Hash Function* en anglais). Une UOWHF est une famille de fonctions de hachage H pour laquelle tout adversaire $\mathcal{A}$ calculatoirement limité gagne le jeu suivant avec probabilité négligeable. Tout d'abord $\mathcal{A}$ choisit un message M et l'envoie au challenger, ensuite ce dernier choisit une clef K parfaitement au hasard et la donne à $\mathcal{A}$. L'adversaire gagne s'il casse la résistance aux collisions ciblées ou propriété TCR (pour *Target Collision Resistance*), c'est-à-dire s'il génère un message M' différent de M qui collisionne pour la clef K (*i.e.* tel que $H_K(M) = H_K(M')$ avec $M \neq M'$).

Shoup [Sho00a] a proposé une construction simple pour une UOWHF qui hache des messages de taille arbitraire, étant donnée une UOWHF qui hache des messages de taille fixée. C'est un mode opératoire itéré, mais avant chaque itération, on choisit un masque parmi plusieurs possibles et on le xor à la valeur de chaînage. Le nombre de masques est logarithmique en la taille du message haché et l'ordre dans lequel ils sont utilisés est choisi attentivement pour maximiser la sécurité du schéma. Cette construction nous évoque le hachage avec tramage, à la nuance que le processus de tramage utilisé ici ne baisse pas la bande passante destinée aux données.

Tout d'abord nous décrivons rapidement la construction de Shoup et ensuite nous montrons comment notre attaque peut être appliquée. La complexité de l'attaque prouve que pour cette construction particulière la borne de sécurité prouvée par Shoup est presque fine.

8.7.1 Description de la famille

Cette construction a quelques similarités avec le mode cascade avec tramage. Soient h une famille de fonctions de hachage universelle à sens unique dont l'entrée est de taille fixe et K une clef, on a donc $h_K \colon \{0,1\}^n \times \{0,1\}^b \to \{0,1\}^n$. Cette fonction de compression est ensuite itérée, comme décrit ci-dessous, pour obtenir une UOWHF d'entrée de taille variable H_K.

Le schéma utilise un ensemble de masques $\mu_0, \ldots, \mu_{k-1}$ (où $2^k - 1$ est la longueur du plus long message possible), chacun d'entre eux étant une chaîne de bits choisie parfaitement au hasard dans $\{0,1\}^n$. La clef de la grande fonction est constituée de K et des masques. Après chaque application de la fonction de compression, un masque est xoré à la valeur de chaînage. L'ordre dans lequel les masques sont appliqués est défini par une suite sur l'alphabet $\mathcal{A} = \{0, \ldots, k-1\}$. Cette suite est $\mathbf{z}[i] = \nu_2(i)$, pour $1 \leq i \leq 2^k - 1$, où $\nu_2(i)$ désigne le plus grand entier ν tel que 2^ν divise i. Soit M un message tel que M_{pad} peut être divisé en m blocs $x_1, \ldots, x_m$ et soit y_0 une chaîne de n bits quelconque. On définit $y_i = h_K\left(y_{i-1} \oplus \mu_{\nu_2(i)}, x_i\right)$ et $H_K(M) = y_m$.

8.7.2 Adaptation de l'attaque

Dans [Sho00a], Shoup prouve le résultat de sécurité suivant :

Théorème 8.2 (Résultat principal de [Sho00a])**.** *Si un adversaire est capable de casser la propriété de résistance aux collisions ciblées de H avec probabilité ε en temps T, alors on peut construire un adversaire qui casse la résistance aux collisions ciblées de h en temps T, avec probabilité $\epsilon/2^k$.*

Dans cette section, nous montrons que cette borne est presque fine. Tout d'abord, nous donnons une définition alternative de la suite de tramage $\mathbf{z}$. On définit :

$$u_i = \begin{cases} 0 & \text{si } i = 1, \\ u_{i-1}.(i-1).u_{i-1} & \text{sinon.} \end{cases}$$

Par exemple, on a $u_4 = 010201030102010$. On montre facilement par récurrence que pour tout i, la taille de u_i est égale à $|u_i| = 2^i - 1$ et que u_i est un préfixe de $\mathbf{z}$. La séquence de tramage est donc simplement u_k.

La définition précédente nous montre également que pour tout i, u_i est compris 2 fois dans u_{i+1}, 4 fois dans u_{i+2},... On peut donc montrer par récurrence que u_i est compris 2^{k-i} fois dans $u_k = \mathbf{z}$.

Enfin, soit ℓ un paramètre que l'on fixera plus tard. Un des facteurs de $\mathbf{z}$ de taille $\ell + 1 < 2^k$ les plus fréquents est le préfixe de taille $\ell + 1$ de $\mathbf{z}$, que l'on note ω. C'est aussi un préfixe de u_j pour $j = \lceil \log_2(\ell + 2) \rceil$ (mais pas de u_{j-1} car u_{j-1} est trop petit). Or u_j lui-même apparaît 2^{k-j} fois dans $\mathbf{z} = u_k$ d'après la remarque précédente donc ω apparaît 2^{k-j} fois dans $\mathbf{z} = u_k$. La probabilité qu'un facteur de taille $\ell + 1$ pris aléatoirement dans $\mathbf{z}$ soit ω est égal au nombre d'occurences de ce candidat divisé par le nombre de chaînes de $\ell + 1$ bits dans $\mathbf{z}$. Par conséquent, cette probabilité est $\frac{2^{k-j}}{2^k - \ell - 1}$. Cette quantité peut être minorée par $2^{-j} \geq \frac{1}{2(\ell+2)}$.

Notre attaque peut être adaptée contre la propriété TCR de H de la manière suivante. Tout d'abord, on choisit au hasard un long message cible M de taille $2^k - 1$. Une fois que la clef est choisie au hasard par le challenger et nous est donnée, on construit un arbre de collision en utilisant les ℓ premières lettres de ω pour son tramage et on continue l'attaque comme indiqué dans la section 8.6. Comme lorsque l'on réussit à raccorder l'arbre de collision à une valeur intermédiaire de chaînage i_0, la probabilité que $i_0 \in \mathit{Range}(\omega)$ est supérieure à $1/2(\ell + 2)$, il faudra recommencer approximativement $2(\ell + 2)$ fois avant de réussir à bien raccorder l'arbre de collision. Le coût de cette étape est donc $2(\ell + 2) \cdot 2^{n-k}$. Le coût total de l'attaque est donc environ :

$$T = 2^{\frac{n}{2} + \frac{\ell}{2} + 2} + 2(\ell + 2) \cdot 2^{n-k} + 2^{n-\ell}.$$

Cette attaque casse la propriété TCR avec probabilité presque 1. Par conséquent, en utilisant le résultat de Shoup, on peut construire un adversaire $\mathcal{A}$ contre h en temps T et avec probabilité de succès environ $1/2^k$. Si h est une boîte noire, la meilleure attaque contre la propriété TCR est la recherche exhaustive. Par conséquent, le meilleur attaquant en temps T contre h a une probabilité de succès égale à $T/2^n$. Quand $n \geq 3k$, on fixe $\ell = k - 1$ et on a $T \simeq (2k + 4) \cdot 2^{n-k}$. Par conséquent le meilleur adversaire en temps T contre la propriété TCR de h a une probabilité de succès en $\mathcal{O}(k/2^k)$, alors que la probabilité de succès de $\mathcal{A}$ dans le même temps est de $1/2^k$. On en déduit qu'il y a uniquement un facteur k entre notre attaque et la preuve de sécurité de Shoup, il n'y a pas donc pas d'attaque vraiment meilleure que la notre et par ailleurs la preuve de Shoup est assez fine.

8.7.3 Amélioration de l'attaque

Il est possible d'améliorer l'attaque précédente de façon à obtenir seulement un facteur $\mathcal{O}(\log_2(k))$ entre la preuve de Shoup et notre attaque. Dans ce but, nous utilisons l'astuce suivante. À la place de s'intéresser au mot le plus fréquent de taille $\ell+1$, nous construisons un arbre de collision pour chacun des facteurs de taille ℓ, puis nous agrégeons tous ces arbres grâce à un dernier arbre de collision de taille s, pour lequel nous utilisons comme tramage le mot le plus fréquent de taille $s+1$. On verra que $s+1$ est plus petit que ℓ et donc qu'il est plus facile de connecter cet arbre de collision au message.

Avant de présenter l'attaque plus en détail, montrons qu'il y a moins de $k\ell$ facteurs de taille ℓ dans u_k. Un mot v de taille ℓ est uniquement déterminé par la valeur de la lettre la plus grande contenue dans v et par son emplacement dans v. Comme l'alphabet contient moins de k lettres, il y a au plus k possibilités pour la lettre la plus grande de v. Il y a au plus ℓ emplacements où on peut placer cette lettre, puisque v à ℓ lettres. Par conséquent, il y a au plus $k\ell$ mots de ℓ lettres.

Soit s tel que $2^s \geq k\ell$, c'est-à-dire $s = \lceil \log_2(k\ell) \rceil$, soit ω le préfixe de u_k de $s+1$ lettres de long et soient $\omega_1, \ldots, \omega_{kl}$ tous les mots de taille ℓ. L'attaque est décrite dans la figure 8.3.

1. Choisir $M = x_1, \ldots, x_{2^k-1}$ au hasard de taille $2^k - 1$ blocs et l'envoyer au challenger qui répond par la clef,
2. Calculer le haché de M, soit $y_1, \ldots, y_{2^k-1}$ les valeurs intermédiaires de chaînage lors du calcul du haché de M.
3. Pour tous les $1 \leq i \leq k\ell$ générer un arbre de collision de taille ℓ en utilisant ω_i pour le tramage, on note $y_{T,i}$ sa valeur cible,
4. Construire un arbre de collision de taille s, en utilisant les s premières lettres de ω pour le tramage et en choisissant comme feuilles de l'arbres les $y_{T,i}$, soit y_T sa valeur cible,
5. Relier ce dernier arbre de collision au message : soient i_0 et B_0 tels que $h(y_T, B_0, \omega[s+1]) = y_{i_0}$, si $i_0 \notin Range(\omega)$ alors recommencer cette étape,
6. Soit $\omega_j = \mathbf{z}[i_0 - s - \ell] \ldots \mathbf{z}[i_0 - s - 1]$, sélectionner l'arbre de collision fait avec ω_j comme tramage et relier IV à une feuille, notée y, de cette arbre en utilisant un message P de taille $i_0 - s - \ell$,
7. Soit T le message induit par le chemin dans l'arbre de collision de y à $y_{T,j}$ et par le chemin dans l'arbre de collision de $y_{T,j}$ à y_T ; renvoyer $M' = P \| T \| B_0 \| x_{i_0+1} \ldots x_{2^k-1}$.

FIG. 8.3 – Résumé de l'attaque améliorée contre une UOWHF.

Le coût de la construction des $k\ell$ arbres de collisions est $\mathcal{O}\left(k\ell \cdot 2^{\frac{n}{2}+\frac{\ell}{2}+2}\right)$, le coût du calcul de l'arbre de collision pour ω est $\mathcal{O}\left(2^{\frac{n}{2}+\frac{s}{2}+2}\right)$, ce qui est négligeable face au coût précédent, le coût du raccord de l'arbre de collision de ω est $2(s+2) \cdot 2^{n-k}$ et enfin le coût du raccord de IV à une feuille de l'arbre est $2^{n-\ell}$. Comme $s \approx \log_2(k\ell)$, le coût total de l'attaque est :

$$T = \mathcal{O}\left(k\ell 2^{\frac{n}{2}+\frac{\ell}{2}+2} + 2(\log_2(k\ell)+2) \cdot 2^{n-k} + 2^{n-\ell}\right).$$

En prenant $\ell = k-1$ et si $n \gg 3k + 4\log_2(k)$, alors $T = \mathcal{O}(\log_2(k) \cdot 2^{n-k})$ et un raisonnement identique à celui mené à la sous-section précédente prouve qu'il n'y a qu'un facteur $\log_2(k)$ entre la preuve de Shoup et notre attaque. La preuve est donc d'autant plus fine.

8.8 Contre-mesure

Dans cette section, nous étudions des contremesures qui permettent d'éviter notre attaque ou d'en limiter l'impact. Pour cela, on conserve l'idée d'ajouter un tramage.

8.8.1 Tramage à complexité exponentielle

Si on utilise une suite de tramage $\mathbf{z}$ telle que tous les mots de taille $\ell+1$ apparaissent à la même fréquence, alors la complexité de l'attaque est égale à $2^{\frac{n}{2}+\frac{\ell}{2}+2} + Fact_{\mathbf{z}}(\ell+1)\cdot 2^{n-k} + 2^{n-\ell}$. Si on note $Fact_{\mathbf{z}}(\ell+1) = 2^i$, la complexité de l'attaque s'écrit alors : $2^{\frac{n}{2}+\frac{\ell}{2}+2} + 2^{n-k+i} + 2^{n-\ell}$. En pratique le terme dominant est 2^{n-k+i}. En prenant $i = k$, on obtiendrait un schéma qui serait parfaitement résistant contre notre attaque. À défaut on essaie de choisir $\mathbf{z}$ telle que pour tout ℓ, tous les mots de taille $\ell+1$ apparaissent à la même fréquence et $Fact_{\mathbf{z}}(\ell+1)$ soit le plus grand possible.

Ouvrons une petite parenthèse. En toute rigueur on ne peut avoir $i = k$ car une suite de taille 2^k contient exactement $2^k - (\ell+1)$ sous-mots de taille $\ell+1$ (pas forcément tous distincts), donc au plus $2^k - (\ell+1)$ facteurs *distincts* de taille $\ell+1$. Par conséquent, $2^i \leq 2^k - (\ell+1)$. Cependant, on veut que la complexité globale de notre attaque soit inférieure à 2^n, il faut donc que $\ell \leq n$ pour que le premier terme $2^{\frac{n}{2}+\frac{\ell}{2}+2}$ soit inférieur à 2^n. On peut donc négliger le terme $\ell+1$ dans $2^k - (\ell+1)$, et on a bien $i \approx k$.

Utiliser un compteur (*i.e.* un grand alphabet) est une manière simple d'obtenir une suite de tramage de grande complexité. Par exemple, on peut prendre un compteur sur k bits et on obtient alors $i \approx k$. C'est essentiellement le choix fait par les concepteurs de Haifa [BD06], avec l'inconvénient qu'une telle suite de tramage consomme k bits dans la bande passante. On remarquera que comme le compteur ne repasse pas à 0, aucune variation de l'attaque de Kelsey et Schneier ne peut être appliquée à la construction avec tramage. Nous montrons dans la partie suivante que cette construction ne permet pas uniquement d'éviter notre attaque, mais qu'elle est bien optimale puisque on peut prouver que la sécurité en seconde préimage est en $\mathcal{O}(2^{n-1})$.

Une autre possibilité pour améliorer la résistance du hashing avec tramage de Rivest contre notre attaque est d'utiliser une suite de tramage de grande complexité sur un petit alphabet (afin de préserver de la bande passante). Il est en effet possible de construire une suite sans carré abélien de complexité exponentielle en se limitant à un alphabet de 6 lettres. Pour définir cette suite, entrecroisons deux suites. Soit $\mathbf{k}$ la suite de Keränen sur l'alphabet $\{a,b,c,d\}$ et soit $\mathbf{u}$ une suite sur $\{0,1\}$ qui ait une complexité exponentielle (on pourra prendre par exemple la suite issue de la concaténation du codage en binaire des nombres entiers par ordre croissant). Nous créons la suite $\mathbf{z}$ sur l'alphabet $\mathcal{A} = \{a,b,c,d,0,1\}$ en alternant $\mathbf{k}$ et $\mathbf{u}$: $\mathbf{z} = \mathbf{k}[1].\mathbf{u}[1].\mathbf{k}[2].\mathbf{u}[2].\ldots$. La suite obtenue hérite des deux propriétés : elle est sans carré et pour tout ℓ, $Fact(\ell) = \Omega\left(2^{\ell/2}\right)$. En prenant alors $\ell = 2k/3$, le coût total de l'attaque passe environ à $2^{n-2k/3}$, soit environ à 2^{123} pour SHA–1.

Enfin, citons une dernière possibilité. Pour tout $1 \leq i \leq k/2$, définissons $u_0 = 0$, $u_i = 0.(i).1.(i).2.\cdots.(i-2).(i).(i-1).(i).(i)$ et la suite de tramage $\mathbf{z} = u_0.u_1.\cdots.u_{2^{k/2}-1}$. On a donc $\mathbf{z} = 0011021220313233041424344\cdots$.Comme pour tout i, la taille de u_i est $|u_i| = 2i+1$, la taille totale de $\mathbf{z}$ est 2^k et son alphabet est $[\![0,\, 2^{k/2}-1]\!]$. On peut facilement montrer que $Fact_{\mathbf{z}}(2) = 2^k - 2$ et, partant, que $Fact_{\mathbf{z}}(\ell+1) = 2^k - (\ell+1)$ pour tout ℓ positif (on a construit cette suite telle que tout mot de deux lettres ne soit présent qu'une seule fois). Notre attaque est donc ramenée à une complexité de $\mathcal{O}(2^n)$. Puisque son alphabet est $[\![0,\, 2^{k/2}-1]\!]$, elle nécessite donc de prendre une bande passante de $k/2$ bits, contrairement à Haifa qui prend k bits de

bande passante ; cependant, cette suite n'est pas prouvée sûre contre toutes les attaques en seconde préimage, alors que comme nous allons le voir, c'est le cas de HAIFA.

8.8.2 Sécurité d'Haifa

Nous commençons par une description d'HAIFA avant de prouver sa résistance aux secondes préimages.

Description d'HAIFA

HAIFA est basée sur la construction cascade classique. La fonction de hachage itérée $H : \{0,1\}^* \to \{0,1\}^n$ est construite en itérant une fonction de compression $h\colon \{0,1\}^n \times \{0,1\}^b \times \{0,1\}^s \to \{0,1\}^n$ qui a trois entrées : la nouvelle entrée est un compteur. Le procédé de hachage d'un message M est le suivant :

- diviser le message $M_{pad} = M \| pad(|M|)$ en m blocs $x_1, \ldots, x_m$ de b bits chacun,
- on fixe $y_0 = IV$ où IV est la valeur initiale,
- pour chaque bloc de message x_i on calcule $y_i = h\left(y_{i-1}, x_i, i\right)$,
- on renvoie $H(M) = y_m$.

Le *padding* utilisé est le *padding* classique décrit en introduction, à la sous-section 2.5.4

Résultat de sécurité

Nous modélisons la fonction de compression par un oracle aléatoire et ce pour deux raisons. Tout d'abord, nous voulons prendre en compte uniquement les attaques génériques, l'adversaire ne doit pas être capable d'exploiter une faiblesse ou propriété particulière de la fonction de compression. Par ailleurs, pour les messages constitués d'un seul bloc, la fonction de hachage itérée est réduite à la fonction de compression, par conséquent si la fonction de compression n'est pas une bonne fonction de hachage, la fonction de hachage itérée ne peut pas être une bonne fonction de hachage non plus. Le résultat de sécurité que nous obtenons en raisonnant à propos du mode opératoire dans le modèle de l'oracle aléatoire est intrinsèquement différent de ceux obtenus par une preuve par réduction comme celle de [RS04].

Nous considérons un adversaire $\mathcal{A}$ qui, étant donné un message M, essaie de trouver un message M' tel que $H(M) = H(M')$. Soit Q le nombre de requêtes que $\mathcal{A}$ peut envoyer à l'oracle de h. Nous rappelons que nous forçons $\mathcal{A}$ à *toujours* renvoyer un message M', même si ce n'est pas une seconde préimage, et à avoir évalué $H(M)$ et $H(M')$ en effectuant les requêtes correspondantes à l'oracle aléatoire. Cela empêche $\mathcal{A}$ de générer un message arbitrairement long.

Théorème 8.3 (Sécurité de HAIFA)**.** *Soit H une fonction de hachage itérée suivant le schéma de* HAIFA *et $\mathcal{A}$ un adversaire en seconde préimage contre H qui fait au plus Q requêtes à h. On a alors :*

$$\Pr\left[\mathcal{A} \Rightarrow x' \colon H(x) = H(x')\right] \leq \frac{2Q}{2^n}.$$

Ce théorème vient essentiellement du fait que quand un adversaire soumet une requête à h, il doit spécifier la valeur i_0 du compteur. En effet, l'adversaire gagne si et seulement si il peut trouver un message $\tilde{M}$ tel que $h^*(IV, \tilde{M}) = y_i$ et $|\tilde{M}|_b = i$ ou tel que $H(\tilde{M}) = y_m$ (on réutilise ici les notations introduites précédemment). Cela implique que pour toute requête où la valeur du compteur est i_0, l'adversaire gagne seulement si l'oracle renvoie y_{i_0} ou y_m.

HAIFA atteint le même niveau de sécurité qu'un oracle aléatoire, ce qui est la meilleure sécurité que l'on peut espérer obtenir.

Démonstration. Nous simulons l'exécution d'un adversaire $\mathcal{A}$ et enregistrons toutes les requêtes faites par $\mathcal{A}$ à l'oracle aléatoire : ces requêtes forment un ensemble S de quadruplets (y, x, i, y') tels que $y' = h(y, x, i)$. Comme expliqué dans le modèle de sécurité, on suppose que $\mathcal{A}$ évalue $H(M)$, de telle sorte que $\mathcal{A}$ fait les requêtes correspondantes à l'oracle à un moment ou à un autre. On notera ces requêtes $(y_{i-1}, x_i, i, y_i)_{1 \leq i \leq m}$, où $m = |M_{pad}|_b$. En particulier, $H(M) = y_m$.

Si $\mathcal{A}$ réussit à trouver une seconde préimage, alors $\mathcal{A}$ a produit en particulier une collision. Il est bien connu que pour le mode cascade avec *renforcement*, une collision sur la fonction de hachage implique une collision sur la fonction de compression [Mer89, Dam89]. Cependant, dans notre cas la collision est *contrainte* : on doit trouver une collision sur un des y_i. Trouver une telle collision est plus dur que trouver une collision quelconque.

Supposons que $\mathcal{A}$ gagne et renvoie un message M'. Notons $(y'_{i-1}, x'_i, i, y'_i)_{1 \leq i \leq m'}$ les requêtes faites pour calculer $H(M') = y_{m'}$, où $|M'_{pad}|_b = m'$. On distingue deux cas.

1. Si $\left|M\right| \neq \left|M'\right|$, alors les valeurs du compteur entrant dans la fonction de compression lors de la dernière invocation sont différentes. Par conséquent, $\mathcal{A}$ a trouvé une seconde préimage sur y_m avec une valeur de compteur différente de m. Notons E_1 l'événement précédent : il existe une requête (y, x, i_0, y') dans S telle que $y' = y_m$ et $i_0 \neq m$.
2. Sinon, $\left|M\right| = \left|M'\right|$. Nous allons formaliser notre intuition de collision contrainte. Considérons l'événement suivant, que nous appellons E_2. On dit que E_2 est réalisé si et seulement si il existe une requête (y, x, i_0, y') dans S telle que $y' = y_{i_0}$ et telle que $(y, x) \neq (y_{i_0-1}, x_{i_0})$. Cela signifie qu'il y a une collision avec la même valeur de compteur. Nous allons montrer que dans le cas où $\left|M\right| = \left|M'\right|$, l'événement E_2 est vrai.

   Puisque $y_m = y'_{m'}$, ou bien il y a une collision pour h (avec comme valeur de compteur m), ou bien $(x_m, y_{m-1}) = (x'_m, y'_{m-1})$. Dans ce dernier cas, ou bien il y a déjà une collision sur h (avec comme valeur de compteur $m-1$) ou $(x_{m-1}, y_{m-2}) = (x'_{m-1}, y'_{m-2})$. On peut répéter cette argument itérativement. Puisque $m = m'$, alors ou bien il y a une collision à un certain point pour h avec la même valeur de compteur, ou bien $x_i = x'_i$, pour tous les i, $1 \leq i \leq m$. Dans ce dernier cas, $M = M'$, ce qui est impossible.

Par conséquent, $\Pr[\mathcal{A} \text{ gagne}] \leq \Pr[E_1 \wedge E_2]$. On remarque que le fait que $E_1 \wedge E_2$ soit vrai n'implique pas nécessairement que $\mathcal{A}$ ait gagné.

Maintenant montrons l'inégalité suivante :

$$\Pr\left[E_1 \wedge E_2\right] \leq \frac{2Q}{2^n}.$$

À chaque fois que $\mathcal{A}$ fait une requête (y, x, i) à l'oracle, la probabilité que y', la réponse de l'oracle choisie aléatoirement par celui-ci, soit égale à y_i ou à y_m est majorée par $2/2^n$. La probabilité que $E_1 \wedge E_2$ devienne vraie à cette étape est donc inférieure à $2/2^n$. Puisque $\mathcal{A}$ envoie Q requêtes à l'oracle durant son exécution, on obtient la borne annoncée. □

Cette borne est fine puisque, si $|M_{pad}|_b = m \geq 2$, nous sommes capables d'exhiber un attaquant en seconde préimage dont la probabilité de succès est égale à $2Q/2^n$. Cet attaquant est le suivant : choisir Q messages x'_i distincts tels que $x'_i \| pad(|x'_i|)$ fassent exactement 1 bloc et soumettre à l'oracle les $(IV, x'_i \| pad(|x'_i|), 1)$. Soit y'_i les réponses de l'oracles. S'il existe un indice i tel que $y'_i = y_m$ alors renvoyer x'_i qui est une seconde préimage de M. S'il existe un indice i tel que $y'_i = y_1$ alors renvoyer $x'_i \| pad(|x'_i|) \| x_2 \| \ldots \| x_{|M|_b}$ qui est une seconde préimage pour M. Si $y_1 \neq y_m$, alors la probabilité de succès de l'attaquant est $2Q/2^n$.

8.9 Conclusion

Dans ce chapitre, nous avons exposé la première attaque en seconde préimage contre le mode cascade avec tramage, attaque qui prouve que les modes opératoires proposés par Rivest n'atteignent pas la sécurité en seconde préimage escomptée. Par ailleurs, cette attaque est suffisamment générique pour pouvoir être adaptée contre la propriété TCR de la construction de Shoup, ce qui nous permet de montrer que la preuve de Shoup est très fine. Enfin, nous avons prouvé qu'il est possible d'obtenir une résistance aux préimages optimale en utilisant le mode HAIFA proposé récemment par Biham et Dunkelman.

Conclusion

Dans ce mémoire, nous avons étudié certains mécanismes cryptographiques permettant d'établir un canal authentifié. Nous avons d'abord adopté une approche générale en considérant la sécurité du protocole dans son ensemble. Ensuite, nous nous sommes intéressés à une composante particulière de ce protocole, composante trop souvent négligée : l'extraction de clefs. Enfin, nous sommes descendus un niveau plus bas afin de détailler des éléments intervenants dans ce protocole : nous avons analysé des modes opératoires destinés à la construction de fonctions de hachage. À chaque étape nous avons à la fois proposé des analyses de sécurité pour des protocoles ou des modes opératoires existants, mais aussi, présenté de nouvelles propositions.

Dans la première partie de ce mémoire, nous avons considéré un protocole créant un canal authentifié, confidentiel et intègre grâce à des outils issus de cryptographie symétrique et de cryptographie asymétrique : la cryptographie asymétrique permet d'effectuer un échange de clefs quand la cryptographie symétrique permet d'établir le canal lui-même tout en garantissant un bon niveau d'efficacité. Nous avons proposé un nouveau protocole d'échange de clefs qui combine les trois facteurs d'authentification classiques (mot de passe, biométrie et carte à puce) de telle sorte que l'adversaire doit casser les trois facteurs d'authentification en même temps pour espérer réussir à s'authentifier. Ce protocole est le premier protocole d'authentification multi-facteurs intégrant la biométrie et la considérant comme une donnée publique. Pour cette raison nous avons proposé un nouveau modèle de sécurité et avons prouvé le protocole dans ce modèle. Ensuite, nous avons étudié CCM, un schéma de chiffrement authentifié avec données authentifiées associées. CCM est prouvé sûr si les messages respectent un format particulier qui limite l'efficacité du schéma. Nous avons montré que ces contraintes de format sont non seulement suffisantes mais aussi nécessaires pour garantir la sécurité, et qu'il ne faut surtout pas s'en affranchir. Par ailleurs, nous avons proposé une amélioration à CCM qui permet, si ce n'est d'obtenir une meilleure efficacité, au moins de garantir une plus grande sécurité.

Dans la deuxième partie de ce mémoire, nous avons étudié des primitives d'extraction de clefs prouvées sûres dans le modèle standard. Nous avons d'abord montré que si l'on conserve les bits de poids faible d'un élément Diffie-Hellman (et que l'on écarte les bits de poids fort) et si l'échange a eu lieu dans un groupe de grande taille pour lequel l'hypothèse DDH est vraie, alors on obtient une chaîne de bits presque uniformément distribuée qui de ce fait peut servir de clef à un algorithme de cryptographie symétrique. Ensuite, nous avons présenté une analyse, dans le modèle standard, des fonctions de hachage, notamment HMAC, comme extracteur d'entropie calculatoire. Cette analyse valide certains des schémas d'extraction de clefs utilisés en pratique, comme celui de TLS, mais surtout met en exergue les bonnes manières d'utiliser les fonctions de hachage pour l'extraction de clefs. Nous avons principalement insisté sur le fait qu'il est préférable d'utiliser SHA-384 que SHA-512 car la taille de la sortie de SHA-384 est tronquée, donc nos résultats s'appliquent, alors que l'utilisation de SHA-512 ne peut être prouvée sûre que dans le modèle de l'oracle aléatoire.

Dans la troisième partie, nous avons examiné certains modes opératoires utilisés pour la

construction de fonctions de hachage. Tout d'abord nous nous sommes intéressés aux modes itérés qui utilisent des algorithmes de chiffrement par bloc dont la clef est fixée une fois pour toute au début et n'évolue plus lors du calcul du haché. L'algorithme SMASH entre dans cette catégorie et nous avons présenté les premières attaques en collision contre les versions améliorées du mode opératoire de SMASH. Suite à cela, nous avons proposé une autre amélioration de ce mode opératoire, amélioration que nous avons prouvée sûre dans le modèle du chiffrement idéal. Enfin, nous avons étudié des modes opératoires dont l'objectif est de renforcer le mode cascade pour obtenir un mode dont la résistance aux secondes préimages est optimale. Nous présentons une nouvelle attaque qui peut être utilisée contre certains modes mis au point par Rivest et prouvons ainsi que ces modes n'atteignent pas la sécurité escomptée. Ensuite, nous démontrons que le mode HAIFA, lui, a bien le niveau de sécurité espéré.

A

Problème des anniversaires généralisé à 4 listes

Dans cette section nous présentons une borne inférieure pour le problème des anniversaires généralisés dans un cas spécifique. Plus précisément, nous donnons une minoration de la probabilité que, parmi 4 listes d'éléments aléatoires uniformes, il existe un élément dans chaque liste tel que le xor de ces éléments est égal à 0. Même si ces résultats sont très connus et communément utilisés et admis sans preuve, comme dans [Wag02], dans un soucis de complétude, nous en donnons ici une preuve.

La preuve qui est présentée ci-dessous est très facilement extensible au cas à $2k$ listes, ce qui donne une minoration en $(1-1/e)\cdot q^{2k}/2^{n+1}$.

Théorème A.4. *Soit $q \leq 2^{n/4}$ et soient $(W_i)_{1\leq i\leq q}$, $(X_j)_{1\leq j\leq q}$, $(Y_k)_{1\leq k\leq q}$ et $(Z_\ell)_{1\leq \ell\leq q}$ 4 suites de variables aléatoires indépendantes et uniformément distribuées sur $\{0,1\}^n$. La probabilité qu'il existe (i,j,k,ℓ) tel que $W_i \oplus X_j \oplus Y_k \oplus Z_\ell = 0$ est minorée par $(1-1/e)\cdot q^4/2^{n+1}$.*

Démonstration. Soit $A_{i,j,k} = W_i \oplus X_j \oplus Y_k$, pour $1 \leq i,j,k \leq q$ et $S = \{A_{i,j,k}\}$ (on remarque que $|S| \leq q^3$). Notre but est de majorer la probabilité que pour tout (i,j,k,ℓ), $W_i \oplus X_j \oplus Y_k \oplus Z_\ell \neq 0$. Cette probabilité est égale à $\Pr[\forall \ell, Z_\ell \notin S]$. Comme les Z_ℓ sont indépendants deux à deux, cette probabilité est égale à :

$$\begin{aligned}\prod_{\ell=1}^{q} \Pr[Z_\ell \notin S] &= \Pr[Z_\ell \notin S]^q = \left(\sum_{Q=1}^{q^3} \Pr[|S| = Q] \cdot \frac{2^n - Q}{2^n}\right)^q \\ &= \left(1 - \frac{\sum_{Q=1}^{q^3} Q \Pr[|S| = Q]}{2^n}\right)^q = \left(1 - \frac{\mathbb{E}(|S|)}{2^n}\right)^q, \qquad \text{(A.1)}\end{aligned}$$

où $\mathbb{E}$ désigne l'espérance d'une variable aléatoire.

Il est facile de montrer que les $A_{i,j,k}$ sont indépendants deux à deux. Pour la suite, on s'inspire d'une preuve donnée par Shoup dans [Sho05], pour tout $(i,j,k) \neq (i',j',k')$, nous introduisons les variables aléatoires intermédiaires $W_{(i,j,k),(i',j',k')}$ qui valent 1 si $A_{(i,j,k)} = A_{(i',j',k')}$ et 0 sinon. On définit également

$$W = 1/2 \cdot \sum_{(i,j,k)\neq(i',j',k')} W_{(i,j,k),(i',j',k')} \geq q^3 - |S|.$$

Prouvons l'inégalité ci-dessus. Pour tout $t \geq 1$, on note par X_t le nombre de t-collisions exactes : on appelle t-collision exacte tout t-uplet de triplets $((i_1, j_1, k_1), \ldots, (i_t, j_t, k_t))$ tel qu'il existe a tel que $a = A_{i_1,j_1,k_1} = \cdots = A_{i_t,j_t,k_t}$ et tel que pour tout (i, j, k) différents des t triplets, $a \neq A_{i,j,k}$. On vérifie facilement que $q^3 = \sum_{t=1}^{q^3} X_t \cdot t$, que $|S| = \sum_{t=1}^{q^3} X_t$ et que $W = \sum_{t=2}^{q^3} X_t \binom{t}{2}$. Comme pour $t \geq 2$, $\binom{t}{2} \geq t - 1$, on a $W \geq \sum_{t=2}^{q^3} X_t(t-1) = q^3 - |S|$, ce qui prouve l'inégalité précédente.

Il est facile de montrer que $W_{(i,j,k),(i',j',k')}$ égal 1 avec probabilité $1/2^n$ et donc que $\mathbb{E}(W) = q^3(q^3-1)/2^{n+1}$. Par conséquent, en utilisant l'équation (A.1),

$$\Pr\left[\forall \ell, Z_\ell \notin S\right] \leq \left(1 - \frac{q^3}{2^n} + \frac{q^3(q^3-1)}{2^{2n+1}}\right)^q \leq \left(1 - \frac{q^3}{2^{n+1}}\right)^q.$$

Comme pour tout $0 \leq X \leq 1$, $1 - X \leq e^{-X}$ et $(1 - 1/e)X \leq 1 - e^{-X}$, si $q^4 \leq 2^n$, nous appliquons la première inégalité à $X = \frac{q^3}{2^{n+1}}$ et la seconde à $X = \frac{q^4}{2^{n+1}}$ et on obtient :

$$\Pr\left[\exists \ell, Z_\ell \in S\right] = 1 - \Pr\left[\forall \ell, Z_\ell \notin S\right] \geq (1 - \frac{1}{e})\frac{q^4}{2^{n+1}}.$$

C'est le résultat que l'on voulait démontrer. □

Bibliographie

[AFP05] M. Abdalla, P.-A. Fouque, and D. Pointcheval. Password-Based Authenticated Key Exchange in the Three-Party Setting. In *PKC 2005*, volume 3386 of *LNCS*, pages 65–84. Springer-Verlag, 2005.

[AP05] M. Abdalla and D. Pointcheval. Simple Password-Based Encrypted Key Exchange Protocols. In *CT-RSA 2005*, volume 3376 of *LNCS*, pages 191–208. Springer-Verlag, 2005.

[BBCM94] Bennett, Brassard, Crepeau, and Maurer. Generalized Privacy Amplification. In *ISIT*, 1994.

[BCJ+05] E. Biham, R. Chen, A. Joux, P. Carribault, C. Lemuet, and W. Jalby. Collisions of SHA-0 and Reduced SHA-1. In *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 36–57. Springer-Verlag, 2005.

[BCK96a] M. Bellare, R. Canetti, and H. Krawczyk. Message Authentication Using Hash Functions : The HMAC Construction. *RSA Laboratories' Cryptobytes*, 2(1), Spring 1996.

[BCK96b] M. Bellare, R. Canetti, and H. Krawczyk. Pseudorandom Functions Revisited : The Cascade Construction and its Concrete Security. In *37th FOCS*, pages 514–523. IEEE Computer Society Press, 1996.

[BCK97] M. Bellare, R. Cannetti, and H. Krawczyk. HMAC : Keyed-Hashing for Message Authentication, 1997. RFC 2104. Disponible à l'adresse `http://www.ietf.org/rfc.html`.

[BCS05] J. Black, M. Cochran, and T. Shrimpton. On the Impossibility of Highly-Efficient Blockcipher-Based Hash Functions. In *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 526–541. Springer-Verlag, 2005.

[BD06] E. Biham and O. Dunkelman. A Framework for Iterative Hash Functions — HAIFA. Présenté au second NIST hash workshop. Disponible à l'adresse `http://www.csrc.nist.gov/pki/HashWorkshop/2006/Papers/DUNKELMAN_NIST3.pdf`, August 2006.

[BDJR97] M. Bellare, A. Desai, E. Jokipii, and P. Rogaway. A Concrete Security Treatment of Symmetric Encryption. In *38th FOCS*, pages 394–403. IEEE Computer Society Press, 1997.

[BDK+05] X. Boyen, Y. Dodis, J. Katz, R. Ostrovsky, and A. Smith. Secure Remote Authentication Using Biometric Data. In *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 147–163. Springer-Verlag, 2005.

[BDPA] G. Bertoni, J. Daemen, M. Peeters, and G. Van Assche. Radiogatùn, a Belt-and-Mill Hash Function. ECRYPT Hash Workshop 2007.

[BDPA08] G. Bertoni, J. Daemen, M. Peeters, and G. Van Assche. On the Indifferentiability of the Sponge Construction. In *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 181–197. Springer-Verlag, 2008.

[Bel06] M. Bellare. New Proofs for NMAC and HMAC : Security Without Collision-Resistance. In *CRYPTO 2006*, volume 4117 of *LNCS*. Springer-Verlag, 2006.

[BGM04] M. Bellare, O. Goldreich, and A. Mityagin. The Power of Verification Queries in Message Authentication and Authenticated Encryption. Eprint cryptology archive 2004/309. Disponible à l'adresse `http://eprint.iacr.org`, 2004.

[BI99] M. Bellare and R. Impagliazzo. A Tool for Obtaining Tighter Security Analyses of Pseudorandom Function Based Constructions, with Applications to PRF→PRP Conversion. Crytology ePrint archive, report 1999/024. Disponible à l'adresse `http://eprint.iacr.org`, 1999.

[BJR+06] J. G. Brainard, A. Juels, R. L. Rivest, M. Szydlo, and M. Yung. Fourth-Factor Authentication : Somebody you Know. In *ACM CCS 2006*, pages 168–178, 2006.

[BK03] J. Bourgain and S. V. Konyagin. Estimates for the Number of Sums and Products and for Exponential Sums Over Subgroups in Fields of Prime Order. *Comptes Rendus Mathématiques*, 337 :75–80, 2003.

[BM92] S. M. Bellovin and M. Merritt. Encrypted Key Exchange : Password-Based Protocols Secure Against Dictionary Attacks. In *1992 IEEE Symposium on Security and Privacy*, pages 72–84. IEEE Computer Society Press, 1992.

[BN00] M. Bellare and C. Namprempre. Authenticated encryption : Relations among notions and analysis of the generic composition paradigm. In *ASIACRYPT 2000*, volume 1976 of *LNCS*, pages 531–545. Springer-Verlag, 2000.

[Boy04] X. Boyen. Reusable Cryptographic Fuzzy Extractors. In *ACM CCS 04*, pages 82–91. ACM Press, 2004.

[BPR00] M. Bellare, D. Pointcheval, and P. Rogaway. Authenticated Key Exchange Secure against Dictionary Attacks. In *EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 139–155. Springer-Verlag, 2000.

[BPR05] M. Bellare, K. Pietrzak, and P. Rogaway. Improved Security Analyses for CBC MACs. In *CRYPTO 2005*, volume 3621 of *LNCS*, pages 527–545. Springer-Verlag, 2005.

[BR93a] M. Bellare and P. Rogaway. Entity Authentication and Key Distribution. In *CRYPTO'93*, volume 773 of *LNCS*, pages 232–249. Springer-Verlag, 1993.

[BR93b] M. Bellare and P. Rogaway. Random Oracles are Practical : A Paradigm for Designing Efficient Protocols. In *ACM CCS 93*, pages 62–73. ACM Press, 1993.

[BRS02] J. Black, P. Rogaway, and T. Shrimpton. Black-Box Analysis of the Block-Cipher-Based Hash-Function Constructions from PGV. In *CRYPTO 2002*, volume 2442 of *LNCS*, pages 320–335. Springer-Verlag, 2002.

[BRW04] M. Bellare, P. Rogaway, and D. Wagner. The EAX Mode of Operation. In *FSE 2004*, volume 3017 of *LNCS*, pages 389–407. Springer-Verlag, 2004.

[BST03] B. Barak, R. Shaltiel, and E. Tromer. True Random Number Generators Secure in a Changing Environment. In *CHES 2003*, volume 2779 of *LNCS*, pages 166–180. Springer-Verlag, 2003.

[CFGP05] O. Chevassut, P. A. Fouque, P. Gaudry, and D. Pointcheval. Key Derivation and Randomness Extraction. Cryptology ePrint Archive, Report 2005/061, 2005. `http://eprint.iacr.org/`.

[CFGP06] O. Chevassut, P. A. Fouque, P. Gaudry, and D. Pointcheval. The Twist-Augmented Technique for Key Exchange. In *PKC 2006*, volume 3958 of *LNCS*. Springer-Verlag, 2006.

[CFK+00] R. Canetti, J. Friedlander, S. Konyagin, M. Larsen, D. Lieman, and I. Shparlinski. On the Statistical Properties of Diffie-Hellman Distributions. *Israel Journal of Mathematics*, 120 :23–46, 2000.

[CFS99] R. Canetti, J. Friedlander, and I. Shparlinski. On Certain Exponential Sums and the Distribution of Diffie-Hellman Triples. *Journal of the London Mathematical Society*, 59(2) :799–812, 1999.

[CK01] R. Canetti and H. Krawczyk. Analysis of Key-Exchange Protocols and Their Use for Building Secure Channels. In *EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 453–474. Springer-Verlag, 2001.

[CK02] R. Canetti and H. Krawczyk. Security Analysis of IKE's Signature-based Key-Exchange Protocol. In *CRYPTO 2002*, volume 2442 of *LNCS*, pages 143–161. Springer-Verlag, 2002. `http://eprint.iacr.org/2002/120/`.

[CPS08] J.-S. Coron, J. Patarin, and Y. Seurin. The Random Oracle Model and the Ideal Cipher Model are Equivalent. In *Crypto 2008*. ACM Press, 2008.

[CS98] R. Cramer and V. Shoup. A Practical Public Key Cryptosystem Provably Secure against Adaptive Chosen Ciphertext Attack. In *CRYPTO '98*, volume 1462 of *LNCS*, pages 13–25. Springer-Verlag, 1998.

[Dam89] I. Damgård. A Design Principle for Hash Functions. In *CRYPTO '89*, volume 435 of *LNCS*, pages 416–427, 1989.

[Dau02] J. Daugman. How Iris Recognition Works. In *ICIP (1)*, pages 33–36, 2002.

[Dau04] J. Daugman. Iris Recognition and Anti-Spoofing Countermeasures. In *7-th International Biometrics Conference*, 2004.

[DGH+04a] Y. Dodis, R. Gennaro, J. Håstad, H. Krawczyk, and T. Rabin. Randomness Extraction and Key Derivation Using the CBC, Cascade and HMAC Modes. In *CRYPTO 2004*, volume 3152 of *LNCS*, pages 494–510. Springer-Verlag, 2004.

[DGH+04b] Y. Dodis, R. Gennaro, J. Håstad, H. Krawczyk, and T. Rabin. Randomness Extraction and Key Derivation Using the CBC, Cascade and HMAC Modes. In *CRYPTO 2004*, volume 3152 of *LNCS*, pages 494–510. Springer-Verlag, 2004.

[DH76] W. Diffie and M. E. Hellman. New Directions in Cryptography. *IEEE Transactions on Information Theory*, 22(6) :644–654, 1976.

[DKRS06] Y. Dodis, J. Katz, L. Reyzin, and A. Smith. Robust Fuzzy Extractors and Authenticated Key Agreement from Close Secrets. In *CRYPTO 2006*, volume 4117 of *LNCS*, pages 232–250. Springer-Verlag, 2006.

[Dod00] Y. Dodis. *Exposure-Resilient Cryptography*. PhD thesis, Massachussetts Institute of Technology (MIT), August 2000.

[DR06] T. Dierks and E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.1, April 2006. *Internet Request for Comment RFC 4346*, Internet Engineering Task Force.

[DR08] T. Dierks and E. Rescorla. The Transport Layer Security (TLS) Protocol version 1.2, March 2008. *Internet Request for Comment RFC 4346*, Internet Engineering Task Force.

[DRS04] Y. Dodis, L. Reyzin, and A. Smith. Fuzzy Extractors : How to Generate Strong Keys from Biometrics and Other Noisy Data. In *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 523–540. Springer-Verlag, 2004.

[Dwo02] NIST Morris Dworkin. Recommendation for Block Cipher Modes of Operation : The CCM Mode for Authentication and Confidentiality, May 2002. NIST Special Publication 800-38C.

[Elg84] T. Elgamal. A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. In *CRYPTO'84*, volume 196 of *LNCS*, pages 10–18. Springer-Verlag, 1984.

[FPSZ06] P.-A. Fouque, D. Pointcheval, J. Stern, and S. Zimmer. Hardness of Distinguishing the MSB or LSB of Secret Keys in Diffie-Hellman Schemes. In *ICALP 2006, Part II*, volume 4052 of *LNCS*, pages 240–251. Springer-Verlag, 2006.

[GÖ5] N. Gürel. Extracting Bits from Coordinates of a Point of an Elliptic Curve. Cryptology ePrint Archive, Report 2005/324. Disponible à `http://eprint.iacr.org/`, 2005.

[GD01] V. D. Gligor and P. Donescu. Fast Encryption and Authentication : XCBC Encryption and XECB Authentication Modes. In *FSE 2001*, volume 2355 of *LNCS*, pages 92–108. Springer-Verlag, 2001.

[GKR04] R. Gennaro, H. Krawczyk, and T. Rabin. Secure Hashed Diffie-Hellman over Non-DDH Groups. In *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 361–381. Springer-Verlag, 2004.

[GM84] S. Goldwasser and S. Micali. Probabilistic Encryption. *Journal of Computer and System Sciences*, 28(2) :270–299, 1984.

[GMR88] S. Goldwasser, S. Micali, and R. L. Rivest. A Digital Signature Scheme Secure Against Adaptive Chosen-message Attacks. *SIAM Journal on Computing*, 17(2) :281–308, April 1988.

[HBK00] D. R. Heath-Brown and S. Konyagin. New Bounds for Gauss Sums Derived from k^{th} Powers, and for Heilbronn's Exponential Sum. *Q. J. Math.*, 51(2) :221–235, 2000.

[HILL99] J. Håstad, R. Impagliazzo, L. Levin, and M. Luby. A Pseudorandom Generator from any One-Way Function. *SIAM Journal of Computing*, 28(4) :1364–1396, 1999.

[HWKS98] C. Hall, D. Wagner, J. Kelsey, and B. Schneier. Building PRFs from PRPs. In H. Krawczyk, editor, *CRYPTO '98*, volume 1462 of *LNCS*, pages 370 – 389. Springer-Verlag, 1998.

[ILL89] I. Impagliazzo, L. Levin, and M. Luby. Pseudo-Random Generation from One-Way Functions. In *Proc. of the 21st STOC*, pages 12–24. ACM Press, New York, 1989.

[JM06] Markus Jakobsson and Steven Myers. *Phishing and Counter-Measures*. J. Wiley and Sons Inc, 2006.

[Jon03] J. Jonsson. On the security of CTR + CBC-MAC. In *SAC 2002*, volume 2595 of *LNCS*, pages 76–93. Springer-Verlag, 2003.

[Jou04] A. Joux. Multicollisions in Iterated Hash Functions. Application to Cascaded Constructions. In *CRYPTO 2004*, volume 3152 of *LNCS*, pages 306–316. Springer-Verlag, 2004.

[JP07] A. Joux and T. Peyrin. Hash Functions and the (Amplified) Boomerang Attack. In *CRYPTO 2007*, volume 4622 of *LNCS*, pages 244–263. Springer-Verlag, 2007.

[Jut01] C. Jutla. Encryption Modes with Almost Free Message Integrity. In *EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 529–544. Springer-Verlag, 2001.

[JW99] A. Juels and M. Wattenberg. A Fuzzy Commitment Scheme. In *ACM CCS 99*, pages 28–36, 1999.

[Ker92] V. Keränen. Abelian Squares are Avoidable on 4 Letters. In *ICALP 1992*, volume 623 of *LNCS*, pages 41–52. Springer-Verlag, 1992.

[Ker03] V. Keränen. On abelian square-free DT0L-languages over 4 letters. In *WORDS 2003*, volume 27, pages 95–109. TUCS General Publication, 2003.

[KK06] J. Kelsey and T. Kohno. Herding Hash Functions and the Nostradamus Attack. In *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 183–200. Springer-Verlag, 2006.

[Kli06] V. Klima. Tunnels in Hash Functions : MD5 Collisions Within a Minute. Cryptology ePrint Archive, Report 2006/105. Disponible à l'adresse `http://eprint.iacr.org/`, 2006.

[Knu05] L. R. Knudsen. SMASH - A Cryptographic Hash Function. In *FSE 2005*, volume 3557 of *LNCS*, pages 228–242. Springer-Verlag, 2005.

[Kra] H. Krawczyk. On Extract-then-Expand Key Derivation Functions and an HMAC-based KDF. Accessible sur la page internet de l'auteur, à l'adresse `http://www.ee.technion.ac.il/~hugo/kdf/`.

[Kra01] H. Krawczyk. The Order of Encryption and Authentication for Protecting Communications (or : How Secure Is SSL ?). In *CRYPTO 2001*, volume 2139 of *LNCS*, pages 310–331. Springer-Verlag, 2001.

[KS99] S. V. Konyagin and I. Shparlinski. *Character Sums With Exponential Functions and Their Applications*. Cambridge University Press, Cambridge, 1999.

[KS05] J. Kelsey and B. Schneier. Second preimages on n-bit hash functions for much less than 2^n work. In *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 474–490. Springer-Verlag, 2005.

[LM92] X. Lai and J. L. Massey. Hash Function Based on Block Ciphers. In *EUROCRYPT'92*, volume 658 of *LNCS*, pages 55–70. Springer-Verlag, 1992.

[LPRR07] M. Lamberger, N. Pramstaller, C. Rechberger, and V. Rijmen. Second Preimages for SMASH. In *CT-RSA 2007*, volume 4377 of *LNCS*, pages 101–111. Springer-Verlag, 2007.

[Luc00] S. Lucks. The Sum of PRPs Is a Secure PRF. In *EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 470–484. Springer-Verlag, 2000.

[Mer89] R. C. Merkle. One Way Hash Functions and DES. In *CRYPTO'89*, volume 435 of *LNCS*, pages 428–446, 1989.

[NZ96] N. Nisan and D. Zuckerman. Randomness is Linear in Space. *J. Comput. Syst. Sci.*, 52(1) :43–52, 1996.

[PGV94] B. Preneel, R. Govaerts, and J. Vandewalle. Hash Functions Based on Block Ciphers : A Synthetic Approach. In *CRYPTO'93*, volume 773 of *LNCS*, pages 368–378. Springer-Verlag, 1994.

[PRR05] N. Pramstaller, C. Rechberger, and V. Rijmen. Breaking a New Hash Function Design Strategy Called SMASH. In *SAC 2005*, volume 3897 of *LNCS*, pages 233–244. Springer-Verlag, 2005.

[PS98] S. Patel and G. Sundaram. An Efficient Discrete Log Pseudo Random Generator. In *CRYPTO'98*, volume 1462 of *LNCS*. Springer-Verlag, 1998.

[RBBK01] P. Rogaway, M. Bellare, J. Black, and T. Krovetz. OCB : A block-cipher mode of operation for efficient authenticated encryption. In *ACM CCS 2001*, pages 196–205. ACM Press, 2001.

[Riv92] R. Rivest. The MD5 Message-Digest Algorithm, april 1992. RFC 1321. Disponible à l'adresse `http://www.ietf.org/rfc.html`.

[Riv05] R. L. Rivest. Abelian square-free dithering for iterated hash functions. *Présenté au ECrypt Hash Function Workshop et au Cryptographic Hash Workshop*, 2005.

[RS92] C. Rackoff and D. R. Simon. Non-Interactive Zero-Knowledge Proof of Knowledge and Chosen Ciphertext Attack. In *CRYPTO'91*, volume 576 of *LNCS*, pages 433–444. Springer-Verlag, 1992.

[RS04] P. Rogaway and T. Shrimpton. Cryptographic Hash-Function Basics : Definitions, Implications, and Separations for Preimage Resistance, Second-Preimage Resistance, and Collision Resistance. In *FSE 2004*, volume 3017 of *LNCS*, pages 371–388, 2004.

[RS08a] P. Rogaway and J. Steinberger. Constructing cryptographic hash functions from fixed-key blockciphers. In *CRYPTO 2008*, volume 5157 of *LNCS*, pages 433–450. Springer-Verlag, 2008.

[RS08b] P. Rogaway and J. Steinberger. Security / Efficiency Tradeoffs for Permutation-Based Hashing. In *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 220–236. Springer-Verlag, 2008.

[RSA78] R. L. Rivest, A. Shamir, and L. M. Adleman. A Method for Obtaining Digital Signature and Public-Key Cryptosystems. *Communications of the Association for Computing Machinery*, 21(2) :120–126, 1978.

[RTS97] J. Radhakrishnan and A. Ta-Shma. Tight Bounds for Depth-two Superconcentrators. In *38th FOCS*, pages 585–594. IEEE Computer Society Press, 1997.

[RW03] P. Rogaway and D. Wagner. A Critique of CCM, 2003. Eprint cryptology archive 2003/070. Disponible à l'adresse `http://eprint.iacr.org`.

[SHA95] Secure Hash Standard. National Institute of Standards and Technology, NIST FIPS PUB 180-1, U.S. Department of Commerce, April 1995.

[Sha02] R. Shaltiel. Recent Developments in Explicit Constructions of Extractors. *Bulletin of the EATCS*, 77 :67–95, 2002.

[Sho00a] V. Shoup. A Composition Theorem for Universal One-Way Hash Functions. In *EUROCRYPT 2000*, pages 445–452, 2000.

[Sho00b] V. Shoup. Using Hash Functions as a Hedge against Chosen Ciphertext Attack. In *EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 275–288. Springer-Verlag, 2000.

[Sho01a] V. Shoup. A Proposal for an ISO Standard for Public-Key Encryption, december 2001. ISO/IEC JTC 1/SC27.

[Sho01b] V. Shoup. OAEP Reconsidered. In *Crypto 2001*, volume 2139 of *LNCS*, pages 239–259. Springer-Verlag, 2001.

[Sho04] V. Shoup. Sequences of Games : a Tool for Taming Complexity in Security Proofs. Cryptology ePrint Archive, Report 2004/332. Disponible à l'adresse `http://eprint.iacr.org/`, 2004.

[Sho05] V. Shoup. *A Computational Introduction to Number Theorie and Algebra*. Cambridge University Press, 2005.

[SPC04] NIST Special Publication 800-38C. Recommendation for Block Cipher Modes of Operation : The CCM Mode for Athentication and Confidentiality, May 2004. Disponible à l'adresse `http://csrc.nist.gov/CryptoTollkit/modes/`.

[SS08] T. Shrimpton and M. Stam. Building a Collision-Resistant Compression Function from Non-Compressing Primitives. In *ICALP 2008*, volume 5125 of *LNCS*. Springer-Verlag, 2008.

[Tho05] S. S. Thomsen. *Cryptographic Hash Functions*. PhD thesis, Technical University of Denmark, 2005.

[TV00] L. Trevisan and S. P. Vadhan. Extracting Randomness from Samplable Distributions. In *41st FOCS*, pages 32–42. IEEE Computer Society Press, 2000.

[Val02] V. Valencia. *Biometric Liveness Testing*, chapter 8. Osborne McGraw Hill, 2002.

[vOW96] P. C. van Oorschot and M. J. Wiener. On Diffie-Hellman Key Agreement with Short Exponents. In *EUROCRYPT 1996*, volume 1070 of *LNCS*, pages 332–343. Springer-Verlag, 1996.

[Wag02] D. Wagner. A Generalized Birthday Problem. In *CRYPTO 2002*, volume 2442 of *LNCS*, pages 288–303. Springer-Verlag, 2002.

[WHF02] D. Whiting, R. Housley, and N. Ferguson. IEEE 802.11-02/001r2 : AES Encryption and Authentication Using CTR Mode and CBC-MAC, March 2002.

[WLF+05] X. Wang, X. Lai, D. Feng, H. Chen, and X. Yu. Cryptanalysis of the Hash Functions MD4 and RIPEMD. In *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 1–18, 2005.

[WY05] X. Wang and H. Yu. How to Break MD5 and Other Hash Functions. In *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 19–35, 2005.

[WYY05a] X. Wang, Y. Lisa Yin, and H. Yu. Finding Collisions in the Full SHA-1. In *CRYPTO 2005*, volume 3621 of *LNCS*, pages 17–36, 2005.

[WYY05b] X. Wang, H. Yu, and Y. Lisa Yin. Efficient Collision Search Attacks on SHA-0. In *CRYPTO 2005*, volume 3621 of *LNCS*, pages 1–16, 2005.

Bibliographie personnelle

[ABF+08] E. Andreeva, C. Bouillaguet, P.-A. Fouque, J. J. Hoch, J. Kelsey, A. Shamir, and S. Zimmer. Second Preimage Attacks on Dithered Hash Functions. In *EUROCRYPT '08*, volume 4965 of *LNCS*, pages 270–288. Springer-Verlag, 2008.

[BCI+07] J. Bringer, H. Chabanne, M. Izabachene, D. Pointcheval, T. Qiang, and S. Zimmer. An Application of the Goldwasser-Micali Cryptosystem to Biometric Authentication. In *ACISP '07*, volume 4586 of *LNCS*, pages 96–106. Springer-Verlag, 2007.

[BCPZ] J. Bringer, H. Chabanne, D. Pointcheval, and S. Zimmer. Another Look at the Generation of Strong Keys From Biometric Data. Accepted at IWSEC2008.

[BDFZ] C. Bouillaguet, O. Dunkelmann, P.-A. Fouque, and S. Zimmer. Second Preimage Resistance of Some Iterated Hash Functions. En cours de soumission.

[BCC+07] E. Bresson, B. Chevallier-Mames, C. Clavier, B. Debraize, P.-A. Fouque, L. Goubin, A. Gouget, G. Leurent, P. Q. Nguyen, P. Paillier, T. Peyrin, and S. Zimmer. Revisiting Security Relations Between Signature Schemes and Their Inner Hash Functions. ECRYPT Hash Workshop, 2007.

[CBPZ07] H. Chabanne, J. Bringer, D. Pointcheval, and S. Zimmer. Génération et utilisation d'une clef biométrique. Demande brevet en France FR0760311, décembre 2007.

[CFPZ] C. Chevalier, P.-A. Fouque, D. Pointcheval, and S. Zimmer. Optimal Randomness Extraction From a Diffie-Hellman Element. En cours de soumission.

[FMVZ08] P.-A. Fouque, G. Martinet, F. Valette, and S. Zimmer. On the Security of the CCM Encryption Mode and of a Slight Variant. In *ACNS '08*, volume 5037 of *LNCS*, pages 411–428. Springer-Verlag, 2008.

[FPSZ06] P.-A. Fouque, D. Pointcheval, J. Stern, and S. Zimmer. Hardness of Distinguishing the MSB or LSB of Secret Keys in Diffie-Hellman Schemes. In *ICALP '06, Part II*, volume 4052 of *LNCS*, pages 240–251. Springer-Verlag, 2006.

[FPZ08] P.-A. Fouque, D. Pointcheval, and S. Zimmer. Hmac is a Randomness Extractor and Applications to TLS. In *ASIACCS '08*, pages 21–32. ACM, 2008.

[FSZ08] P.-A. Fouque, J. Stern, and S. Zimmer. Two-Permutation Collision-Resistant Hash Function. In *SAC '08*, LNCS. Springer-Verlag, 2008.

[PZ08] D. Pointcheval and S. Zimmer. Multi-Factor Authenticated Key Exchange. In *ACNS '08*, volume 5037 of *LNCS*, pages 277–295. Springer-Verlag, 2008.

Résumé

Établir un canal garantissant l'authentification de façon efficace requiert l'utilisation de nombreux outils cryptographiques. Dans ce mémoire, nous nous intéressons à la sécurité de plusieurs d'entre eux, présents à différentes étapes dans le protocole de mise en place du canal.

Dans un premier temps, nous abordons l'analyse de deux protocoles qui, mis bout à bout, assurent la mise en place d'un canal authentifié, confidentiel et intègre : un algorithme d'échange de clefs authentifié et un algorithme de chiffrement authentifié. Le premier algorithme permet de générer des clefs en alliant plusieurs moyens d'authentification (biométrie, mot de passe, carte à puce). Le second est un algorithme normalisé appelé CCM.

Dans un second temps, nous nous intéressons plus particulièrement à la phase d'extraction de clef, étape charnière entre l'échange de clef et son utilisation pour établir un canal sécurisé. Nous présentons une méthode simple pour extraire une clef symétrique d'un élément Diffie-Hellman, puis nous analysons l'utilisation de HMAC comme fonction d'extraction de clef.

Dans un troisième temps, nous concentrons notre attention sur les fonctions de hachage, très utilisées à plusieurs niveaux du protocole. Plus précisément, nous analysons la sécurité d'un mode opératoire basé sur un algorithme de chiffrement par bloc dont on fixe la clef, puis, nous examinons des modes opératoires qui cherchent à garantir une sécurité en seconde préimage optimale.

Mots-clés: Cryptologie, preuve de sécurité, authentification, échange de clefs, extraction d'entropie, mode opératoire, fonction de hachage.

Abstract

Several cryptographic tools are required to establish an authenticated channel. In this thesis, we are interested in the security of some of them, used at different levels of the protocol which establishes the channel.

In the first part, we analyze two algorithms which, used together, establish a channel which guarantees privacy, authentication and integrity : an authenticated key exchange protocol and an authenticated encryption scheme. The first one allows to generate keys using several authentication factors (biometrics, password, secure device). The second one is a standardized algorithm call CCM.

In the second part, we are interested more particularly by the key extraction phase, phase which occurs between key exchange and the establishment of the secure channel. We present a simple method to extract a symetric key from a Diffie-Hellman element, and then analyse HMAC as a computationnal randomness extractor.

In the third part, we analyse some hash function operating modes. More precisely, we examin the security of a fixed-key blockcipher based operating mode and then study operating modes which tries to guarantee an optimal second preimage resistance.

Keywords: Cryptology, security proof, authentication, key exchange, key extraction, operating mode, hash function.