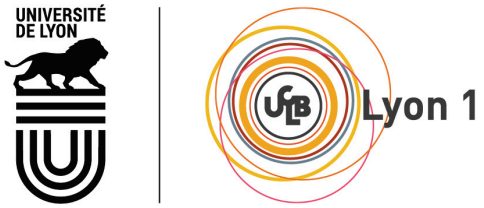
Méthodes de calculs sur les données chiffrées

Marie Paindavoine



N° d'ordre NNT : 2017LYSE1009

THÈSE DE DOCTORAT DE L'UNIVERSITÉ DE LYON
opérée au sein de
l'Université Claude Bernard Lyon 1

École Doctorale ED512
Informatique et Mathématiques

Spécialité de doctorat : informatique

Soutenue publiquement le 27/01/2017, par :
Marie Paindavoine

Méthodes de calculs sur les données chiffrées

Devant le jury composé de :

Mme Minier Marine, Professeure, Université de Lorraine	Présidente
Mme Fontaine Caroline, Chargée de recherche, CNRS, Telecom Bretagne	Rapporteure
M. Phan Duong Hieu, Professeur, Université de Limoges	Rapporteur
Mme Chevalier Céline, Maître de conférences, Université Paris 2	Examinatrice
M. Muntean Traian, Professeur, Université Aix-Marseille	Examinateur
Mme Önen Melek, Ingénieure de Recherche, Eurecom	Examinatrice
M. Laguillaumie Fabien, Professeur, Université Lyon 1	Directeur de thèse
M. Canard Sébastien, Ingénieur de recherche, Orange Labs	Co-directeur de thèse

Travaux effectués au sein de l'Applied Crypto Group, Orange Labs, et du Laboratoire d'Informatique et du Parallélisme, U. Lyon, CNRS, INRIA, UCBL, ENS Lyon.

Remerciements

Cette thèse est loin d'avoir été un travail solitaire. J'ai eu la chance de recevoir du soutien de nombreuses personnes, et ces quelques lignes ont du mal à refléter ce que je leur dois. La langue française, si riche par ailleurs, manque cruellement de mots lorsqu'il s'agit de dire quelque chose d'aussi simple et important que "merci".

Tout d'abord, j'aimerais remercier mes directeurs de thèse, Sébastien Canard et Fabien Laguillaumie. Je leur suis très reconnaissante pour leur encadrement tout au long de ces trois années, pour leur disponibilité et leur expertise. Sébastien, merci pour m'avoir beaucoup apporté, avec l'enthousiasme qui te caractérise quand il s'agit de découvrir de nouveaux domaines. Fabien, merci pour ton exigence et ta rigueur, notamment au niveau des preuves de sécurité. Même si j'ai encore beaucoup à apprendre, tes remarques toujours pertinentes m'ont permis de m'améliorer grandement.

Un immense merci à Caroline Fontaine et à Phan Duong Hieu d'avoir accepté de rapporter mes travaux, malgré un planning malheureux entre congés de Noël et organisation d'Asiacrypt pour Hieu. Leurs commentaires ont grandement contribué à améliorer la qualité de ce manuscrit. Merci à Céline Chevalier, Marine Minier, Traian Muntean et Melek Önen de compléter mon jury de thèse.

Jacques Fournier et Ronan Lashermes, merci de la confiance que vous m'avez accordée quand vous m'avez acceptée en stage. Ces quelques mois à Gardanne m'ont permis de découvrir le monde de la recherche et donné le bagage nécessaire pour affronter (presque) sereinement ce travail de thèse.

En plus des noms déjà cités, j'ai eu la chance de collaborer avec de nombreuses personnes, que je remercie chaleureusement : Aïda Diop, Nadia El Mrabet, Louis Goubin, Nizar Kheir, Mohamed Sabt, Bastien Vialla. Ces collaborations m'ont énormément appris, techniquement bien sûr, mais aussi sur le travail de recherche et ont contribué à façonner ma propre manière de travailler.

Merci à l'équipe NPS (Networks and Products Security) de m'avoir accueillie pour ces travaux, et pour le début de ma carrière de chercheuse ! Cette équipe ne serait pas la même sans Jean-François, et l'énergie qu'il met à en faire un espace convivial et chaleureux. Un immense merci aux membres de l'équipe, avec qui j'ai toujours pris grand plaisir à échanger, et qui m'ont permis d'élargir ma vision de la cybersécurité.

J'aimerais particulièrement remercier Olivier, avec qui j'ai partagé un bureau pendant mes deux premières années. Ses talents affûtés en cryptanalyse m'ont permis de ne pas perdre trop de temps sur des idées bancales. Merci également à mon co-bureau de dernière année, Nicolas, à la source d'innombrables débats sur l'efficacité d'un protocole cryptographique.

Mais parce que la vie, même au bureau, n'est pas qu'une suite de discussions techniques, merci à notre fabuleux équipage O'Sec, avec lequel nous avons partagé l'Armor Cup, la régate interne d'Orange. Ça tartine ! Julien, ta patience et pédagogie ont été déterminantes pour la réussite de cette aventure, surtout lorsque nous répétions encore, et toujours les même erreurs. Olivier, nous n'aurions jamais passé une aussi bonne régate si tu n'avais pas pris la lourde responsabilité de l'organisation. Erwan, Mohamed, Thibault, Xavier, merci d'avoir complété si efficacement notre équipage.

N'oublions pas les anciens doctorants et post-doctorants de l'équipe, partis vers d'autres ho-

rizons : Amira, Baptiste, Ghada, Mouhannad, merci pour tous les moments qu'on a pu partager, autour de problèmes techniques, d'une pause ou d'une randonnée dans les magnifiques paysages normands.

Pour des raisons pratiques, je n'ai pu me rendre que trop peu fréquemment au LIP, siège de ma deuxième équipe. Que ce soit à la journées des doctorants du LIP ou aux journées AriC, mes passages à Lyon furent courts mais toujours riches de rencontres avec des chercheurs brillants.

Un immense merci aux assistant-es de l'équipe, et notamment à Damien, Chiraz, Évelyne et Marie, qui m'ont à plusieurs reprises guidée dans le labyrinthe administratif - à distance ! - et facilitée toutes les démarches.

Mayla, Mohamed, nous avons commencé (et achevé !) notre thèse ensemble. Partager ce voyage, avec ses hauts et ses bas, nous a fait passer du statut de collègues à celui d'amis chers. Merci pour votre aide, vos conversations, vos éclats de rire : votre amitié a été un soutien précieux. Cette thèse n'aurait pas été la même sans vous, et j'espère que nous aurons l'occasion de partager d'autres aventures ensemble.

Enfin, ma gratitude va à ma famille, et particulièrement mes parents, mon frère - avec, évidemment, sa femme et sa fille - et Cécile, dont l'amitié remonte à si longtemps qu'elle a toute sa place ici. Vos encouragements, vos conseils et votre soutien ont été déterminants pour la réussite de cette thèse et je vous en suis profondément reconnaissante.

Table des matières

Chapitre 1

Introduction

Sommaire

Si la cryptographie peut avoir comme un air d'espionnage aux yeux d'un non-initié, celle-ci ne se limite pas à la protection des secrets d'État. Elle est aujourd'hui omniprésente dans notre vie quotidienne. Elle se trouve notamment au sein de nos téléphones, de nos cartes bancaires, de nos ordinateurs, afin de protéger nos données personnelles ainsi que les données confidentielles des entreprises pour lesquelles nous travaillons.

Assurer le secret des communications est un besoin ancien, et des formes de cryptographies se retrouvent chez beaucoup de civilisations anciennes : Égypte, Mésopotamie, Grèce antique [Kah96]... D'innombrables protocoles de chiffrement - moyens plus ou moins élaborés de rendre un message inintelligible, sauf pour son destinataire légitime - ont ensuite vu le jour. En parallèle, la cryptanalyse, l'art de lire des messages chiffrés sans la clé secrète, a aussi pris son envol. La quasi-totalité des premiers protocoles de chiffrement a été cassée, à l'exception notable du chiffre de Vernam, que sa complexité rend cependant délicat à utiliser. Cette bataille entre concepteurs de schémas et attaquants n'aurait pu rester qu'un habile concours d'ingéniosité. Mais un nouvel élément est entré en jeu, transformant profondément la cryptologie. L'avènement de l'ordinateur, d'abord allié de poids des cryptanalystes anglais lors de la seconde guerre mondiale, a par la suite permis la conception de protocoles résistant durablement aux attaques.

1.1 Cryptographie et fonctionnalité

Les premières tentatives de cryptographie avaient principalement comme objectif de garantir la *confidentialité* des messages. Ce but était supposé atteint grâce à des jeux de substitutions et de permutations ; de l'élémentaire chiffre de César à des machineries raffinées, dont la plus célèbre est probablement Enigma. Les premiers schémas de cryptographie requéraient que l'émetteur et le destinataire du message s'accordent sur un *algorithme* de chiffrement. Ce mode de fonctionnement est critiqué par Kerckhoffs dès la fin du XIX^{e} siècle. En effet, selon lui, le secret de l'algorithme de chiffrement ne suffit pas à garantir le secret de la communication. Un premier pivot théorique s'opère alors : le seul élément qui doit demeurer secret est la clé de l'échange, l'algorithme pouvant être connu de tous. Ce pivot est un élément majeur dans le déploiement de la cryptographie à grande échelle, notamment dans le monde des télécommunications. Face à des réseaux toujours plus étendus et plus denses, pouvoir utiliser le même algorithme permet d'assurer des transmissions sans accrocs. Ce pivot permet également la conception de standards mondiaux, publics et étudiés attentivement par la communauté.

De plus, la numérisation a soulevé de nouveaux défis pour la cryptographie : il faut, en plus de la confidentialité, garantir l'authenticité et l'intégrité des échanges. Le problème ne se soulevait en effet pas de manière si aiguë lorsque les communications se font sur support physique. L'authenticité d'un message est garantie par une signature. Or, contrefaire une signature écrite reste plus compliqué que de copier-coller une signature numérique apposée à la suite d'un document. De même, modifier, de manière imperceptible, un message écrit sur une tablette ou du papier est une tâche bien plus ardue que d'effacer quelques octets d'informations sur une transmission numérique. Ainsi, en plus du chiffrement, la communauté cryptographique a développé de nouveaux outils pour répondre à ces questions nées du numérique.

Les nombreux standards de chiffrement, de signature et d'authentification de messages, tous validés par la communauté scientifique, pourraient donner l'impression que la cryptographie a parfaitement réussi sa transition entre l'ère artisanale et l'ère numérique. Néanmoins, la transmission de l'information est désormais chose si aisée, de même que son stockage, dans des proportions de plus en plus importantes, que l'on voit émerger de nouveaux enjeux liés à l'information. La question n'est plus seulement de transmettre à grande échelle, mais aussi d'exploiter cette masse d'information créée par internet afin de la rendre significative.

L'analyse de masses de données est non seulement devenue une pratique omniprésente dans ce qu'il est convenu d'appeler le "monde connecté", mais aussi une part intégrante du modèle économique de celui-ci. Elle est censée permettre de prévoir la météo, la production agricole, les comportements des consommateurs, les fluctuations des marchés ou encore le prochain président... La place de ces analyses dans notre quotidien, et surtout dans le quotidien des décideurs du monde économique, vient du fait qu'elle permet de réduire toute incertitude liée au processus de décision : elle garantit des choix optimaux en terme de confort et de profit.

En particulier, l'analyse des comportements des utilisateurs de services internet est cruciale pour les fournisseurs. Cela leur fournit des éléments pour optimiser, sécuriser et rentabiliser leurs services. Par exemple, un fournisseur de stockage externe a intérêt à vérifier que les données stockées par les utilisateurs ne contiennent pas de duplicata. Ces services étant particulièrement populaires pour stocker des médias, comme des vidéos, s'assurer de ne pas garder en mémoire un très grand nombre d'exemplaires de la dernière série à la mode permet de diminuer les coûts de la plateforme. D'autres services sont intrinsèquement appelés à inspecter les données des utilisateurs. Par exemple, un fournisseur de logiciel de contrôle parental doit, pour assurer la sécurité des plus jeunes, analyser le trafic internet émis et reçu par un ordinateur. Enfin, ces analyses ont une valeur économique importante pour de nombreux fournisseurs de services gratuits, services pour lesquels les utilisateurs ne sont pas prêts à payer, mais auxquels ils ne sont pas prêts non plus à renoncer. Les GAFAM (Google, Apple, Facebook, Amazon, Microsoft) tirent une part non-négligeable (voire, pour certains, la quasi totalité) de leurs revenus de l'analyse des comportements des utilisateurs. Cela leur permet, par exemple, de cibler les publicités afin d'augmenter le nombre de vues de ces dernières, et donc leur rentabilité. Les utilisateurs profitent également de l'exploitation de leurs données personnelles. Par exemple, la géolocalisation des mobiles permet à des fournisseurs de services GPS de calculer le chemin le plus rapide en fonction du trafic routier en temps réel. Néanmoins, ces données ont beau paraître dématérialisées, leur caractère profondément personnel et potentiellement sensible est indéniable.

"Quand c'est gratuit, c'est vous le produit à vendre[1]". La prise de conscience autour de l'utilisation de nos données personnelles par les acteurs d'internet est croissante [NTI16]. En effet, nos données de navigation ou d'utilisation de nos smartphones révèlent une grande quantité d'information sur nous : notre lieu de domicile, de travail, nos trajets quotidiens, notre condition médicale [JP16]... Une première solution à ce problème vient d'algorithmes d'anonymisation, dont le but est de préserver la valeur monétaire des informations collectées auprès des utilisateurs, sans révéler leur vie privée. En effet, des données correctement anonymisées ne compromettent pas la vie privée des utilisateurs, tout en étant source d'informations significatives.

1. "If you are not paying for it, you're not the customer ; you're the product being sold", commentaire d'un certain Andrew Levis ayant eu un certain succès [Pre]...

Par exemple, dans le challenge D4D (data for development) lancé par Orange [Ora], des comptes rendus d'appels sénagalais anonymisés sont mis à la disposition d'universitaires. L'analyse des mobilités des personnes a ainsi permis de mieux comprendre des évolutions épidémiologiques comme celle du paludisme, d'Ebola ou de la schistosomiase. D'autres résultats ont permis d'identifier les régions en fort besoin de couverture énergétique, ou encore les axes de transport dont l'utilisation va croître, permettant un déploiement optimisé, en prise avec les réalités du terrain. Effectuée correctement pour éviter la ré-indentification d'individus précis au sein d'une base de données [NS09], l'anonymisation pourrait donc apparaître comme une première solution en termes de compromis entre vie privée et analyse des données. Dans ce contexte, Apple a récemment annoncé le déploiement de la *differential privacy* à l'ensemble des données remontées par les utilisateurs [App16].

Néanmoins, l'anonymisation ne répond qu'à la question de l'exploitation de données agrégées. Pour des données individuelles, comme le trafic émis et reçu par un seul ordinateur, seul le chiffrement apparaît comme une réponse adaptée pour en préserver la confidentialité. Or, chiffrer ces données empêche leur analyse. Le chiffrement priverait alors les utilisateurs d'un grand nombre de services auxquels ils sont désormais habitués. Le simple partage d'un fichier entre différentes personnes, si celui-ci est chiffré, devient un problème. Les enjeux de la cryptographie se transforment ainsi au fur et à mesure de l'évolution des communications. Il ne suffit plus d'assurer la confidentialité, l'intégrité et l'authenticité d'un message. Il faudrait aussi pouvoir exploiter les données chiffrées afin de maintenir un niveau de service et d'ergonomie, tout en respectant la vie privée des utilisateurs. C'est le problème auquel nous nous intéressons dans ce manuscrit.

Calcul sur les données chiffrées. Dans ce contexte, l'idéal serait de pouvoir calculer sur des données chiffrées sans rien apprendre des données claires sous-jacentes. Cette possibilité ouvre un monde quasiment magique, où il n'est plus nécessaire de *voir* l'information pour l'analyser. Cela permettrait de déléguer des calculs sur des données sensibles, sans avoir à nécessairement faire confiance à la personne opérant ces calculs. Cela permettrait de garder privées nos données de géolocalisation sans renoncer à un service GPS efficace. Dans ce monde, même recevoir une publicité ciblée ne serait plus aussi dérangeant : on aurait la garantie que les informations ayant permis de nous cibler restent privées. Ici, le mot calcul est pris au sens large du terme. Il peut s'agir d'effectuer une opération sur des données chiffrées, mais aussi simplement de rechercher de l'information parmi un flux chiffré. Plusieurs pistes de recherche ont été explorées pour réaliser ces calculs.

La première est le chiffrement homomorphe. Cet adjectif est dérivé du grec et signifie "même forme". Ce type de chiffrement permet de transposer la structure des messages clairs dans le monde chiffré. Ainsi, une opération sur les chiffrés se traduira par une opération sur les clairs. Il devient possible d'obtenir le chiffré du résultat d'une multiplication ou d'une addition, uniquement à partir des chiffrés des opérandes. Dans ce schéma, n'importe qui peut évaluer n'importe quel polynôme, mais le résultat du calcul reste chiffré. Introduit pour la première fois en 1978 [RAD78], un tel schéma de chiffrement est resté un problème ouvert pendant une trentaine d'années. De nombreux systèmes de chiffrement permettent en effet d'effectuer soit des additions [Pai99], soit des multiplications sur les chiffrés [RSA78, Gam84], mais un protocole permettant un enchaînement arbitraire de ces deux opérations n'est proposé qu'en 2009, par Gentry [Gen09b]. Cette percée dans le domaine des calculs chiffrés souffre toutefois d'un problème d'efficacité, malgré les nombreuses améliorations qui lui ont été apportées.

Une deuxième piste est la proposition de schémas rendant possible un traitement spécifique des données. Le chiffrement cherchable [BCOP04], par exemple, permet d'effectuer des recherches sur des données protégées. Un logiciel de contrôle parental pourrait, par exemple, utiliser ce type de schéma pour détecter des URL préalablement identifiées comme inappropriées pour le jeune public.

D'autres méthodes existent mais ne seront pas développées dans ce mémoire. En 1982,

Yao [Yao82] propose un problème très simple connu sous le nom de problèmes des millionnaires : est-ce que deux millionnaires peuvent déterminer lequel des deux est le plus riche sans avoir à se dévoiler leurs fortunes respectives ? Ce problème a ouvert la voie du calcul multi-parties, où plusieurs acteurs effectuent des calculs sur l'ensemble de leurs entrées, sans qu'aucune partie ne révèle aux autres ses données propres. Les protocoles de calcul multi-parties, néanmoins, ne fonctionnent que de manière interactive. Enfin, le chiffrement fonctionnel [BSW11] répond en quelque sorte au problème miroir du chiffrement homomorphe : l'évaluation d'une fonction nécessite une clé émise par le propriétaire des données mais le résultat est en clair. Toutefois, cette dernière catégorie de schémas, bien qu'extrêmement intéressante en terme de cas d'usage, reste inefficace pour la plupart des fonctions.

1.2 Nos contributions

Nous nous intéressons dans ce manuscrit aux solutions permettant de préserver la vie privée des utilisateurs tout en maintenant les fonctionnalités de services internet.

Amélioration de l'efficacité des calculs génériques. Si l'efficacité du chiffrement homomorphe ne cesse de s'améliorer, des points bloquants demeurent. Parmi ceux-ci se trouve la procédure de réamorçage, ou *bootstrapping*, très coûteuse. Celle-ci est cependant nécessaire pour obtenir du chiffrement totalement homomorphe, c'est-à-dire un système de chiffrement permettant des calculs arbitraires sur des données chiffrées. En minimisant le nombre d'appels à cette procédure, nous proposons une manière efficace de réduire le temps de calcul homomorphe. Nous prouvons que minimiser le nombre de réamorçages est un problème NP-complet et présentons un solveur qui obtient une bonne estimation du minimum de réamorçages nécessaires, ainsi que leur placement dans les calculs. Ces résultats ont fait l'objet de l'article, *Minimizing the Number of Bootstrappings in Fully Homomorphic Encryption* présenté à la conférence SAC 2015 [PV15].

Déduplication vérifiable. La déduplication des données consiste pour un fournisseur de stockage externe à ne conserver qu'un seul exemplaire de chaque fichier, quel que soit le nombre d'utilisateurs l'ayant téléversé sur la plateforme. Cette pratique est très répandue car elle permet d'économiser de l'espace de stockage. Encore une fois, un schéma de chiffrement classique, offrant l'indistinguabilité des messages chiffrés, empêcherait le fournisseur de stockage de dédupliquer les données qu'il héberge. De nombreux systèmes de chiffrement, rassemblés sous le nom de chiffrement verrouillé par le message, ont ainsi été proposés afin de pouvoir faire de la déduplication sur données chiffrées. Néanmoins, dans ces schémas, il est extrêmement facile pour un utilisateur d'empêcher un serveur de mener à bien le processus de déduplication, empêchant ainsi le schéma de chiffrement de remplir la fonctionnalité pour lequel il a été conçu. C'est pourquoi nous proposons un schéma de chiffrement verrouillé par le message vérifiable. Avec notre protocole, le serveur peut s'assurer que l'utilisateur a correctement chiffré ses messages, et donc que le processus de déduplication ne peut être évité. Ces résultats ont été présentés dans l'article *Verifiable Message-Locked Encryption*, à la conférence CANS 2016 [CLP16].

Détection d'intrusions sur du trafic chiffré. Le chiffrement du trafic internet est aujourd'hui indispensable pour préserver la confidentialité lors d'échanges impliquant des données personnelles sensibles ou encore des données confidentielles d'entreprises. Néanmoins, et peut-être aussi contre-intuitivement, celui-ci pose également des problèmes de sécurité. Le rapport de Dell Security 2016 [Del16] pointe l'augmentation des intrusions qui profitent des communications chiffrées. En effet, le standard HTTPS, aujourd'hui largement répandu, aveugle les solutions de sécurité chargées d'inspecter le trafic. Celles-ci deviennent incapables de distinguer entre du trafic sain, légitime, et du trafic malveillant, résultant d'une attaque. Nous proposons de nouvelles techniques de chiffrement de canal de communications permettant la détection d'intrusions sur le

réseau. Notre première solution, basée sur du chiffrement cherchable, améliore l'état de l'art de la détection d'intrusions sur du trafic chiffré, tant en terme de performances que de fonctionnalités. Notre seconde solution, qui utilise du chiffrement totalement homomorphe, permet une détection des intrusions équivalente à ce qui se fait aujourd'hui sur du trafic en clair, mais avec des performances amoindries par rapport aux protocoles existants. Ces résultats ont fait l'objet de deux brevets [CDKP16b, CDKP16a] et de l'article *BlindIDS : Market-Compliant, Privacy-Friendly and Security-Aware Intrusion Detection Systems over Encrypted Traffic*, publié à AsiaCCS [CDK+17].

1.3 Structure du manuscrit

Ce manuscrit est divisé en trois parties. La première partie du mémoire est consacrée à des rappels informatiques (chapitre 2) et cryptographiques (chapitre 3).

La deuxième partie de ce mémoire est consacrée au chiffrement totalement homomorphe, qui offre la possibilité de faire des calculs génériques sur les données chiffrées. Bien que cette primitive soit jeune, nombreux sont les schémas qui ont été proposés. Au chapitre 4, nous présentons ces différents schémas et les défis inhérents au calcul sur les données chiffrées. Ces défis sont principalement algorithmiques : calculer sur les données chiffrées suppose de ne pouvoir exploiter aucune information sur la structure de ces données, modifiant ainsi profondément la manière de concevoir un algorithme efficace pour atteindre un objectif donné. Ensuite, nous présentons au chapitre 5 notre première contribution, un modèle qui permet de placer au mieux les réamorçages lors de l'évaluation homomorphe.

La troisième partie de ce mémoire est consacrée à la conception de schémas *ad hoc*, qui visent à rendre possible une fonctionnalité donnée sur les messages chiffrés. Le premier cas d'usage traité au chapitre 6 est la recherche de duplicatas parmi les fichiers d'un serveur de stockage externe. Le deuxième cas d'usage est la détection d'intrusions sur du trafic chiffré. Pour ce cas, le chiffrement cherchable offre une solution satisfaisante, bien que seul le chiffrement totalement homomorphe permette de traiter le cas complet. Ce travail est détaillé au chapitre 7.

Première partie

Préliminaires

Chapitre 2

Outils informatiques

Sommaire

Dans ce chapitre, nous introduisons les outils issus de l'informatique théorique nécessaires à la compréhension des chapitres suivants. Nous présentons tout d'abord la théorie de la complexité. Ensuite, nous rappelons quelques éléments de la théorie des graphes. Enfin, nous présentons rapidement la théorie des langages.

2.1 Théorie de la complexité

Dans ce chapitre, nous exposons les idées nécessaires à la compréhension de la suite de ce manuscrit, et nous renvoyons le lecteur à [AB09] pour plus de précisions. La complexité algorithmique vise à évaluer la difficulté à résoudre un problème. Nous nous concentrons sur la complexité en temps, c'est-à-dire le nombre d'étapes nécessaire, en fonction de la taille des entrées d'un problème, pour qu'un algorithme puisse le résoudre. Ce nombre d'étapes diffère selon les capacités de l'algorithme considéré. Il est donc nécessaire de formaliser celles-ci : c'est ce qu'on appelle le modèle de calcul. Nous utilisons le modèle standard de la *machine de Turing*.

Machine de Turing déterministe. Une machine de Turing est la donnée d'un quintuplet (Σ, Q, T, q_0, F) où :

- Σ est un ensemble de caractères, auquel s'ajoute des caractères spéciaux comme le caractère blanc, formant l'alphabet de travail Σ_+ ;
- Q est l'ensemble des états de la machine ;
- $T : \Sigma_+ \times Q \to \Sigma_+ \times Q \times \{G, D\}$ est la fonction de transition qui à chaque couple (caractère, état) inscrit un caractère dans la case, change éventuellement d'état et se déplace vers la gauche ou la droite ;
- q_0 est l'état initial de la machine ;
- $F \subset Q$ est l'ensemble des états finaux de la machine.

Pour mesurer la complexité d'un algorithme résolvant un problème, on utilise la notation de Landau "grand O".

Définition 2.1 (Notation de Landau)**.** *Soit f et g deux fonctions. La fonction g a une complexité*

en temps en $O(f)$ si et seulement si :

$$\exists k \in \mathbb{R},\ \exists c \in \mathbb{R},\ \forall x \geq k\ |g(x)| \leq c \cdot |f(x)|.$$

Les problèmes se classent en fonction du nombre d'étapes qu'une machine de Turing effectue avant de se retrouver sur un état final correspondant à une de leurs solutions.

Problème linéaire, polynomial, exponentiel. Soit n la taille de l'entrée d'un problème algorithmique.

- Ce problème est dit *linéaire* si la complexité du meilleur algorithme connu le résolvant est $O(n)$.
- Ce problème est dit *polynomial* s'il existe un polynôme P tel la complexité du meilleur algorithme connu le résolvant soit $O(P(n))$.
- Ce problème est dit *exponentiel* si la complexité du meilleur algorithme connu le résolvant est $O(2^n)$.

Nous étudions ici deux types de problèmes : les problèmes de décision et les problèmes d'optimisation.

Définition 2.2 (Problème de décision)**.** *Un problème de décision D est un ensemble d'instances $\mathcal{I}$ et une question. $\mathcal{I}$ est partitionné en un ensemble $\mathcal{I}^+$ d'instances positives et un ensemble $\mathcal{I}^-$ d'instances négatives. Résoudre D consiste à déterminer, pour un $I \in \mathcal{I}$ donné, si $I \in \mathcal{I}^+$ (ou si $I \in \mathcal{I}^-$).*

Définition 2.3 (Problème d'optimisation)**.** Un *problème d'optimisation* $\mathcal{O}$ est un quadruplet $(\mathcal{I}, \mathsf{Sol}, m, \mathsf{but})$.

- $\mathcal{I}$ est l'ensemble des *instances* de $\mathcal{O}$
- Pour $I \in \mathcal{I}$, $\mathsf{Sol}(I)$ est l'ensemble des *solutions réalisables* de I.
- Soit $I \in \mathcal{I}$ et $x \in \mathsf{Sol}(I)$, $m(I, x)$ est la *valeur* de la solution x de l'instance I.
- Le but prend la valeur min ou max et indique si l'on souhaite *minimiser* ou *maximiser* la valeur des solutions.

Remarque 1. Tout problème d'optimisation définit un problème de décision sous-jacent. Les instances de ce problème de décision $\mathcal{O}_D$ associé à $\mathcal{O}$ sont de la forme $(I, k) \in (\mathcal{I} \times \mathbb{Z})$. La question est de savoir s'il existe $x \in \mathsf{Sol}(I)$ tel que $m(x, I) \leq k$ dans le cas où $\mathsf{but} = \min$ (et $(x, I) \geq k$ dans le cas où $\mathsf{but} = \max$). Nous parlerons du problème de décision *associé* au problème d'optimisation.

Un problème de décision D est dans la classe P s'il peut être résolu en temps polynomial par une machine de Turing déterministe.

Un problème de décision D est dans la classe NP si, étant donné une instance I et une valeur x, une machine de Turing déterministe peut vérifier en temps polynomial si x est oui ou non solution de I. La classe de complexité P est incluse dans NP. Dans le reste de ce mémoire, nous supposons que l'inclusion réciproque est fausse.

Pour déterminer la difficulté d'un problème, il est courant de la comparer à celle d'un autre problème à l'aide d'une *réduction*. La réduction est une transformation d'un problème P dont la complexité est connue en un problème Q dont la complexité est inconnue. Si cette transformation satisfait certaines conditions, elle permet de montrer que résoudre Q est au moins aussi difficile que résoudre P.

Définition 2.4 (Réduction d'un problème D_1 vers D_2)**.** *Soit D_1 et D_2 deux problèmes de décision. Soit I_1 une instance de D_1. Une réduction est un couple d'algorithmes (f, g) tel que :*

- *f est un algorithme polynomial transformant I_1 en une instance $I_2 = f(I_1)$ de D_2.*
- *g est un algorithme polynomial transformant une solution s_2 de D_2 pour I_2 en une solution $s_1 = g(s_2)$ de D_1 pour I_1.*

On dit que D_1 se réduit à *D_2 et on note $D_1 \leq D_2$.*

Cette définition est synthétisée dans la figure 2.1.

Problèmes	D_1 (NP-difficile)		D_2
Instances	$I_1 \in I_{D_1}$	$\overset{f}{\longmapsto}$	$f(I_1) \in I_{D_2}$
Solutions	$s_1 = g(I_1, s_2) \in \mathsf{Sol}_{D_1}(I_1)$	$\overset{g}{\longmapsto}$	$s_2 \in \mathsf{Sol}_{D_2}(f(I_1))$

Figure 2.1 – Schéma d'une réduction d'un problème D_1 vers un problème D_2.

NP-difficile, NP-complet. Un problème de décision D est *NP-difficile* si tous les problèmes de la classe NP s'y réduisent par la réduction définie précédemment. Un problème NP-difficile est donc au moins aussi difficile que n'importe quel problème de la classe NP, mais il n'appartient pas forcément à cette classe. Un problème de décision D est *NP-complet*, s'il est NP-difficile et dans NP.

Ainsi, pour montrer qu'un problème est NP-complet, il suffit de vérifier deux points :

— qu'il soit effectivement dans NP, c'est-à-dire que l'on puisse statuer sur la validité d'une solution donnée en temps polynomial ;

— et qu'il soit NP-difficile, c'est-à-dire que l'on puisse montrer une réduction d'un problème connu pour être NP-difficile vers notre problème.

Enfin, nous donnons un résultat tiré de [CS90] sur les liens entre complexité des problèmes de décision et complexité des problèmes d'optimisation.

Théorème 2.5. *Soit O un problème d'optimisation. Si le problème de décision O_D associé est NP-complet, alors il n'existe pas d'algorithme polynomial permettant de résoudre O, sous l'hypothèse que $P \neq NP$.*

2.2 Théorie des graphes

À première vue, la gestion d'emploi du temps, l'allocation de fréquence radio à des antennes, l'analyse de réseaux sociaux, la recherche des chemins les plus courts dans un plan de métro ou encore la minimisation du nombres de réamorçages en chiffrement homomorphe (voir chapitre 5) n'ont pas grand chose à voir entre eux. Et pourtant, ils sont reliés par la structure qui permet de les modéliser : les graphes. La modélisation consiste à isoler dans un problème les informations qui seront utiles à sa résolution. En les organisant au sein d'une structure connue, ici les graphes, il est plus aisé de ramener la résolution du problème à celle d'un problème déjà étudié, ou, si le problème est nouveau, de se concentrer sur sa *substantifique moelle* afin de le résoudre.

Définition 2.6 (Graphe)**.** *Un* graphe *G est un couple (V, E), avec V l'ensemble des* sommets *de G et $E \subseteq V \times V$ l'ensemble des* arêtes *de G.*

Ainsi, un graphe peut être vu comme la représentation graphique d'une relation $\mathcal{R}$ existant entre les sommets. Deux sommets u et v sont reliés par une arête si $(u, v) \in \mathcal{R}$. Pour le cas de la gestion d'emploi du temps, les sommets seront par exemple les heures auxquels des cours ont lieu (8h-10h, 9h30-11h30, 14h-15h) et les arêtes les reliant représenteront les conflits entre deux horaires (comme le besoin de deux salles de cours différentes). De même, le problème de l'allocation de fréquence radio s'étudiera en prenant pour sommets les antennes et comme relation "l'antenne a interfère avec l'antenne b". Pour modéliser un réseau social, les sommets seront les individus et les arêtes la relation "sont amis sur Facebook".

Pour $u, v \in V^2$, le couple (u, v) note l'arête reliant u et v. On appelle *voisinage* de u l'ensemble $\{v|(u, v) \in E\}$. Le voisinage est important pour définir la notion de *degré* d'un sommet.

Définition 2.7 (Degré)**.** *Soit $G = (V, E)$ un graphe et soit $u \in V$ un sommet. On appelle* degré *de u le cardinal de son voisinage, i. e., $|\{v|(u, v) \in E\}|$. Le degré du sommet u est noté $d(u)$.*

On note $\delta(G)$ le degré minimal *du graphe G, i. e., $\delta(G) = \min\{d(u)|u \in V\}$.*

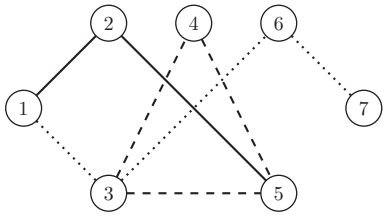


Figure 2.2 – Un graphe.

Si l'on revient à l'exemple d'un itinéraire en métro, la notion qui vient assez naturellement après la définition d'une arête est la notion de chemin.

Définition 2.8 (Chemin, cycle)**.** *Soit $G = (V, E)$ un graphe. Un chemin de u_0 à u_n, noté* $(u_0 - u_n)$-chemin*, est une suite $((u_0, u_1), \ldots, (u_{n-1}, u_n))$ telle que $\forall i,\ (u_i, u_{i+1}) \in E$. Le nombre d'arêtes d'un chemin définit sa* longueur *et on appelle* ordre *son nombre de sommets. Un* cycle *est un chemin tel que $u_0 = u_n$.*

Exemple 1 : La figure 2.2 représente un graphe. Le degré du sommet 3 est 4. En pointillé, nous avons un chemin du sommet 1 au sommet 7 et en tirets, un cycle.

Si un graphe simple permet de déjà répondre à plusieurs catégories de problèmes (l'allocation de fréquence radios ou le plus court chemin en métro par exemple), il peut être intéressant pour d'autres cas d'ajouter de l'information à cette structure de base. Dans notre cas, une information qui sera primordiale est l'orientation des arêtes. En effet, les relations étudiées entre les sommets ne sont pas systématiquement transitives : le couple (u, v) peut satisfaire la relation sans que ce ne soit le cas du couple (v, u).

Définition 2.9 (Graphe orienté)**.** *Un graphe $G = (V, E)$ est dit* orienté *si $\exists (u, v) \in V^2,\ u \neq v,\ (u, v) \in E$ et $(v, u) \notin E$.*

Dans le cas d'un graphe orienté, on ne parle plus d'arête, mais d'*arc*. Le sommet u est l'*origine* de l'arc (u, v) et le sommet v est appelé sa *destination*. Les définitions précédentes s'adaptent facilement à ce cas asymétrique

Définition 2.10 (Degré entrant/sortant, chemin orienté, cycle orienté)**.** Soit $G = (V, E)$ un graphe orienté et soit $u \in V$ un sommet.

- Le *degré entrant* de u est le nombre d'arcs qui ont u pour destination,
- Le *degré sortant* de u est le nombre d'arcs qui ont u pour origine.
- Un *chemin orienté* de u_0 à u_n est une suite d'arcs $((u_0, u_1), \ldots, (u_{n-1}, u_n))$ telle que $\forall i,\ (u_i, u_{i+1}) \in E$.
- Un *cycle orienté* est un chemin orienté dont les extrémités sont confondues, une suite d'arcs de la forme $((u_0, u_1), \ldots, (u_{n-1}, u_0))\ :\ \forall i,\ (u_i, u_{i+1}) \in E$.

Deux types de sommets sont distingués dans les graphes orientés : les *sources* et les *puits*.

Définition 2.11 (Source et puits)**.** *Soit $G = (V, E)$ un graphe orienté et $u \in V$ un sommet. Le sommet u une* source *(respectivement, un* puits*) si son degré entrant (respectivement, sortant) est nul.*

Exemple 2 : Un exemple de graphe orienté acyclique est donné figure 2.3. On remarque que si les graphes non-orientés représentés sur les figure 2.2 et 2.3 sont identiques, le cycle présent sur la figure 2.2 n'est plus présent sur la figure 2.3 du fait de l'orientation des arêtes. De même, il n'y a plus de chemin du sommet 1 au sommet 7. Enfin, les sommets 1 et 7 sont des sources et les sommets 5 et 6 sont des puits.

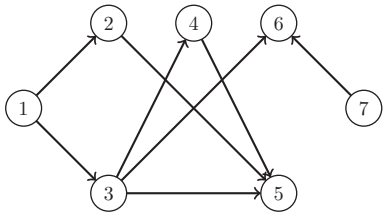


Figure 2.3 – Un graphe orienté acyclique.

Pour le problème de minimisation des réamorçages présenté chapitre 5, nous utilisons des graphes dont les sommets représentent des opérations. La relation modélisée par les arcs est "la sortie de l'opération a est une entrée de l'opération b". Ces graphes que nous étudions ont la particularité de ne jamais présenter de cycles : c'est une des conditions requises pour faire des calculs sur les données chiffrées (voir chapitre 4).

Définition 2.12 (Graphe orienté acyclique (DAG))**.** *Soit $G = (V, A)$ un graphe orienté. G est* acyclique *si G ne contient pas de cycle orienté. L'abréviation DAG vient de l'anglais* directed acyclic graph.

Comme nous l'avons évoqué plus haut, un des avantages de se ramener à un modèle bien établi est que notre problème peut avoir été déjà étudié pour ce modèle. Dans la suite de ce manuscrit, nous utilisons principalement les deux problèmes suivants : le problème du (s, t)-séparateur et la couverture d'un graphe par sommets.

Problème du (s, t)-séparateur

Un (s, t)-séparateur est un ensemble de sommets du graphe tel que tous les chemins de s à t possèdent un sommet dans cet ensemble. Plus formellement, nous avons la définition suivante.

Définition 2.13 ((s, t)-séparateur)**.** *Soit $D = (V, A)$ un graphe orienté acyclique. Soit s une source de D et t un puits de D. Soit $W \subseteq V \setminus \{s, t\}$ un sous-ensemble de sommets de D tel que tout chemin allant de s à t ait un sommet dans par W. L'ensemble de sommets W est appelé* (s, t)-séparateur.

En particulier, après suppression des sommets de W dans D, il n'existe plus de chemins entre s et t. On parle de *coupe* de graphe. Le problème du (s, t)-séparateur, ou de coupe minimale, est le problème d'optimisation consistant à trouver un (s, t)-séparateur de cardinalité minimale. Ce problème est lié à un autre problème célèbre de la théorie des graphes qui est celui du flot maximal. Ainsi, le calcul d'un (s, t)-séparateur minimal peut se faire en utilisant un algorithme de flots [Ber85]. Précisément, la complexité du calcul d'un (s, t)-séparateur minimal est $O(|V| \cdot |A| \cdot \log(|V|^2/|A|))$ [GT86].

Exemple 3 : Notons s le sommet 1 du graphe et t le sommet 5 du graphe représenté 2.3 Ils sont respectivement une source et un puits de ce graphe. La figure 2.4 donne un exemple de (s, t)-séparateur.

Couverture par sommets d'un graphe

Dans le chapitre 5, nous utilisons une variante du problème de couverture par sommets d'un graphe orienté acyclique. De manière informelle, ce problème consiste à trouver un sous-ensemble S de sommets tel que tout arc ait soit son origine, soit sa destination dans S. Un tel ensemble S est une *couverture* du graphe, et si un arc a son origine ou sa destination dans S, on dit qu'il est *couvert* par S. La définition formelle de ce problème en problème d'optimisation est la suivante.

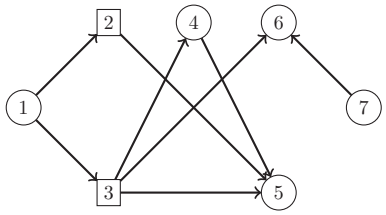


Figure 2.4 – Les sommets encadrés forment un (1,5)-séparateur du graphe.

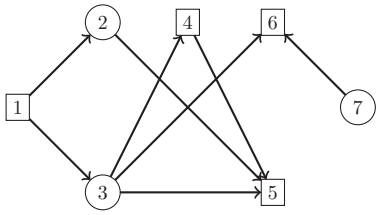


Figure 2.5 – Dans les rectangles, une couverture par sommets du graphe.

Définition 2.14 (Problème de couverture par sommets d'un graphe orienté acyclique)**.** *Le problème de couverture par sommets d'un graphe orienté acyclique est défini par le quadruplet* $(\mathcal{I}, \mathsf{Sol}, m, \mathsf{but})$ *tel que :*

- $\mathcal{I}$ *est l'ensemble des graphes orientés acycliques ;*
- *soit* $G = (V, E) \in \mathcal{I}$*, une solution* $S \in \mathsf{Sol}(G)$ *est un sous-ensemble de* V *tel que pour tout arc* $(u, v) \in A$*,* u *ou* v *est dans* S *;*
- *la valeur* $m(S, G)$ *d'une solution* S *est son cardinal* $|S|$ *;*
- $\mathsf{but} = \min$*.*

Le problème de décision associé est NP-complet [Kar72].

Exemple 4 : Un exemple de couverture par sommets dans un graphe orienté acyclique est présenté figure 2.5.

2.3 Langages rationnels

Un compilateur a pour objectif de transformer une série d'instructions écrites dans un langage de programmation donné en langage machine. Pour cela, il doit être capable d'identifier les séquences de caractères représentant une variable, une constante, un mot-clé correspondant à une instruction... Il doit être également capable de vérifier que le code écrit par l'utilisateur respecte bien la syntaxe établie afin d'effectuer sa transformation vers le langage machine sans heurts. C'est à ce genre de préoccupations que répond la théorie des langages, en fournissant des modèles afin de décrire et d'analyser des séquences de caractères. Nous nous intéressons à un cas particulier de langage, le cas des langages rationnels, et plus particulièrement aux différentes manières qui existent pour décrire ce type de langage.

2.3.1 Généralités

Nous commençons par définir un langage formel. Comme un langage naturel, les briques de base d'un tel objet sont les alphabets et les mots.

Définition 2.15 (Alphabet, mots)**.** *Un* alphabet Σ *est un ensemble fini non vide de symboles. Un* mot *est une suite finie (éventuellement vide) d'éléments de* Σ*.*

On note Σ^* l'ensemble de tous les mots formés à partir de l'alphabet Σ. Le mot vide, qui n'est composé d'aucun caractère, est noté ε.

Définition 2.16 (Langage)**.** *Un* langage *est un sous-ensemble de* Σ^**.*

Nous disons qu'un mot est *reconnu* par un langage s'il appartient à celui-ci. Les opérations ensemblistes usuelles peuvent s'appliquer aux langages. Nous notons donc :

$$\begin{aligned} L_1 \cup L_2 &= \{u \in \Sigma^* | u \in L_1 \text{ ou } u \in L_2\}, \\ L_1 \cap L_2 &= \{u \in \Sigma^* | u \in L_1 \text{ et } u \in L_2\}, \\ \overline{L} &= \{u \in \Sigma^* | u \notin L\}. \end{aligned}$$

Une des opérations spécifiques aux langages est la concaténation, ou produit, de langages. Soit x et y deux mots de Σ, le mot $u = xy$ est défini comme étant la concaténation des mots x et y, c'est-à-dire, comme le mot x suivi du mot y. Cette opération est étendue aux langages de la manière suivante :

$$L_1 L_2 = \{u \in \Sigma^* | \exists (x, y) \in L_1 \times L_2,\ u = xy\}.$$

Bien que cette opération n'ait rien à voir avec le produit ensembliste, on note $u^0 = \varepsilon$ et u^n le mot u concaténé n fois avec lui même. De même, L^0 désigne le langage réduit au mot vide $\{\varepsilon\}$ et L^n désigne le langage L concaténé n fois avec lui-même.

La deuxième opération spécifique aux mots et aux langages qui nous intéresse est la *fermeture de Kleene*, ou l'étoile de Kleene. Elle se définit comme suit sur les mots

$$u^\star = \bigcup_{i \geq 0} u^i,$$

et peut être étendue aux langages

$$L^\star = \bigcup_{i \geq 0} L^i.$$

Ces opérations permettent de définir une classe particulière de langages, utilisée au chapitre 7.

Définition 2.17 (Langage rationnel)**.** *Soit* Σ *un alphabet. Un langage rationnel sur* Σ *est défini récursivement de la manière suivante, avec un nombre fini d'appels à la dernière étape.*

- $\{\varepsilon\}$ *et* $\varnothing$ *sont des langages rationnels.*
- *Pour tout caractère* a *de* Σ*,* $\{a\}$ *est un langage rationnel.*
- *Si* L_1 *et* L_2 *sont des langages rationnels, alors* $L_1 \cup L_2$*,* $L_1 L_2$ *et* $L_1^\star$ *sont des langages rationnels.*

Dans le reste de ce mémoire, nous nous limitons aux langages rationnels. Le problème que l'on se pose est le suivant : étant donné un langage, comment décrire de manière compacte les mots reconnus par ce langage ? Pour cela, nous utilisons les deux outils que nous décrivons dans les sections suivantes : les expressions rationnelles et les automates.

2.3.2 Expressions rationnelles

Les expressions rationnelles décrivent les langages rationnels de manière plus compacte que la définition ensembliste du langage.

Définition 2.18 (Expression rationnelle)**.** *Soit* Σ *un alphabet. Une* expression rationnelle *est définie de la manière suivante.*

- ε *et* $\varnothing$ *sont des expressions rationnelles.*
- $\forall a \in \Sigma$*,* a *est une expression rationnelle.*
- *Si* e_1 *et* e_2 *sont des expressions rationnelles, alors* $(e_1|e_2)$*,* $(e_1 e_2)$ *et* $e_1^\star$ *sont des expressions rationnelles.*

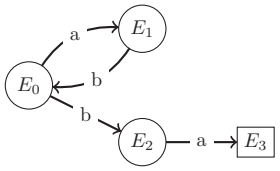


Figure 2.6 – L'automate correspondant à l'expression rationnelle $(ab)^*ba$.

L'opérateur | note, pour les expressions, le connecteur logique "ou".

Exemple 5 : Considérons l'alphabet Σ des lettres latines minuscules. Un langage rationnel simple sur cet alphabet est $L = \{\texttt{cryptologie}, \texttt{cryptographie}\}$. Sa description ensembliste est :

$$(\{\texttt{c}\}\{\texttt{r}\}\{\texttt{y}\}\{\texttt{p}\}\{\texttt{t}\}\{\texttt{o}\})\,(\{\texttt{l}\}\{\texttt{o}\}\{\texttt{g}\}) \cup (\{\texttt{g}\}\{\texttt{r}\}\{\texttt{a}\}\{\texttt{p}\}\{\texttt{h}\}))\,(\{\texttt{i}\}\{\texttt{e}\})\,.$$

L'expression rationnelle permettant de le décrire est la suivante : $\texttt{crypto}\,(\texttt{log}|\texttt{graph})\,\texttt{ie}$.

Les expressions rationnelles permettent d'analyser la syntaxe d'un langage, mais sont peu pratiques pour des applications calculatoires. Pour cela, nous avons recours aux automates.

2.3.3 Automates

Les automates finis sont des machines permettant de reconnaître des langages rationnels.

Définition 2.19 (Automate fini non-déterministe (NFA))**.** *Un automate fini déterministe est la donnée d'un quintuplet (Σ, Q, T, q_0, F) où :*

- *Σ est l'alphabet considéré ;*
- *Q est l'ensemble des états pris par l'automate ;*
- *$T : \Sigma \times Q \to Q$ est la fonction de transition qui à chaque couple (caractère, état) associe l'état suivant pris par l'automate ;*
- *q_0 est l'état initial de l'automate ;*
- *$F \subset Q$ est l'ensemble des états finaux, ou acceptés, de l'automate.*

Le théorème de Kleene donne l'équivalence entre un langage rationnel et un automate.

Théorème 2.20 (Kleene [Kle56])**.** *Un langage sur un alphabet Σ est rationnel si et seulement s'il est reconnu par un automate fini.*

Exemple 6 : Un automate fini est représenté graphiquement sur la figure 2.6. Chaque nœud représente un état de l'automate, et les flèches entre les nœuds matérialisent la fonction de transition entre les états. Un état initial est indiqué par l'indice zéro, et un état final par un carré.

L'expression rationnelle correspondant à l'automate de la figure 2.6 est $(ab)^*ba$.

Chapitre 3

Outils cryptographiques

Sommaire

Nous présentons dans ce chapitre les définitions des notions cryptographiques nécessaires à la compréhension des prochaines parties. Nous commençons par des rappels mathématiques, puis nous introduisons la méthodologie de la sécurité prouvée ainsi que les principales primitives utilisées dans ce manuscrit. Enfin, nous rappelons le fonctionnement des systèmes de preuves à divulgation nulle de connaissance.

3.1 Rappels mathématiques

3.1.1 Groupe

Définition 3.1 (Groupe)**.** Un *groupe* $(\mathbb{G}, \cdot)$ est un ensemble $\mathbb{G}$ muni d'une loi de composition interne $\cdot : \mathbb{G} \times \mathbb{G} \to \mathbb{G}$ satisfaisant les propriétés suivantes.

— *Associativité :* $\forall (x, y, z) \in \mathbb{G}^3$, $(x \cdot y) \cdot z = x \cdot (y \cdot z)$.

— *Élément neutre :* $\exists\ e \in \mathbb{G}$ tel que $\forall x \in \mathbb{G}$, $x \cdot e = e \cdot x = x$.

— *Existence d'un symétrique :* $\forall\ x \in \mathbb{G}$, il existe un élément $y \in \mathbb{G}$ tel que $x \cdot y = y \cdot x = e$.

Dans le cas de groupes non-spécifiés, nous utilisons des notations de lois de groupes multiplicatives. Par conséquent, le symétrique d'un élément $x \in \mathbb{G}$ est appelé *inverse* et noté x^{-1}, et l'élément neutre de $\mathbb{G}$ est noté $1_{\mathbb{G}}$ (et simplement 1 s'il n'y a pas d'ambiguïté sur le groupe). Nous utilisons exclusivement des groupes *commutatifs* et *finis*, définis comme suit.

Définition 3.2. Soit $(\mathbb{G}, \cdot)$ un groupe.

— *Commutativité :* $\mathbb{G}$ est dit *commutatif* ou *abélien* si $\forall\ (x, y) \in \mathbb{G}^2$, $x \cdot y = y \cdot x$.

— *Ordre d'un groupe :* le cardinal de $\mathbb{G}$, noté $|\mathbb{G}|$, est appelé *ordre* de $\mathbb{G}$. $\mathbb{G}$ est dit *fini* si $|\mathbb{G}|$ est fini.

Les groupes les plus utilisés en cryptographie ont la particularité d'être d'ordre premier, donc *cycliques*, et tous leurs éléments du groupe différents de l'élément neutre sont des *générateurs* du groupe.

Définition 3.3. Soit $(\mathbb{G}, \cdot)$ un groupe.
— *Sous-groupe engendré par un élément :* soit $x \in \mathbb{G}$, l'ensemble $\{x^n, n \in \mathbb{N}\}$, que l'on notera $\langle x \rangle$, est également un groupe que l'on appelle *sous-groupe engendré* par x.
— *Groupe cyclique :* un groupe $\mathbb{G}$ est *cyclique* s'il existe $x \in \mathbb{G}$ tel que $\langle x \rangle = \mathbb{G}$. Un tel élément sera appelé *générateur* de $\mathbb{G}$.

Nous définissons dans l'exemple suivant les groupes $(\mathbb{Z}_p, +)$ et $(\mathbb{Z}_p^\times, \cdot)$, qui sont fréquemment utilisés en cryptographie.

Exemple 1 : Soit $p \geq 2$ un entier. L'ensemble $\{0, \ldots, p-1\}$ muni de l'addition modulo p est un groupe abélien d'ordre p. L'associativité et la commutativité se déduisent des propriétés de l'addition d'entiers. L'élément neutre est 0. Comme $a+(p-a) = 0 \mod p$, l'inverse d'un élément a est $(p-a)$.

Le groupe $\mathbb{Z}_p^\times$ est défini sur le même ensemble, mais avec une loi de groupe induite par la multiplication modulo p. Il s'agit donc de trouver le sous-ensemble de $\{0, \ldots, p-1\}$ admettant un inverse multiplicatif modulo p. 0 est éliminé d'emblée. Le théorème de Bezout permet de caractériser de tels éléments : ce sont ceux qui sont premiers avec p. Dans le cas où p est premier, $\mathbb{Z}_p^\times$ est donc l'ensemble $\{1, \ldots, p-1\}$.

3.1.2 Couplages et groupes bilinéaires

En 1985, Neal Koblitz et Victor Miller ont introduit de manière indépendante l'usage des courbes elliptiques en cryptographie. En effet, les mathématiciens cherchaient des groupes sur lesquels on pouvait faire des calculs de manière efficace, mais pour lesquels certains problèmes calculatoires, et notamment, le problème du logarithme discret (voir section 3.2.1), restaient difficiles.

Les couplages ont, eux, été introduits en géométrie algébrique par André Weil en 1948. Ils se sont d'abord révélés un outil puissant en cryptanalyse : l'attaque MOV [MVO91] résolvant de manière efficace le problème du logarithme discret pour certaines courbes elliptiques, notamment les courbes supersingulières, alors privilégiées pour leur calcul simple de cardinalité. Depuis 2000, les couplages ont aussi été utilisés de manière constructive, permettant de construire des briques cryptographie jusqu'alors inaccessibles [Jou00, BF01].

Les *groupes bilinéaires* sont l'environnement dans lequel on peut définir et utiliser les *couplages*. Ils ont les propriétés suivantes.

Définition 3.4 (Groupes Bilinéaires)**.** *Les* groupes bilinéaires *sont un ensemble de trois groupes* $\mathbb{G}_1$*,* $\mathbb{G}_2$ *et* $\mathbb{G}_T$ *de même ordre* p *premier muni d'une application* $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$*, appelée* couplage*, qui a les propriétés suivantes.*
— *Bilinéarité :* $\forall\ g \in \mathbb{G}_1$*,* $\widetilde{g} \in \mathbb{G}_2$ *et* $a, b \in \mathbb{Z}_p$*,* $e(g^a, \widetilde{g}^b) = e(g, \widetilde{g})^{a \cdot b}$.
— *Non-dégénérescence :* $\forall\ g \in \mathbb{G}_1$ *et* $\widetilde{g} \in \mathbb{G}_2$*, si* $e(g, \widetilde{g}) = 1_{\mathbb{G}_T}$ *alors* $g = 1_{\mathbb{G}_1}$ *ou* $\widetilde{g} = 1_{\mathbb{G}_2}$.
— *Efficacité : l'application* e *est calculable efficacement.*

Dans la suite de ce manuscrit, la notation $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, e)$ désigne un environnement bilinéaire composé de trois groupes d'ordre p et d'une application $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$.

Galbraith, Paterson et Smart [GPS08] ont donné une typologie des couplages basée sur les relations entre les deux groupes $\mathbb{G}_1$ et $\mathbb{G}_2$.
— *Type 1 :* il existe 2 isomorphismes efficacement calculables $\phi_1 : \mathbb{G}_1 \to \mathbb{G}_2$ et $\phi_2 : \mathbb{G}_2 \to \mathbb{G}_1$.
— *Type 2 :* il existe un isomorphisme efficacement calculable $\phi_2 : \mathbb{G}_2 \to \mathbb{G}_1$ sans que cela ne soit le cas dans le sens inverse.

— *Type 3 :* il n'existe aucun isomorphisme efficacement calculable entre $\mathbb{G}_1$ et $\mathbb{G}_2$.

Les couplages de type 1 sont aussi appelés *couplages symétriques*, et l'on écrit fréquemment $\mathbb{G}_1 = \mathbb{G}_2$. Ceux de types 2 et 3 sont dits *asymétriques*. L'absence d'isomorphisme entre $\mathbb{G}_1$ et $\mathbb{G}_2$ permet l'émergence de nouvelles hypothèses de sécurité et donc de nouveaux schémas.

3.2 Sécurité prouvée

Le but premier de la cryptographie est de *protéger* une communication confidentielle. La résistance d'un schéma aux attaques extérieures est donc primordiale. Néanmoins, les premiers travaux pour définir cette notion de résistance de manière formelle ne sont finalement apparus qu'assez tard dans l'histoire de la cryptographie. En 1883, Auguste Kerckhoffs énonce ce qui apparaît aujourd'hui comme une évidence, mais qui va à l'encontre des pratiques de l'époque : entre autres, que la sécurité d'un système cryptographique doit uniquement reposer sur le secret de la clef (et non sur le secret d'un algorithme plus ou moins élaboré mis au point par les utilisateurs).

3.2.1 Preuve par réduction

Au chapitre précédent, nous nous attachions à déterminer la difficulté à résoudre certains problèmes. La question qui nous intéresse maintenant est de déterminer la difficulté à casser un schéma cryptographique. Ces deux questions sont assez proches et l'outil pour les résoudre est le même : la réduction. Dans les réductions de complexité, en effet, nous avons un problème connu pour être difficile à résoudre, et nous prouvons qu'un problème de complexité inconnue peut permettre de résoudre le premier problème : nous en déduisons que la difficulté de résolution du second problème est au moins équivalente à celle du premier problème.

Pour prouver la sécurité d'un schéma cryptographique, nous suivons un raisonnement semblable. Nous partons d'un problème connu pour être difficile, et prouvons qu'obtenir certaines informations sur les messages chiffrés avec le protocole évalué permet de résoudre notre problème de départ. Cette méthodologie, introduite par Goldwasser et Micali dans [GM84], est devenue incontournable dans la cryptographie moderne.

Quels problèmes peuvent être considérés comme suffisamment difficiles pour assurer la confidentialité des messages échangés ? Nous pourrions nous appuyer sur des problèmes NP-complets dont la difficulté est conjecturée avec des arguments solides (sous l'hypothèse que P $\neq$ NP). Mais ces problèmes sont peu adaptés à la cryptographie, car ils ne permettent pas de construire des fonctions à trappe, base de la cryptographie à clé publique. Nous devons donc nous appuyer sur des hypothèses *ad hoc*. Même si leur difficulté est moindre que celle des problèmes NP, la communauté cryptographique suppose que ces hypothèses sont suffisamment étudiées pour offrir des bases solides à la conception de schéma cryptographiques. De nouvelles hypothèses peuvent également être proposées pour obtenir des schémas plus fonctionnels ou plus efficaces, et dans ce cas, des gages de leur solidité devront être donnés, soit sous la forme d'une réduction à une hypothèse connue, soit sous la forme d'une preuve dans le modèle des groupes génériques [Sho97].

Un problème est considéré comme difficile si un algorithme ayant une complexité polynomiale le résout avec une probabilité négligeable à partir d'une certaine taille d'entrée λ. On appelle λ le paramètre de sécurité.

Définition 3.5 (Fonction négligeable)**.** *Une fonction $\varepsilon : \mathbb{N} \to \mathbb{R}$ est dite négligeable si et seulement si :*

$$\forall P \in \mathbb{R}[X],\ \exists k \in \mathbb{N},\ \forall n \geq k, \varepsilon(n) \leq \frac{1}{P(n)}.$$

Expérience $\mathbf{Exp}^{\mathsf{attaque}}_{\Pi,\mathcal{A}}(\lambda)$
E_1 ;
$\vdots$
E_n ;
Si attaque = succès, retourner 1 ;

Figure 3.1 – Expérience de sécurité générique.

Problèmes difficiles

De manière générale, nous introduisons les hypothèses de sécurité au fur et à mesure des schémas concernés. Nous présentons néanmoins ici le problème du logarithme discret qui est souvent la base des problèmes considérés difficiles dans les groupes d'ordre premier.

Logarithme discret. Soit $\mathbb{G}$ un groupe cyclique d'ordre p premier, et g un générateur de $\mathbb{G}$, nous considérons la fonction exponentielle en base $g : x \in \mathbb{Z}_p \mapsto g^x$. Cette fonction est une surjection de $\mathbb{Z}_p$ dans $\mathbb{G}$. Le *logarithme discret en base g* de $\mathbb{G}$ est consiste à trouver un antécédent d'un élément h de $\mathbb{G}$ pour cette foncion. Si la fonction exponentielle se calcule en temps polynomial, pour de larges catégories de groupes, il n'existe pas d'algorithme efficace pour calculer le logarithme discret d'un élément. Le problème du logarithme discret est alors défini comme suit : étant donné un groupe $\mathbb{G}$ cyclique d'ordre p premier, un générateur g et un élément h, trouver $a \in \mathbb{Z}_p$ tel que $h = g^a$.

Trouver des groupes $\mathbb{G}$ dans lesquels ce problème est suffisamment robuste pour fonder la sécurité de schémas cryptographiques est une question fondamentale pour la sécurité de nombreux schémas cryptographiques. Grossièrement, il existe deux types d'attaques contre le logarithme discret : les attaques génériques qui fonctionnent dans n'importe quel groupe et des attaques utilisant des spécificités du groupe considéré. Les attaques génériques, telles que Baby-step/giant-step ou l'attaque ρ de Pollard ont une complexité en temps en $O(\sqrt{p})$ où p est l'ordre du groupe attaqué. Les attaques spécifiques, comme les cribles de nombres, sont plus efficaces pour certaines catégories de groupes, notamment les corps finis à petite caractéristique, où l'algorithme de résolution devient quasiment polynomial [BGJT14].

3.2.2 Modèle de sécurité

Nous avons défini ci-dessus les conditions pour lesquelles casser la sécurité d'un schéma est considéré comme un problème difficile. Mais que signifie réellement casser la sécurité d'un schéma ? S'agit-il de retrouver une partie d'un message, un message, voire la clé ? De quelles ressources dispose l'adversaire ? C'est cette notion plutôt floue qu'il s'agit de clarifier avec un modèle de sécurité. Le modèle de sécurité détermine les éléments auxquels l'adversaire a accès et son objectif final, ainsi que l'environnement dans lequel a lieu l'attaque.

Expériences de sécurité

Les expériences de sécurité (ou jeux de sécurité) ont pour but de définir les éléments en possession de l'adversaire ainsi que son objectif final. Lorsqu'on parle de la sécurité d'un schéma, c'est toujours par rapport à une expérience particulière : c'est en effet celle-ci qui formalise la propriété de sécurité que l'adversaire ne doit pas être capable de casser. L'entité lançant le défi est nommée challengeur. Nous représentons une expérience de sécurité pour une attaque générique d'un adversaire $\mathcal{A}$ contre un schéma Π pour un paramètre de sécurité λ dans la figure 3.1.

La capacité de l'adversaire à réussir une expérience de sécurité est quantifiée par son *avantage*. C'est la différence des probabilités de réussite entre une attaque menée par un adversaire $\mathcal{A}$ réel

contre un schéma Π contre une expérience $\mathbf{Exp}^{\mathsf{attaque}}_{\Pi,\mathcal{A}}(\lambda)$ et une attaque menée aléatoirement contre cette même expérience. On le note $\mathsf{Adv}^{\mathsf{attaque}}_{\mathcal{A},\Pi}(\lambda)$. Le système est dit sûr pour la propriété décrite dans l'expérience si, pour tout adversaire s'exécutant en temps polynomial, cet avantage est négligeable.

Pour obtenir les meilleures garanties de sécurité, il est préférable que les attaques de l'adversaire ait lieu dans l'environnement réel ; où tous les éléments du protocole à évaluer se comportent comme dans leur définition. C'est ce qu'on appelle le *modèle standard*. Néanmoins, pour plusieurs raisons, notamment pour obtenir des schémas plus efficaces, il peut être nécessaire d'avoir recours à des idéalisations de l'environnement de l'adversaire pour faire la preuve de sécurité.

Modèle de l'oracle aléatoire

Ce modèle a été introduit par Fiat et Shamir [FS86] en 1986 et formalisé en 1993 par Bellare et Rogaway [BR93].

Dans ce modèle, on remplace certaines des fonctions de hachage (voir section 3.3.3) auxquelles l'adversaire a accès par un oracle qui renvoie à chaque requête différente des précédentes une valeur parfaitement aléatoire. Comme une vraie fonction de hachage, l'oracle aléatoire est déterministe : la même requête produira la même valeur.

Ce modèle est largement répandu, car il est à la source de nombreuses constructions cryptographiques efficaces. Des schémas ont néanmoins été produits [CGH98, CGH04] et prouvés sûr dans le modèle de l'oracle aléatoire alors que l'instanciation de cet oracle par une fonction de hachage les rendait non sûrs. Ainsi, théoriquement, les preuves données dans ce modèle sont moins robustes que des preuves données dans le modèle standard. Ces contre-exemples sont cependant à relativiser car assez artificiels : depuis l'introduction de ce modèle, aucun protocole prouvé sûr dans ce modèle n'a vu sa sécurité remise en cause.

Modèle des groupes génériques

Ce modèle a été introduit par Shoup [Sho97] et permet de donner des gages sur la robustesse d'hypothèse de sécurité non standards. Dans ce modèle, ce sont les opérations de groupes qui sont modélisées par un oracle. On considère que l'attaquant n'a pas accès à la structure particulière du groupe mais seulement à des définitions génériques des opérations sur les éléments du groupe. Il ne peut construire des nouveaux éléments qu'à partir des éléments qui lui sont fournis dans l'hypothèse et d'appels aux opérations de groupe. Prouver qu'une hypothèse est sûre dans le modèle des groupes génériques revient donc à montrer que l'adversaire ne peut pas construire des éléments qui lui permettrait de distinguer l'élément défi de l'hypothèse d'un élément aléatoire.

3.3 Primitives cryptographiques

Dans cette section, nous présentons les différentes briques cryptographiques qui permettent de construire des protocoles. Pour chacune de ces primitives, nous donnons également les propriétés de sécurité qui lui sont associées.

3.3.1 Chiffrement

Le chiffrement d'un message vise à assurer sa confidentialité lors de sa transmission d'un utilisateur A (Alice) vers un utilisateur B (Bob). Il existe deux grandes catégories de chiffrement.

— *Le chiffrement à clé secrète* est le premier type de chiffrement inventé. Alice et Bob se mettent préalablement d'accord sur l'utilisation d'un algorithme qui fonctionne à l'aide d'un secret commun : la clé secrète. La clé est utilisée à la fois pour chiffrer les données et pour les déchiffrer. Ce chiffrement est aussi appelé *chiffrement symétrique*.

— *Le chiffrement à clé publique* est indiscutablement la plus grande avancée de la cryptographie moderne. Il repose sur un constat simple : seul le récepteur du message, Bob, a besoin d'être en possession d'un secret. En effet, il est le seul à devoir être capable de déchiffrer les données. Le chiffrement, lui, peut s'effectuer de manière publique, à la manière d'un cadenas que tout le monde peut fermer, mais que seul le possesseur de la clé peut ouvrir.

Dans la suite de ce mémoire, nous nous concentrons surtout sur les schémas de chiffrement à clé publique dont nous donnons maintenant la syntaxe formelle. Un tel schéma se compose de cinq algorithmes définis de la manière suivante.

— L'algorithme PPGen prend en entrée un paramètre de sécurité λ et retourne les paramètres publics du schéma pp.
— L'algorithme KeyGen prend en entrée les paramètres publics pp et retourne une clé publique pk et une clé privée sk.
— L'algorithme $\mathsf{Encrypt}$ prend en entrée les paramètres publics pp, la clé publique pk et un message M et retourne un chiffré c.
— L'algorithme $\mathsf{Decrypt}$ prend en entrée les paramètres publics pp, la clé privée sk et un chiffré c et retourne un message M'.

Un schéma de chiffrement symétrique présente une syntaxe similaire, mais il ne fait pas de différence entre la clé utilisée pour chiffrer et celle utilisée pour déchiffrer : on parlera alors de *clé secrète*, et on la notera également sk. La première propriété attendue d'un schéma de chiffrement est la *correction* : un message chiffré avec une clé publique pk doit être déchiffré correctement par la clé privée sk associée à pk. En d'autres termes, il est requis que le cas où $\mathsf{Decrypt}(\mathsf{pp}, \mathsf{sk}, \mathsf{Encrypt}(\mathsf{pp}, \mathsf{pk}, M)) \neq M$ n'arrive qu'avec une probabilité négligeable.

Modèle de sécurité

Pour un schéma de chiffrement, diverses propriétés de sécurité peuvent être formalisées selon le but que l'adversaire veut atteindre et ses capacités. L'attaque la plus forte conduit à un *cassage total* du schéma : c'est l'attaque où l'adversaire cherche à recouvrer la clé privée de l'utilisateur. L'adversaire peut alors déchiffrer n'importe quel message. Mais un schéma résistant à cette attaque n'est pas pour autant considéré comme sûr. Il est requis que l'adversaire ne puisse obtenir aucune information sur le texte en clair à partir d'un chiffré. Cette notion est équivalente à la propriété *d'indistinguabilité* notée IND : l'adversaire a une probabilité négligeable de distinguer deux chiffrés de messages différents. Il y a plusieurs variantes de cette attaque selon les ressources de l'adversaire.

— L'attaque à clairs choisis. Dans cette attaque, notée $\mathsf{IND\text{-}CPA}$ pour *chosen plaintexts attack*, l'adversaire peut choisir les messages clairs dont il aura à distinguer les chiffrés. C'est l'attaque la plus fréquemment considérée. Cette attaque est décrite à la figure 3.2.
— L'attaque à chiffré choisis. Dans cette attaque, notée $\mathsf{IND\text{-}CCA}$ pour *chosen ciphertexts attack*, en plus de choisir les messages clairs, l'adversaire a accès à un oracle de déchiffrement lui permettant de déchiffrer un certain nombre de messages avant la requête du challengeur.
— L'attaque à chiffrés choisis adaptative. Dans cette attaque, notée $\mathsf{IND\text{-}CCA2}$, l'adversaire a la possibilité d'envoyer autant de requêtes qu'il le souhaite à l'oracle de déchiffrement, en les adaptant éventuellement à la requête du challengeur.

Dans les deux derniers types d'attaques, nous supposons que l'oracle de déchiffrement rejette toutes les requêtes correspondant aux messages clairs choisis par l'attaquant.

Dans le cas du jeu $\mathsf{IND\text{-}CPA}$, un attaquant aléatoire a une probabilité de réussite $1/2$, l'avantage de $\mathcal{A}$ est donc calculé comme étant :

$$\mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathcal{A},\Pi}(\lambda) = |\mathtt{Pr}(\mathcal{A} : b = b') - 1/2|.$$

Exemple de schéma de chiffrement

Nous présentons enfin le schéma de chiffrement ElGamal [Gam84].

Expérience $\mathbf{Exp}^{\mathsf{IND\text{-}CPA}}_{\Pi,\mathcal{A}}(\lambda)$
$pp \leftarrow \mathsf{PPGen}(1^\lambda)$; $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda, \mathsf{pp})$;
$(M_0, M_1) \leftarrow \mathcal{A}(1^\lambda, \mathsf{pp}, \mathsf{pk})$;
$b \xleftarrow{\$} \{0, 1\}$;
$C_b \leftarrow \mathsf{Encrypt}(\mathsf{pp}, \mathsf{pk}, M_b)$;
$b' \leftarrow \mathcal{A}(C_b)$;
Si $b' \neq b$, retourner 0 ;
Retourner 1 ;

Figure 3.2 – Jeu de l'indistinguabilité à clairs choisis : $\mathbf{Exp}^{\mathsf{IND\text{-}CPA}}_{\Pi,\mathcal{A}}(\lambda)$

— PPGen. Soit λ le paramètre de sécurité. L'algorithme retourne $\mathsf{pp} = (\mathbb{G}, q, g)$ avec $\mathbb{G}$ un groupe d'ordre q premier et g un générateur de $\mathbb{G}$.
— KeyGen. L'algorithme de génération des clés prend en entrée $\mathsf{pp} = (\mathbb{G}, q, g)$, et choisit un x aléatoire dans $\mathbb{Z}_p$. Il retourne la clé privée $\mathsf{sk} = x$ et la clé publique $\mathsf{pk} = h = g^x$.
— Encrypt. L'algorithme de chiffrement prend en entrée pp, pk et un message M appartenant à $\mathbb{G}$. Il choisit un s aléatoire dans $\mathbb{Z}_q$ et retourne le chiffré c comme la paire $(c_1, c_2) = (g^s, h^s \cdot M)$.
— Decrypt. L'algorithme de déchiffrement prend en entrée pp, sk et un chiffré $c = (c_1, c_2)$. Il retourne $M = c_2/c_1^x$.

Il n'est pas difficile de prouver que ce schéma est correct. En effet,

$$\frac{c_2}{c_1^x} = \frac{h^s \cdot M}{(g^s)^x} = \frac{h^s \cdot M}{(g^s)^x} = \frac{(g^x)^s \cdot M}{(g^s)^x} = \frac{g^{xs} \cdot M}{g^{xs}} = M.$$

De plus le schéma de chiffrement ElGamal est IND-CPA sous l'hypothèse Diffie-Hellman décisionnelle, que nous rappelons ci-dessous.

Hypothèse 3.6 (Diffie-Hellman Décisionnel DDH)**.** *Soit $g, g^a, g^b, g^z \in G$, le problème consiste à décider si $z = ab$ ou si z est aléatoire.*

3.3.2 Proxy de rechiffrement

Le proxy de rechiffrement est une primitive permettant à un proxy (ou serveur) de rechiffrer un message à l'attention d'une partie A pour une partie B. Cette primitive est constituée des algorithmes suivants.

— $\mathsf{PPGen}(\lambda) \rightarrow \mathsf{pp}$. Cet algorithme renvoie les paramètres publics du schéma pp.
— $\mathsf{KeyGen}(\mathsf{pp}) \rightarrow (\mathsf{sk}, \mathsf{pk})$. L'algorithme de génération de clés, exécuté par chacune des parties, prend en entrée les paramètres publics du schéma et renvoie une paire de clé publique, clé privée $(\mathsf{sk}, \mathsf{pk})$.
— $\mathsf{ReKeyGen}(\mathsf{pp}, \mathsf{sk}_A, \mathsf{pk}_B) \rightarrow \mathsf{rk}_{A\rightarrow B}$. L'algorithme de génération de clé de rechiffrement de A vers B est exécuté par A. À partir des paramètres publics, de la clé secrète de A et de la clé secrète de B, l'algorithme retourne la clé de rechiffrement $\mathsf{rk}_{A\rightarrow B}$ qui permet de transformer un chiffré destiné à A en un chiffré destiné à B.
— $\mathsf{Enc}(\mathsf{pp}, M, \mathsf{pk}) \rightarrow c_1$. L'algorithme de chiffrement, exécutable par n'importe quelle partie, prend en entrée les paramètres publics pp la clé publique pk et un message M du destinataire du message et retourne un chiffré c.
— $\mathsf{ReEnc}(\mathsf{pp}, \mathsf{rk}_{A\rightarrow B}, c_1) \rightarrow c_2/\perp$. L'algorithme de rechiffrement, exécuté par le proxy, prend en entrée les paramètres publics pp, la clé de rechiffrement $\mathsf{rk}_{A\rightarrow B}$ de A vers B et un chiffré c_1 destiné à A. Il retourne un chiffré c_2 destiné à B qui ne peut être rechiffré ou un message d'invalidité $\perp$.

— $\mathsf{Dec}_1(\mathsf{pp}, \mathsf{sk}, c_1) \to M/\perp$. Ce premier algorithme de déchiffrement prend en entrée les paramètres publics pp, la clé secrète d'une partie sk et un chiffré c_1 issu de la procédure Enc et retourne un message M ou $\perp$.
— $\mathsf{Dec}_2(\mathsf{pp}, \mathsf{sk}, c_2) \to M/\perp$. Ce deuxième algorithme de déchiffrement prend en entrée les paramètres publics pp, la clé secrète d'une partie sk et un chiffré c_2 issu de la procédure ReEnc et retourne un message M ou $\perp$.

Nous donnons une instantiation de cette primitive basée sur le chiffrement totalement homomorphe au chapitre 7.

3.3.3 Fonctions de hachage

De manière générale, une fonction de hachage permet simplement de compresser des données de taille arbitraire en une chaîne de caractère de taille donnée. Dans cette section, nous décrivons les *fonctions de hachage cryptographiques*, c'est-à-dire des fonctions de hachage avec des propriétés de sécurité supplémentaires. Dans le reste de ce mémoire, toutes les fonctions de hachages considérées seront cryptographiques. Soit $h : X = \{0,1\}^* \to Y$ une fonction de hachage, les propriétés attendues d'une fonction de hachage cryptographique sont formalisées de la manière suivante.

Résistance à la préimage. Pour tout y dans Y, la probabilité de trouver x dans X tel que $h(x) = y$ est négligeable.

Résistance à la seconde préimage. Étant donné x dans X tel que $h(x) = y$, la probabilité de trouver $x' \neq x$ dans X tel que $h(x') = y$ est négligeable.

Résistance aux collisions. La probabilité de trouver x et x' dans X tels que $h(x) = h(x')$ est négligeable.

Nous utilisons au chapitre 6 (une variante du) Leftover Hash Lemma [ILL89] comme source d'entropie. Ce lemme permet, pour certaines familles de fonctions de hachage, d'estimer la distance entre la distribution de sortie d'une fonction de hachage et la distribution uniforme. Il s'applique aux familles k-universelles de fonctions de hachage.

Définition 3.7 (Famille k-universelle de fonctions de hachage)**.** *Soit $\mathcal{H}$ une famille de fonctions de hachage de X dans Y. $\mathcal{H}$ est dite* k-universelle *si pour tout h dans $\mathcal{H}$ et tout $x_1, \cdots, x_k$ deux à deux distincts dans X,*

$$\mathtt{Pr}\left[h(x_1) = \cdots = h(x_k)\right] = \frac{1}{|Y|^{k-1}}.$$

Exemple 2 : La famille des produits scalaires $\mathcal{H}_a = \langle a, \cdot \rangle : \left(\mathbb{Z}_p^\ell\right)^2 \to \mathbb{Z}_p$ avec a un vecteur aléatoire de $\mathbb{Z}_p^\ell$ est une famille de fonctions de hachage 2-universelle. Soit $X = (x_1 || \cdots || x_\ell)$ et $\tilde{X} = (\tilde{x_1} || \cdots || \tilde{x_\ell})$ dans $\mathbb{Z}_p^\ell$ tels que $X \neq \tilde{X}$ et $k(X) = k(\tilde{X})$. Sans perte de généralité, on peut supposer en particulier que $x_1 \neq \tilde{x_1}$. On en déduit successivement :

$$\begin{aligned}
\prod_{i=1}^{\ell} a_i^{x_i} &= \prod_{i=1}^{\ell} a_i^{y_i}, \\
\prod_{i=1}^{\ell} a_i^{(x_i - y_i)} &= 1, \\
a_1^{x_1 - y_1} &= \left(\prod_{i=2}^{\ell} a_i^{(x_i - y_i)}\right)^{-1}, \\
a_1 &= \left(\prod_{i=2}^{\ell} a_i^{(x_i - y_i)}\right)^{-\frac{1}{(x_1 - y_1)}}.
\end{aligned}$$

Maintenant, calculons la probabilité qu'une telle collision apparaisse. Pour chaque élément $a_2, \cdots, a_\ell$, il y a exactement $|\mathbb{Z}_p| = p$ choix, mais pour obtenir la collision, a_1 est lui entièrement déterminé par $(a_2, \cdots a_\ell)$. La probabilité de collision est donc

$$\frac{p^{\ell-1}}{p^\ell} = \frac{1}{p} = \frac{1}{|\mathbb{Z}_p|}.$$

Ce résultat nous sera utile au chapitre 6.

Lemme 3.8 (Leftover Hash Lemma [ILL89])**.** *Soit $\mathcal{H}$ une famille 2-universelle de fonctions de hachage de X vers Y. Si h est tirée uniformément dans $\mathcal{H}$ et x est tiré uniformément dans X, alors la distribution $(h, h(x))$ est au plus à distance $(1/2)\sqrt{|Y|/|X|}$ de la distribution uniforme sur $\mathcal{H} \times Y$.*

3.3.4 Mise en gage

Un schéma de *mise en gage* cryptographique permet de s'engager sur la valeur d'une donnée, même si celle ci n'est pas révélée immédiatement. Un cas d'usage trivial mais parlant est un jeu de pile ou face à distance. Alice choisit un bit (pile ou face) et ne le dévoile pas tant que Bob n'a pas effectué le tirage. Sans un protocole *ad hoc*, Alice pourrait facilement tricher sur le bit choisi. L'acteur s'engageant sur la valeur (dans notre exemple, Alice) est le prouveur $\mathcal{P}$, la personne recevant la mise en gage est le vérifieur $\mathcal{V}$ (Bob). Bien que la valeur engagée reste secrète, le prouveur n'a plus la possibilité de la changer : le vérifieur peut demander à ce qu'elle soit rendue publique à tout moment.

Un tel schéma Γ se compose de trois algorithmes définis de la manière suivante.

— L'algorithme d'initialisation Setup prend en entrée le paramètre de sécurité λ et retourne les paramètres publics pp.
— L'algorithme d'engagement Commit prend en entrée les paramètres publics pp et un message M et retourne une mise en gage C et une valeur r permettant de vérifier la validité de l'engagement.
— L'algorithme d'ouverture Open qui prend en entrée C, r et M et retourne 1 si C est une mise en gage valide du message M pour r et 0 sinon.

Sécurité. Le prouveur et le vérifieur ont des attentes apparemment contradictoires d'un schéma de mise en gage Γ. Le vérifieur souhaite qu'il soit *résistant aux collisions* (RC) pour le prouveur, tandis que ce dernier souhaite que sa valeur reste *confidentielle* (PRV) vis-à-vis du vérifieur. Plus précisément, les propriétés suivantes sont requises.

— Le schéma est résistant aux collisions (ou contraignant) si aucun adversaire probabiliste polynomial ne peut construire une collision sur une mise en gage, c'est-à-dire un triplet (C, r, r') ainsi que deux messages M et M' distincts tels que $\mathsf{Open}(C, r, M) = \mathsf{Open}(C, r', M') = 1$. L'expérience de résistance aux collisions est décrite figure 3.3
— Le schéma est confidentiel si aucun adversaire probabiliste polynomial $\mathcal{A}$ ne peut générer une paire de messages (M_0, M_1) telle que $\mathcal{A}$ puisse distinguer leurs mises en gage respectives. L'expérience de confidentialité est décrite figure 3.4

Ces propriétés peuvent être calculatoires ou inconditionnelles (ou parfaites). Si elles sont calculatoires, cela signifie qu'un adversaire ayant un avantage non négligeable contre une de ces propriétés a un avantage non négligeable contre une hypothèse de sécurité calculatoire. Un schéma de mise en gage ne peut être à la fois inconditionnellement contraignant et confidentiel [Dam98].

Expérience $\mathbf{Exp}^{\mathsf{RC}}_{\Gamma,\mathcal{A}}(\lambda)$
$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$;
$(C,(M,r),(M',r')) \leftarrow \mathcal{A}(1^\lambda,\mathsf{pp})$;
Si $M = M'$, retourner 0 ;
Retourner $\mathsf{Open}(C,r,M) == 1 \wedge \mathsf{Open}(C,r',M') == 1$;

Figure 3.3 – Jeu de la résistance aux collisions d'une mise en gage : $\mathbf{Exp}^{\mathsf{RC}}_{\Gamma,\mathcal{A}}(\lambda)$

Expérience $\mathbf{Exp}^{\mathsf{PRV}}_{\Gamma,\mathcal{A}}(\lambda)$
$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$;
$(M_0,M_1) \leftarrow \mathcal{A}(1^\lambda,\mathsf{pp})$;
$b \xleftarrow{\$} \{0,1\}$;
$(C_b,r_b) \leftarrow \mathsf{Commit}(\mathsf{pp},M_b)$;
$b' \leftarrow \mathcal{A}(C_b)$;
Si $b \neq b'$, retourner 0 ;
Retourner 1 ;

Figure 3.4 – Jeu de la confidentialité d'une mise en gage : $\mathbf{Exp}^{\mathsf{PRV}}_{\Gamma,\mathcal{A}}(\lambda)$

Mise en gage de Pedersen

Pedersen [Ped91] a proposé le schéma de mise en gage suivant, défini sur un groupe $\mathbb{G}$ d'ordre premier p.

- $\mathsf{Setup}(1^\lambda)$: soit λ le paramètre de sécurité, cet algorithme génère un premier p, un groupe $\mathbb{G}$ d'ordre p et deux de ses générateurs g et h.
- $\mathtt{Commit}(M)$: pour mettre en gage un message $M \in \mathbb{Z}_p$, $\mathcal{P}$ choisit un r aléatoire dans $\mathbb{Z}_p$ et retourne $(C,r) \leftarrow (g^r \cdot h^M, r)$.
- $\mathtt{Open}(C,M,r)$: le triplet (C,M,r) est valide si $C = g^r \cdot h^M$.

Ce protocole est parfaitement confidentiel (au sens de la théorie de l'information) et calculatoirement contraignant sous l'hypothèse du logarithme discret.

3.4 Preuves à divulgation nulle de connaissance

Lorsqu'il s'agit de convaincre un lecteur de la validité d'une relation en mathématiques, il suffit de fournir une preuve qui peut être ensuite vérifiée par des pairs. A priori, aucune autre propriété que la correction n'est attendue de cette preuve. Notamment, il semblerait complètement contre intuitif d'exiger que la preuve ne révèle aucune information sur les éléments qui la composent, en dehors du fait qu'ils vérifient effectivement la relation en question.

Dans les protocoles qui suivent, un prouveur $\mathcal{P}$ va interagir avec un vérifieur $\mathcal{V}$ pour prouver que des éléments secrets $x_1,\cdots,x_n$ vérifient une certaine relation $\mathcal{R}$. Les propriétés attendues d'une tel protocole sont les mêmes que celles d'une preuve mathématique : la *complétude* et la *validité*. Dans tout le manuscrit, nous considérons que le prouveur $\mathcal{P}$ a affaire à un vérifieur $\mathcal{V}$ honnête.

Complétude. $\mathcal{V}$ refuse une preuve valide (c'est-à-dire une preuve sur un ensemble de secrets qui vérifie la relation) avec une propriété négligeable.

Validité. $\mathcal{V}$ accepte une preuve émise par un prouveur ne connaissant aucun secret qui vérifie le prédicat avec une probabilité négligeable.

Dans ce manuscrit, nous nous concentrons sur les *preuves de connaissance*. Il ne s'agit pas seulement pour $\mathcal{P}$ de prouver que des éléments vérifiant une relation existent, mais qu'il en

connaît au moins un exemple : c'est ce qu'on appelle un *témoin* pour la relation $\mathcal{R}$. Nous notons une preuve de connaissance de la manière suivante : $\mathsf{PoK}\,(x : \mathcal{R}(x, y))$ où x est le secret sur lequel porte la preuve et y est public. Les preuves de connaissances sont caractérisées par la présence d'un *extracteur de connaissance*. La définition formelle d'un tel extracteur peut se trouver dans [BG92].

Définition 3.9. *Un système de preuve interactif $(\mathcal{P}, \mathcal{V})$ est une preuve de connaissance pour une relation $\mathcal{R}$ s'il existe un extracteur de connaissance K tel que pour tout x vérifiant $(x, y) \in \mathcal{R}$ et pour tout prouveur $\mathcal{P}^*$ capable de produire une preuve acceptée avec une probabilité p_x proche de 1 par $\mathcal{V}$:*

$$\mathcal{Pr}\left[K^{P^*}(x) \in \mathcal{R}\right] \geq poly(p^*).$$

En d'autres termes, la probabilité que l'extracteur K puisse, en accédant à P^*, produire un témoin x valide pour la relation $\mathcal{R}$ est au moins polynomiale en la capacité de P^* à convaincre $\mathcal{V}$ qu'il connaît x.

Les protocoles de preuve de connaissance à divulgation nulle de connaissance suivent classiquement un schéma en trois passes qui est décrit figure 3.5. Le triplet (t, c, s) est la *transcription* de la communication entre $\mathcal{P}$ et $\mathcal{V}$ sur une entrée x.

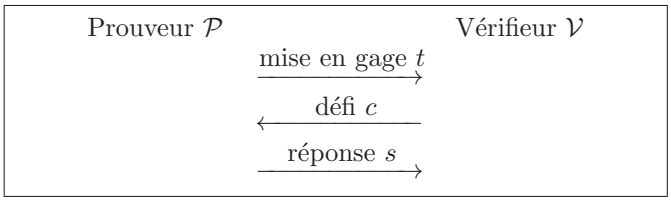


Figure 3.5 – Protocole de preuve de connaissance interactif

Essentiellement, la propriété de non-divulgation signifie que pour tout vérifieur $\mathcal{V}$, et pour tout secret x, un simulateur $\mathcal{S}$ fonctionnant en temps polynomial peut générer une transcription d'une preuve interactive qui aurait eu lieu entre un véritable $\mathcal{P}$ et $\mathcal{V}$. En d'autres termes, le vérifieur n'apprend rien de ce qu'il reçoit d'un véritable prouveur $\mathcal{P}$. En particulier, il ne peut rien calculer sur le secret x qu'il n'était pas déjà capable de calculer avant la preuve. Nous notons z la chaîne de caractère représentant l'information préalable de $\mathcal{V}$ sur le secret x. Enfin, nous notons $\mathsf{Vue}_{\mathcal{P},\mathcal{V}}(x, z)$ la transcription d'une preuve entre $\mathcal{P}$ et $\mathcal{V}$ portant sur le secret x, où $\mathcal{V}$ a l'information préalable z. Ces notations permettent de définir formellement la propriété de non-divulgation de la manière suivante.

Définition 3.10 (Non-divulgation)**.** *Une preuve $\mathsf{PoK}\,(\{x_i\} : \mathcal{R}(x_1, \cdots x_n))$ est à divulgation nulle de connaissance (pour un vérifieur honnête) si pour tout vérifieur (honnête) en temps polynomial $\mathcal{V}$ et prouveur $\mathcal{P}$, il existe un simulateur en temps polynomial $\mathcal{S}$ tels que les distributions suivantes soient distinguables avec une probabilité négligeable :*

- *l'ensemble des vues $\mathsf{Vue}_{\mathcal{P},\mathcal{V}}(x, z)$ pour $\mathcal{V}$ pour tout secret $x = \{\{x_i\}$ vérifiant la relation $\mathcal{R}$ et toute information préalable $z \in \{0, 1\}^*$;*
- *l'ensemble des sorties $\mathcal{S}(x, z)$ du simulateur $\mathcal{S}$ avec pour entrée le secret x et l'information préalable z.*

3.4.1 L'heuristique de Fiat-Shamir

Dans le reste de ce manuscrit, nous utilisons plutôt des protocoles non-interactifs. L'heuristique de Fiat-Shamir [FS86] est une méthode permettant de transformer un protocole à trois passes du type de celui décrit figure 3.5 en protocole non interactif. Pour assurer que le prouveur ne puisse pas tricher lors de la preuve de connaissance, il est nécessaire que le défi envoyé par le vérifieur ne puisse pas être prédit par le prouveur. Pour rendre le protocole non interactif, le

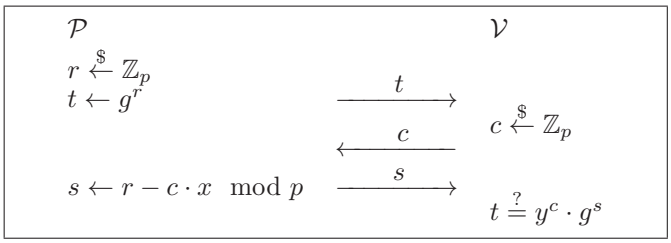


Figure 3.6 – Protocole de Schnorr

vérifieur est ainsi remplacé par une fonction pseudo-aléatoire afin de générer le défi. En pratique, on utilise des fonctions de hachage cryptographiques en tant que fonctions pseudo-aléatoires, ce qui implique que la sécurité de ces protocoles ne peut être prouvée que dans le modèle de l'oracle aléatoire [PS00].

Plus précisément, le défi est alors calculé comme suit : $c = h(\mathsf{pp}\|t)$ où pp sont les éléments publics impliqués dans la preuve et t la mise en gage calculée par le prouveur. Le prouveur calcule alors la réponse pour ce défi c. La preuve est le couple (c, s). Pour vérifier la validité de la preuve, le vérifieur teste si le défi est formé correctement $(c \stackrel{?}{=} h(\mathsf{pp}\|t))$ puis que la réponse s est correcte pour le défi c. Nous donnons un exemple de cette transformation dans la section suivante, puis nous donnerons les preuves de connaissance sous leur forme non-interactive comme c'est celle qui sera utilisée lors de la conception de schémas.

Nous noterons les preuves non-interactives ainsi : $\mathsf{NIZK}\left(\{x_i\} : \mathcal{R}(x_1, \cdots x_n)\right)$.

3.4.2 Exemples de preuves de connaissance

Preuve de connaissance d'un logarithme discret

Le but de ce protocole est de prouver la connaissance d'un logarithme discret dans un groupe cyclique d'ordre premier p par rapport à une base g commune au prouveur et au vérifieur, on le note donc ainsi $\mathsf{PoK}\left(x : y = g^x\right)$. Proposé par Schnorr en 1989 [Sch89], il est décrit figure 3.6.

En utilisant l'heuristique de Fiat-Shamir, nous pouvons transformer cette preuve interactive en une preuve non interactive de la manière suivante [Cam98].

$\mathcal{P}$ choisit aléatoirement r dans $\mathbb{Z}_p$ et calcule $t = g^r$. Il en dérive le défi c à l'aide d'une fonction de hachage H : $c = H(g\|y\|t)$. Dans la preuve de sécurité, H est modélisée par un oracle aléatoire. Il calcule enfin la réponse à ce défi $s = r - c \cdot x \mod p$. La preuve π est le couple (c, s).

Un vérifieur $\mathcal{V}$ peut s'assurer de la validité de la preuve en testant si $c \stackrel{?}{=} H(g\|y\|y^c g^s)$.

Preuve de connaissance d'un double logarithme discret

Nous décrivons dans cette section une preuve de connaissance non-interactive proposée par [CS97] pour prouver la connaissance du double logarithme discret de y relatif à deux bases g et h : $\mathsf{NIZK}\left(x : y = g^{h^x}\right)$. Soit $\mathbb{G}$ un groupe d'ordre p premier et λ le paramètre de sécurité. $\mathcal{P}$ choisit des r_i aléatoirement dans $\mathbb{Z}_p$ et calcule pour chaque i : $T_i = g^{h^{r_i}}$. Soit H une fonction de hachage, le défi est alors $c = H(g\|h\|y\|T_1\| \cdots \|T_\lambda)$. $\mathcal{P}$ calcule enfin sa réponse $s_i = r_i - c[i] \cdot x \mod p$, où $c[i]$ est est le i-ème bit de c pour tout $1 \leq i \leq \lambda$. La preuve π est le n-uplet :

$$\pi = (c, s_1, \cdots, s_\lambda)\,.$$

Pour statuer sur la validité de π, $\mathcal{V}$ vérifie si pour $1 \leq i \leq \lambda$,

$$\begin{cases} T_i \stackrel{?}{=} g^{h^{s_i}} & \text{si } c[i] = 0 \\ T_i \stackrel{?}{=} y^{h^{s_i}} & \text{si } c[i] = 1. \end{cases}$$

Deuxième partie

Calculs génériques sur les données chiffrées

Chapitre 4

Schémas et Challenges

Sommaire

Comme signalé en introduction, assurer la confidentialité des données ne suffit plus à rendre compte des usages actuels d'internet (réseaux sociaux, économie de partage, internet des objets...). Offrir aux utilisateurs la confidentialité de leurs données sans en compromettre un usage flexible et efficace est un défi majeur de la cryptographie moderne. Il y a actuellement plusieurs méthodes pour effectuer des calculs génériques sur les données chiffrées : le chiffrement totalement homomorphe, qui est l'objet principal de cette thèse, mais il y a aussi le chiffrement fonctionnel et le calcul multi-parties qui répondent à d'autres cas d'usages. Il faut aussi se poser la question de ce qu'on entend par "calculs génériques". En effet, nous verrons que l'algorithmique sur les données chiffrées et celle sur les données en clair sont très différentes, ce qui a pour conséquence de limiter les traitements effectivement possibles sur les données chiffrées.

4.1 Le chiffrement homomorphe

Le chiffrement homomorphe permet de déléguer des calculs sur des données chiffrées à une tierce personne. Cela ouvre un nombre incalculable d'usages, notamment à l'heure de l'informatique dans les nuages. Les premiers à avoir pressenti l'énorme potentiel du calcul direct sur les données chiffrées sont Rivest, Adleman et Dertouzos, qui introduisent la notion d'*homomorphisme confidentiel* [RAD78]. Ils imaginent un schéma où il serait possible de calculer sur les données chiffrées une fonction publique, mais sans rien révéler ni des entrées, ni de la sortie de la fonction évaluée, et les multiples applications de celui-ci. De manière plus formelle, un tel schéma est défini avec la syntaxe suivante.

— L'algorithme de génération des paramètres $\mathsf{HE.PPGen}$ prend en entrée 1^λ et retourne les paramètres publics pp.
— L'algorithme de génération des clés $\mathsf{HE.KeyGen}$ prend en entrée les paramètres publics pp et retourne une clé privée sk et une clé publique pk.
— L'algorithme de chiffrement $\mathsf{HE.Enc}$ prend en entrée les paramètres publics pp, une clé publique pk et un message M et retourne un chiffré $c = \mathsf{HE.Enc}(\mathsf{pp}, \mathsf{pk}, M)$.

— L'algorithme d'évaluation $\mathsf{HE.Eval}$ prend en entrée les paramètres publics pp la clé publique pk, une fonction f et un vecteur de chiffrés $(c_0, \dots c_k)$ chiffrant respectivement les messages $(M_0, \cdots M_k)$. Il retourne un chiffré $c = \mathsf{HE.Enc}(\mathsf{pp}, \mathsf{pk}, f(M_0, \dots, M_k))$.
— L'algorithme de déchiffrement $\mathsf{HE.Dec}$ prend en entrée les paramètres publics pp, la clé privée sk et un chiffré c. Il retourne un message $M = \mathsf{HE.Dec}(\mathsf{pp}, \mathsf{sk}, c)$.

On classifie les schémas de chiffrement homomorphe en fonction des catégories de fonction f que le schéma est capable d'évaluer correctement. Notons que quel que soit le schéma, le domaine de f est limité à l'espace des messages, qui pour l'ensemble des schémas actuels est un groupe de type $\mathbb{Z}_p$.

— Les schémas *partiellement homomorphes* ne peuvent évaluer qu'un nombre limité de fonctions. On parle d'homomorphisme additif pour les schémas pouvant évaluer uniquement des additions, comme le protocole de Paillier [Pai99] ou une variante d'ElGamal [CGS97], et d'homomorphisme multiplicatif pour ceux pouvant évaluer des multiplications sur les chiffrés, comme ElGamal (voir section 3.3.1) ou RSA [Gam84, RSA78]. Le schéma proposé dans [BGN05] peut évaluer un nombre arbitraire d'additions, ainsi qu'une unique multiplication. Un schéma permettant l'évaluation de multiplications et d'un couplage a été proposé par Castagnos et Laguillaumie [CL12].
— Les schémas *quelque peu homomorphes* (en anglais, *somewhat homomorphic*) peuvent évaluer sur les chiffrés des polynômes de petit degré, ce dernier étant fixé par les paramètres du schéma utilisé. Ils ont donc également des capacités calculatoires limitées, mais, contrairement aux schémas partiellement homomorphes, ils peuvent mélanger des additions et des multiplications et être transformés en des schémas totalement homomorphes.
— Les schémas *homomorphes par niveaux* (en anglais, *leveled homomorphic*) présentent moins de restrictions sur le nombre de multiplications évaluable. En effet, pour tout entier n, il est possible de fixer les paramètres d'un tel schéma pour évaluer un polynôme de degré n. Leur principal inconvénient est que la taille des paramètres utilisés pour le schéma grandit linéairement avec n.
— Les schémas *totalement homomorphes* (en anglais *fully homomorphic*) sont des schémas pouvant évaluer des polynômes dont le degré est arbitraire sur les données chiffrées. Ils sont construits à partir des deux types de schémas précédents.

Dans la suite de ce chapitre, nous nous intéressons particulièrement à ces trois derniers types de schémas.

4.1.1 Un panorama des schémas actuels

Nous présentons ici les principaux schémas quelque peu homomorphes et homomorphes par niveaux car ce sont ceux qui permettent la construction de schémas totalement homomorphes.

Les schémas quelque peu homomorphes

La première construction d'un schéma quelque peu homomorphe, basée sur les réseaux idéaux, à été présentée par Gentry [Gen09b]. Si cette construction n'a aujourd'hui plus aucun intérêt pratique, largement dépassée en efficacité par les schémas qui lui ont fait suite, le cadre qu'elle a posé est pour l'instant le seul permettant d'obtenir un schéma totalement homomorphe.

En effet, les capacités calculatoires de ce schéma ne sont plus limitées par la structure du groupe dans lequel ont lieu les opérations, comme c'est le cas pour un schéma partiellement homomorphe, mais par l'ajout d'un bruit au chiffré. Ce bruit grandit au fur et à mesure des opérations, et quand il dépasse un certain seuil, il empêche le déchiffrement. Gentry a proposé une technique astucieuse de gestion de ce bruit afin de pouvoir continuer les calculs, le *bootstrapping*, que nous étudions en détails dans la section 4.1.2.

Une instanciation particulière de ce schéma a été réalisée sur les entiers, en se basant sur le problème du plus grand diviseur approximé [vDGHV10]. Celle-ci est particulièrement intéressante car les amélioriations apportées au schéma de Gentry (telles que celles décrites dans la suite du chapitre) n'ont pu être faites qu'en changeant l'hypothèse de sécurité. Or, ces mêmes améliorations ont pu être réalisées sur le schéma de [vDGHV10] en conservant le problème du plus grand diviseur approximé (et ses variantes) [CMNT11, CNT12, CCK+13, CLT14]. Ceci peut s'expliquer par les réductions entre le problème Learning with Errors et le problème du PGCD approximé exposées dans [CS15].

Les schémas homomorphes par niveaux

Un schéma de chiffrement homomorphe est dit *homomorphe par niveaux* s'il respecte la définition suivante.

Définition 4.1 (Famille de schémas homomorphes par niveaux [BV11a])**.** *Un schéma de chiffrement homomorphe par niveaux est un schéma homomorphe dont la procédure de génération de clés* FHE.KeyGen *prend une entrée supplémentaire $L \in \mathbb{N}$. La procédure* FHE.Eval *associée peut alors évaluer correctement des polynômes de degré L, de telle manière que la taille des chiffrés ne dépende pas de L.*

Le premier de ces schémas [BV11a] fait reposer sa sécurité sur le problème LWE (Learning With Errors) introduit par Oded Regev en 2005 [Reg05]. Afin d'en améliorer l'efficacité, plusieurs versions basées sur la version anneau du problème LWE [LPR10] ont été proposées, par exemple [BV11b, BGV12].

Une lignée de schémas homomorphes par niveaux [LTV12, BLLN13] repose, elle, sur le problème NTRU. Néanmoins, de récentes attaques sur ce problème posent la question de la sécurité de ces schémas à long terme [ABD16].

Deux approches sont possibles afin de pouvoir évaluer des polynôme d'un degré fixé. La première est le changement de modulo. Introduite par [BGV12], elle consiste à changer de modulo au fur et à mesure des opérations homomorphes. Pour évaluer un polynôme de degré L, il est nécessaire d'ajouter une suite de L modulos à la clé publique.

Afin d'éviter cet ajout, [Bra12] introduit la technique de l'invariance de l'échelle. Grâce à cette technique, les calculs se font tous sur le même modulo. Le schéma FV [FV12] est ainsi la version invariante de [BGV12] et YASHE [BLLN13] est la version invariante de [LTV12]. Une comparaison de l'efficacité de ces deux versions invariantes a été effectuée dans [LN14].

Enfin, le schéma [GSW13] repose également sur le problème LWE, mais a des propriétés uniques. En effet, la multiplication homomorphe est une simple multiplication de matrice et ne nécessite pas de changement de clé. De plus, nous verrons également que le bruit présent dans ce schéma se comporte différemment de celui des autres schémas.

4.1.2 Calculs sur les données chiffrées

Comme expliqué dans [Gen14], les calculs réalisables grâce au chiffrement homomorphe peuvent être modélisés sous la forme de circuits arithmétiques. Ce modèle permet essentiellement de calculer des machines de Turing. En effet, toute machine de Turing à n étapes peut être calculée avec un circuit de taille $O(n \log(n))$ [Gen14].

Circuits arithmétiques

Dans cette section nous donnons les définitions de base liées aux circuits arithmétiques.

Définition 4.2 (Circuit Arithmétique)**.** *Un circuit arithmétique $\mathcal{C} = (\mathcal{G}, \mathcal{W})$ est un graphe orienté acyclique et un ensemble de n variables $X = \{x_1, x_2, \ldots, x_n\}$ sur un domaine $\mathbb{D}$. Les sommets $\mathcal{G}$ de $\mathcal{C}$ sont appelés* portes*. Les arcs $\mathcal{W}$ de $\mathcal{C}$ sont appelés* fils*. Une porte de degré*

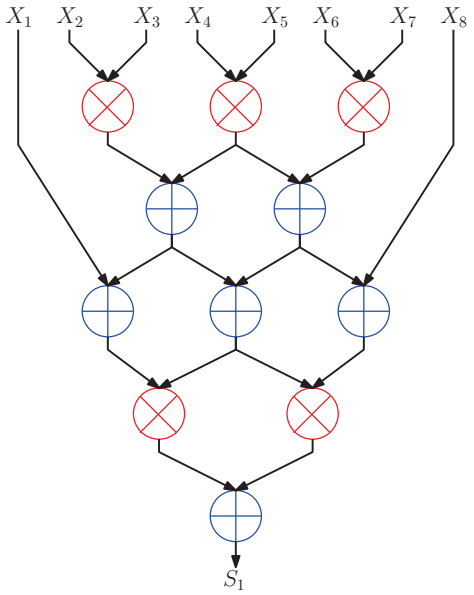


Figure 4.1 – Exemple de circuit arithmétique constitué de portes multiplicatives et additives.

entrant 0 est une porte-entrée *et elle est étiquetée avec une variable de X ou un élément de $\mathbb{R}$. Les autres portes ont un degré entrant égal à 2, et sont étiquetés soit par $\times$, soit par $+$. Ces portes sont respectivement appelées* porte multiplicative *et* porte additive*. Toute porte de degré sortant égal à 0 est nommée* porte-sortie.

Dans le cas d'un circuit défini sur $\mathbb{F}_2$, nous parlons de *circuit booléen*. Pour ce type de circuit uniquement, nous considérons des portes de degré entrant 1, étiquetées `NOT` et nommées *porte-`NOT`*.

Hormis les restrictions sur les degrés entrants des portes, nous pouvons voir que la définition d'un circuit reste très proche de celle d'un graphe orienté acyclique (voir chapitre 2). Ainsi, nous extrapolons aux circuits la notion de chemin dans un graphe. Néanmoins, dans les circuits arithmétiques, ce n'est pas la notion d'ordre du chemin dans le graphe associé qui sera importante par la suite, mais celle d'*ordre multiplicatif*.

Définition 4.3 (Ordre multiplicatif)**.** *Soit $\mathcal{C} = (\mathcal{G}, \mathcal{W})$ un circuit arithmétique, et soit $\mathcal{P}$ un chemin dans $\mathcal{C}$. L'*ordre multiplicatif *de $\mathcal{P}$ est le nombre de portes multiplicatives présentes sur $\mathcal{P}$.*

Définition 4.4 (Profondeur multiplicative)**.** *Soit $\mathcal{C} = (\mathcal{G}, \mathcal{W})$ un circuit arithmétique. La* profondeur multiplicative *de $\mathcal{C}$ est l'ordre multiplicatif maximal pris sur l'ensemble des chemins de $\mathcal{C}$*

Exemple 1 : La figure 4.1 représente un circuit qui prend en entrée les variables $\{X_1, X_2, \cdots, X_8\}$ et qui calcule le polynôme $S_1 = (X_2X_3 + 2X_4X_5 + X_6X_7) \cdot (X_1 + X_2X_3 + 2X_4X_5 + X_6X_7 + X_8)$. La profondeur multiplicative de ce circuit est 2.

Bruit et bootstrapping

Comme nous l'avons signalé dans la section 4.1.1, les chiffrés homomorphes incluent, dans l'ensemble des schémas actuels, un bruit qui grandit au fur et à mesure des opérations, jusqu'à rendre le déchiffrement impossible. Pour obtenir un schéma totalement homomorphe, les chiffrés doivent subir une procédure particulière : le *bootstrapping*, ou *réamorçage*. À ce jour, il s'agit de la seule manière d'obtenir un schéma de chiffrement totalement homomorphe.

Remarque 2. Le mot anglais *bootstrapping* est issu du nom *bootstrap*, ou tirants de bottes, qui sont des lanières placées sur les bottes pour aider à les enfiler ou les enlever. Sa première utilisation en informatique désigne le lancement d'un système à partir d'un ordinateur éteint. Cette utilisation fait référence à la légende du baron de Münchhausen. Celui-ci s'étant enfoncé dans une flaque de boue avec son cheval, il serra très fort son cheval entre ses genoux et tira sur ses *bootstraps* pour se désembourber. La traduction est malaisée, le français ne disposant pas d'un nom équivalent à *bootstrap* qui permettrait de créer facilement un verbe. D'ailleurs, dans la traduction française de l'histoire, le baron utilise ses lacets, ou ses cheveux, pour se tirer d'affaire. La traduction habituellement retenue en informatique est *amorçage*, en référence au démarrage d'un système. Dans le cadre du chiffrement homomorphe et de ce manuscrit, nous emploierons désormais le terme *réamorçage*, car cela nous semble mieux porter l'idée d'un rafraîchissement du chiffré, qui permet de *re*lancer le système, plutôt que de le lancer. Malheureusement, toute image a disparu...

La procédure de réamorçage consiste essentiellement à déchiffrer le chiffré à l'intérieur de la machine homomorphe. Il est en effet possible de fournir à la procédure $\mathsf{FHE.Eval}$ un chiffré de la clé privée de l'utilisateur, le circuit de déchiffrement du schéma, ainsi qu'un chiffré. Le résultat de cette procédure est alors un nouveau chiffré du même message (car $\mathsf{FHE.Eval}$ renvoie toujours un chiffré), mais qui aura été *rafraîchi* par le circuit de déchiffrement. Cette technique n'est possible qu'au prix d'une hypothèse supplémentaire pour la sécurité du schéma, connue sous le nom d'hypothèse de *sécurité circulaire*. Celle-ci consiste à supposer qu'il est sûr de fournir un chiffrement d'une clé privée sous sa propre clé publique. Il s'agit d'une hypothèse assez forte car elle n'est pas impliquée par la notion classique d'indistinguabilité.

Afin de mieux comprendre à quels moments durant l'évaluation cette procédure de réamorçage est nécessaire, nous présentons ici un modèle de croissance du bruit selon les opérations et les schémas. Nous représentons le niveau de bruit l_i d'un chiffré c_i par un entier. Si un chiffré c_i est la sortie de l'algorithme $\mathsf{FHE.Enc}$ et n'a pas encore subi d'évaluation d'opération, son niveau est de 1. Ensuite, son niveau augmente au fur et à mesure des opérations exécutées lors de la procédure $\mathsf{FHE.Eval}$.

Le bruit induit par les portes additives est logarithmique par rapport à celui généré par les portes multiplicatives. Dans la plupart des applications, il peut être considéré comme négligeable face à celui induit par les portes multiplicatives. Ainsi, soit c_1 et c_2 deux chiffrés de niveau l_1 et l_2 et $c_3 = \mathsf{FHE.Eval}(c_1 + c_2)$. Le niveau l_3 de c_3 est alors $l_3 = \max(l_1, l_2)$.

L'effet d'une porte multiplicative sur le niveau de bruit divise les schémas de chiffrement homomorphe en trois types. Soit c_1 et c_2 deux chiffrés de niveau respectifs l_1 et l_2 et $c_3 = \mathsf{FHE.Eval}(c_1 \times c_2)$.

— Les schémas *exponentiels*, qui sont les schémas à modulo unique, parmi lesquels [Gen09b, vDGHV10, CMNT11, CCK+13, CLT14]. Dans ces schémas, $l_3 = l_1 + l_2$. Ainsi, l'évaluation d'un circuit de profondeur multiplicative D requiert que le niveau maximal que le schéma puisse gérer soit de 2^D. Cela devient très vite prohibitif, notamment du point de vue de la taille des paramètres, et en pratique le niveau maximal proposé par ces schémas est 2.
— Les schémas *linéaires* sont les schémas qui utilisent une suite de modulos, par exemple [BGV12, CNT12, LTV12]. Dans ces schémas, $l_3 = \max(l_1, l_2) + 1$.
— Les schémas *asymétriques* [GSW13]. Dans ce cas, $l_3 = l_1 + 1$.

Quand ce niveau atteint un seuil $l_{\max}$, qui dépend des paramètres du schéma considéré, le déchiffrement devient impossible. Il faut donc procéder avant ce seuil au réamorçage. Notons que le réamorçage ne remet pas le niveau de bruit d'un chiffré à 1, mais à une certaine constante, fixée par le schéma. On ne peut obtenir un schéma totalement homomorphe qu'à partir de schémas dits réamorçables [Gen09a].

Définition 4.5 (Schéma réamorçable)**.** *Un schéma homomorphe (que ce soit partiellement/ quelque peu/par niveaux) est dit* réamorçable *s'il est capable d'évaluer correctement :*

— *son propre circuit de déchiffrement ;*

— *une porte multiplicative supplémentaire.*

À ce jour, il n'existe pas de schéma de chiffrement partiellement homomorphe qui soit réamorçable. Pour obtenir un schéma réamorçable à partir d'un schéma quelque peu homomorphe, il est nécessaire de passer par une étape de *squashing* [Gen09b], qui réduit drastiquement la profondeur multiplicative du circuit de déchiffrement du schéma. Cette étape se fait néanmmoins au prix d'hypothèses supplémentaires, comme la somme de sous-ensembles peu denses (sparse subset sum problem). Afin d'assurer la sécurité du schéma sous ces nouvelles hypothèses, il est souvent nécessaire d'augmenter la taille des paramètres [Lee11].

Enfin, dans le cas des schémas homomorphes par niveaux, l'utilisateur peut directement fixer les paramètres du schéma, et notamment la profondeur maximale L, pour pouvoir évaluer correctement le circuit de déchiffrement et une porte multiplicative supplémentaire, et obtenir ainsi un schéma réamorçable. Notons qu'il pourrait également fixer les paramètres pour évaluer un circuit de profondeur connue, mais le réamorçage reste indispensable soit pour évaluer des circuits de profondeur multiplicative importante, qui génèreraient des paramètres publics et des chiffrés trop gros, ou encore si les opérations à effectuer sur les chiffrés ne sont pas connues à l'avance.

4.1.3 Implantations et performances

Malheureusement, ce formidable enrichissement en fonctionnalités du chiffrement classique se paie au prix fort en terme d'efficacité. La première implantation proposée par Gentry et Halevi [GH11] fonctionne pour un paramètre de sécurité maximal de 72, générer les clés met plus de deux heures et un réamorçage, nécessaire après chaque multiplication, nécessite 30 minutes.

Depuis, de très nombreux schémas ont été implantés, avec une efficacité croissante [CMNT11, PBS11, CNT12, GHS12, FSF+13, BLLN13, LN14, HS14, HS15, LP16]. Néanmoins, parmi ces implantations, peu sont publiques [PBS11, LN14, HS14, HS15, LP16]. Les tests présentés dans ce mémoire utilisent la librairie HELib [HS14, HS15].

Quant aux réamorçages, si Ducas et Micciancio ont proposé une implantation en moins d'une seconde, leur procédure est applicable uniquement au schéma [GSW13]. De plus, elle ne concerne que les chiffrés d'un message d'un seul bit [DM15]. La procédure implantée dans HElib prend, elle, environ 6 minutes, mais pour un espace de clairs beaucoup plus grand et des chiffrés regroupant des vecteurs de messages [HS15]. Le coût amorti par bit d'un réamorçage exécuté sous HELib est à peu près du même ordre que celui de [DM15]. Cette technique a été améliorée par [CGGI16], arrivant à un réamorçage en 0.1 seconde, mais toujours avec les mêmes restrictions.

4.2 Algorithmique des données chiffrées

Dans cette section, nous décrivons les difficultés rencontrées lorsqu'on veut appliquer des algorithmes à des données chiffrées. Il est en effet souvent requis de les réécrire afin qu'ils soient compatibles avec la procédure FHE.Eval.

4.2.1 Boucles et conditions

Le problème le plus courant lorsqu'un utilisateur veut exécuter un algorithme sur des données chiffrées est la présence de conditions dépendant de données chiffrées. En effet, si l'on peut calculer des formules booléennes sur des bits chiffrés pour évaluer une condition, il est impossible d'utiliser le résultat (qui est chiffré) pour modifier le comportement de l'algorithme. De plus, cette impossibilité ne vaut pas seulement en l'état actuel des connaissances, mais semble inhérente à la nature même du chiffrement homomorphe. Si un programme a un comportement totalement différent selon une condition donnée (dépendante de données chiffrées), et que l'on puisse publiquement orienter l'exécution du programme en fonction de la valeur de la condition,

un adversaire n'aurait plus aucun mal à casser l'indistinguabilité à clairs choisis du schéma de chiffrement homomorphe utilisé. Il lui suffirait simplement d'évaluer (par exemple) le programme décrit par l'algorithme 4.1 sur des bits chiffrés pour savoir s'ils valent 0 ou 1.

Algorithme 4.1 Limites de l'exécution directe de programme sur des données chiffrées.

Entrée : c_b le chiffré d'un bit donné en challenge à un adversaire $\mathcal{A}$ contre le jeu IND-CPA d'un schéma FHE.
Sortie : $\mathcal{A}$ peut deviner la valeur chiffrée par c_b avec un avantage non-négligeable

```
si (c_b ∧ FHE.Enc(1)) == FHE.Enc(0) alors
    retourner 0
sinon
    retourner 1
fin si
```

Il faut donc nécessairement évaluer l'intégralité des branches du programme quand leur exécution est censée dépendre des données chiffrées. De même, une condition d'arrêt de boucle ne peut dépendre des données chiffrées : le programme doit exécuter le nombre maximal possible de tours de boucle. Cela a pour conséquence directe d'augmenter la complexité des algorithmes exécutés sur des données chiffrées. En effet, il est impossible d'exploiter la structure des données sous-jacentes pour résoudre un problème plus vite : la personne exécutant l'algorithme ne doit rien apprendre sur les données claires. Par exemple, trier une liste aléatoire ou une liste déjà triée doit prendre le même temps. Cela signifie que la complexité des algorithmes évalués doit être calculée dans le pire cas. En pratique, malgré l'amélioration notable des performances des implantations de chiffrement homomorphe, cela obère fortement les temps d'exécution sur les données chiffrées.

Nous donnons ici les deux pratiques les plus répandues pour contourner ce problème et pouvoir évaluer des conditions (de type si ... alors ...) et des terminaisons de boucles dépendant des données chiffrées [FSF+13]. Néanmoins, comme indiqué précédemment, cela vient au prix d'une perte d'efficacité non-négligeable. Ces questions ont également été étudiées dans [CDS15], où les auteurs proposent un compilateur transformant un programme écrit pour des entrées en clair en un programme pouvant être évalué sur des entrées chiffrées.

Évaluation de conditions

Pour évaluer une condition de type si p alors q sinon r, où p est un prédicat sur des données chiffrées, nous évaluons simplement l'expression suivante :

$$p \cdot q + (1 - p) \cdot r.$$

Cette transformation ne fait plus fuir d'information sur la branche du programme retournée en sortie. En effet, les deux branches sont exécutées. De plus, le prédicat p étant évalué sur des données chiffrées, le bit de résultat est également chiffré. C'est donc également le cas de $1 - p$. Or la multiplication de données chiffrées avec d'autres valeurs - que ces dernières soient des constantes ou également des chiffrés - retourne un résultat chiffré. Seul le possesseur de la clé secrète peut maintenant avoir accès au résultat du calcul. Par exemple, cette transformation appliquée au programme décrit par l'algorithme 4.1 retourne soit le chiffré du bit $c_b \wedge \mathsf{FHE.Enc}(1)$ soit celui du bit $1 - (c_b \wedge \mathsf{FHE.Enc}(1))$. À la condition que le schéma soit IND-CPA, ceci ne révèle aucune information sur les données claires.

Terminaison de boucles

Enfin, il reste le problème de la terminaison de boucle quand la condition dépend des données chiffrées. Comme précédemment, on cherche à éviter que le comportement du programme exécuté sur les données chiffrées diffère selon les entrées de celui-ci. Dans le même esprit que le programme

décrit plus haut, on peut par exemple voir directement les conséquences d'un programme composé d'une boucle while $c \neq \mathsf{FHE.Enc}(0)$: il suffirait à un adversaire de surveiller le temps d'exécution pour déduire la valeur de c. Nous cherchons donc à avoir des boucles dont la taille soit indépendante des données chiffrées pour assurer un temps d'exécution constant. Pour cela, nous évaluons le nombre de tours de boucles maximal pire cas en clair, et nous rendons la boucle stationnaire dès lors que la condition de terminaison est remplie en utilisant les instructions conditionnelles décrites à la section précédente.

4.2.2 Un cas d'usage : les réseaux de tri

Le fait de ne pas pouvoir exploiter la structure sous-jacente des données en clair et donc que la complexité des algorithmes évalués sur les données est toujours celle du pire cas rend impossible l'utilisation des nombreuses améliorations algorithmiques qui ont été réalisées en clair. Par exemple, pour trier des données chiffrées, l'algorithme classique de tri le plus efficace est assez contre intuitivement le tri à bulles. Afin d'améliorer la complexité des algorithmes à évaluer, il ne faut pas chercher des algorithmes classiques, où le but est souvent d'améliorer la complexité dans le cas moyen, mais il faut chercher des alternatives où le nombre d'opérations effectuées dépende le moins possible des données claires.

Dans [FSF+13] et [MFF+13], les auteurs se posent la question du tri de données chiffrées. Ils font déjà la remarque de la complexité en pire-cas des algorithmes, et concluent en utilisant le tri à bulles. Nous proposons ici une autre piste, les réseaux de tri, dont le tri à bulles est un cas particulier. Le principal avantage des réseaux de tri est que le nombre de comparaisons à effectuer pour trier une liste ne dépend pas de l'état préalable de la liste. Ainsi, la complexité de l'algorithme est constante : la complexité pire-cas n'est pas pire que la complexité meilleur-cas. Le réseau de tri le plus efficace est le Odd-Even-Merge [Bat68].

Ce tri consiste en trois phases distinctes. Il ne permet de trier que des listes dont la longueur est exactement une puissance de deux et travaille par dichotomie. La première phase est la phase paire qui consiste à comparer tous les éléments d'indice pair. La deuxième phase est la phase impaire, et, symétriquement, consiste à comparer tous les éléments d'indices impairs. Enfin, on opère une phase de fusion en comparant les éléments d'indices $2i$ avec ceux d'indices $2i+1$ où i varie de 0 à la longueur de la liste à trier (moins un). Pour une liste de taille n, la complexité de cet algorithme est $O(n(\log(n))^2)$, à comparer avec la complexité du tri à bulles qui est $O(n^2)$.

Exemple 2 : Nous donnons figure 4.2 un exemple du tri Odd-Even-Merge appliqué à la liste $(2, 5, 4, 0, 3, 6, 7, 1)$. Les arêtes verticales représentent les comparateurs : une entrée (x, y) donne la sortie (x', y') où $x' = \min(x, y)$ et $y' = \max(x, y)$. Pour une liste de 8 éléments, le tri Odd-Even-Merge demande 19 comparaisons, alors que le tri à bulles en demande 28.

4.3 Le chiffrement fonctionnel

Le chiffrement fonctionnel [BSW11] répond à des cas d'usages différent de ceux du chiffrement homomorphe. En effet, pour de nombreux problèmes nécessitant des calculs sur des données chiffrées, le chiffrement homomorphe est limité par le fait que le résultat du calcul effectué est chiffré. De plus, l'évaluation de fonctions est une procédure publique, qui peut donc être exécutée par n'importe qui. Le chiffrement fonctionnel répond en quelque sorte au problème miroir. Le propriétaire de la clé secrète peut générer une clé d'évaluation permettant d'évaluer une fonction précise. Il faut nécessairement être en possession de cette clé pour évaluer la fonction. Néanmoins, contrairement au chiffrement homomorphe, le résultat de la fonction est obtenu en clair par l'agent qui effectue l'évaluation.

Par exemple, considérons un système de luttes contre les fraudes à la carte bancaire. Si les transactions bancaires sont chiffrées sous la clé publique du propriétaire de la carte, un logiciel de détection de fraudes pourrait chercher des transactions suspectes, par exemple dépassant un

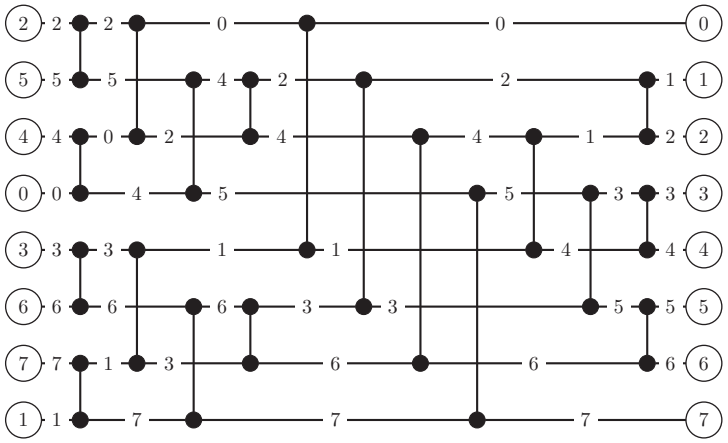


Figure 4.2 – Le tri Odd-Even-Merge.

certain montant, ou dans une aire géographique donnée, sans que ne soit divulguée l'intégralité des transactions de la personne.

Le chiffrement fonctionnel recouvre différents types de chiffrements. Une forme très simple de ce type de schéma est le chiffrement basé sur l'identité [Sha84, Coc01, BF01] ou sur les attributs [SW05]. L'émetteur du message le chiffre pour une certaine catégorie de personnes et seules les personnes appartenant à cette catégorie peuvent déchiffrer le message. Une classe plus large de chiffrement fonctionnel est le chiffrement par prédicat [BW07, SBC+07]. On attribue au message à chiffrer un vecteur d'attributs x et on fournit une clé secrète dépendante d'une fonction f qui permet le déchiffrement si et seulement si $f(x) = 1$. Dans les autres cas, rien n'est divulgué, ni à propos du message, ni à propos des attributs. De nombreux schémas permettent de réaliser ce type de chiffrement lorsque le prédicat est sous la forme d'un produit scalaire, par exemple [KSW08, AFV11].

Les progrès réalisés dans le domaine du chiffrement fonctionnel consistent surtout à élargir la classe de fonctions qu'il est possible de manipuler. Par exemple, il est possible de construire un chiffrement par attributs [GVW13, BGG+14, BV16] ou un chiffrement par prédicat pour tout circuit [GVW15]. Une autre avancée dans ce sens repose sur les obfuscateurs indistinguables [GGH+13], permettant théoriquement de réaliser le chiffrement fonctionnel sans aucune restriction sur les fonctions à évaluer. Néanmoins, ces derniers reposent sur des hypothèses de sécurité encore mal assurées, comme le montrent les nombreux travaux de cryptanalyse, notamment, [MSZ16, CGH+15]. Bien que la classe de fonctions évaluables grâce au chiffrement fonctionnel soit toujours plus large, seuls les schémas de chiffrement basés sur l'identité, et dans une moindre mesure, sur les attributs sont implémentés [Vol].

4.4 Le calcul multi-parties

Nous présentons enfin des solutions de calculs multi-parties sur des données chiffrées. La principale différence entre ces solutions et le chiffrement homomorphe ou le chiffrement fonctionnel est la présence d'interactivité. Ici, la question est moins la délégation du calcul que la collaboration entre plusieurs parties qui se méfient les unes des autres. Le but est alors d'obtenir un certain résultat sur un ensemble de données dont nous ne possédons qu'une partie, sans rien apprendre sur les données des autres membres et sans que ces derniers n'apprennent rien sur les nôtres. Des chiffrements homomorphes multi-clés ont été proposés pour essayer de résoudre ce type de problématiques, mais leur coût reste prohibitif [LTV12, BP16].

Transfert inconscient

Le transfert inconscient OT (de l'anglais *oblivious transfer*) a été introduit par Rabin en 1981 [Rab81]. C'est un protocole qui permet à un émetteur S de transmettre de manière inconsciente une information parmi d'autres à un destinataire R.

Formellement, dans un protocole 1-parmi-2-OT, l'émetteur S génère deux messages (M_0, M_1). Le transfert assure que l'émetteur reçoit un message M_b avec $b \in \{0, 1\}$, sans rien apprendre du message M_{1-b}. De son côté, S ne sait pas quel message a été sélectionné. Plusieurs réalisations de protocoles de transfert inconscient ont été proposés, d'abord à l'aide de cryptographie à clé publique [NP01]. Dans [IKNP03], les auteurs proposent une technique à base de schémas symétriques, moins coûteux que la cryptographie à clé publique. Cette technique a été depuis optimisée [ALSZ13, KK13], permettant de construire des schémas de calculs multi-parties de plus en plus efficaces.

Circuits embrouillés de Yao

Ces circuits ont été conçus par Yao [Yao82, Yao86] afin de résoudre le *problème du millionnaire*. La question est la suivante : deux millionnaires peuvent ils décider qui est le plus riche des deux de manière incontestable, sans se révéler leur fortune respective ?

Un circuit embrouillé (en anglais, *garbled circuit* [BMR90]) résout ce genre de problèmes en permettant l'évaluation conjointe de fonctions sur des entrées chiffrées. Comme pour le chiffrement homomorphe, cette fonction doit être représentée sous forme de circuit booléen.

Plus formellement, Bellare et al. [BHR12] définissent la syntaxe de ces circuits comme un algorithme Gb qui transforme une fonction $f : \{0,1\}^n \to \{0,1\}^m$ en un triplet de fonctions (F, e, d) ayant les propriétés suivantes.

- La fonction d'encodage e transforme une entrée $x \in \{0,1\}^n$ de la fonction f en une entrée brouillée X à l'aide d'un clé secrète k.
- La fonction de décodage d envoie une sortie brouillée Y du circuit sur la sortie correspondante $y \in \{0,1\}^m$ de la fonction f.
- La fonction embrouillée F prend en entrée une entrée brouillée X et renvoie une sortie brouillée Y.

Pour être correct, ce protocole doit vérifier l'équation suivante : $f(x) = d(F(X))$. La principale propriété de sécurité attendue d'un tel schéma est la *confidentialité*. De manière informelle, celle-ci signifie qu'un adversaire ayant accès au triplet (F, e, d) n'apprend pas plus d'informations que ce qui est révélé par la seule sortie finale $y = d(F(X))$. Les définitions formelles et les preuves de sécurité sont données dans [BHR12]. Le coût prohibitif des circuits de Yao en a longtemps fait un outil uniquement théorique. Néanmoins, de nombreux travaux en ont récemment grandement amélioré l'efficacité, par exemple, [MGBF14, ZRE15, MGC+16]. Cette technique de calcul sur les données chiffrées s'est prouvée féconde, permettant de nombreuses applications, comme l'intersection d'ensemble privés [HEK12], des tests biométriques [HMEK11], ou encore, combinés à un protocole de transfert inconscient, la détection d'intrusions sur du trafic chiffré [SLPR15].

Chapitre 5

Minimisation du nombre de réamorçages

Sommaire

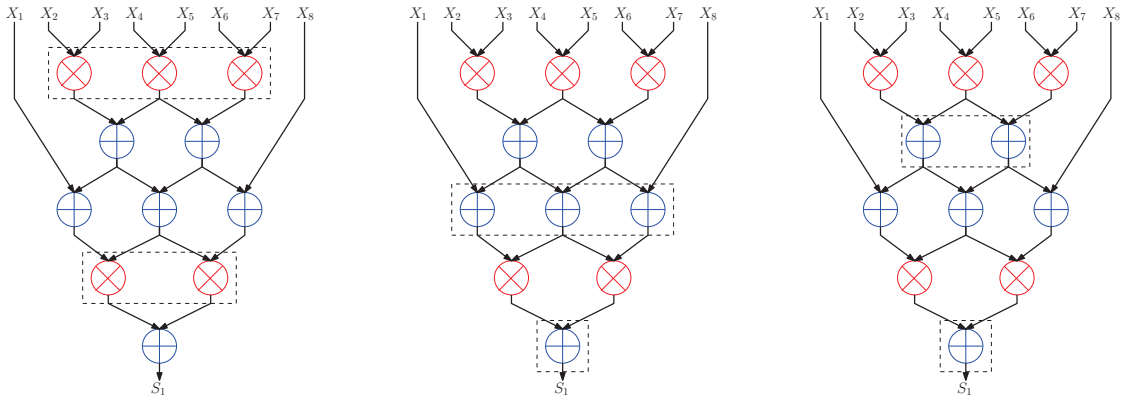
Comme nous l'avons vu au chapitre 4, le réamorçage est à ce jour l'unique méthode connue pour obtenir un schéma totalement homomorphe. Si le temps d'exécution de cette procédure a été grandement amélioré pour certains schémas totalement homomorphes [DM15, CGGI16], le réamorçage reste une procédure coûteuse pour les schémas utilisés en pratique. Un moyen simple mais efficace pour améliorer les temps d'évaluation homomorphe d'une fonction est donc de n'y faire appel qu'un nombre minimal de fois.

Dans ce chapitre, nous proposons une définition formelle du problème de minimisation des réamorçages dans un circuit, puis nous fournissons une analyse théorique de la complexité du problème. Enfin, nous proposons une méthode pratique de recherche de solutions et montrons son efficacité à l'aide d'expérimentations. Ces résultats sont issus de l'article [PV15], publié à SAC 2015.

5.1 Définition du problème et premières heuristiques

Dans cette section, nous définissons formellement le problème de la minimisation du nombre de réamorçages et nous montrons les limites des heuristiques naïves qui peuvent être utilisées pour le résoudre.

De manière informelle, le problème de la minimisation du nombre de réamorçages associés à un niveau $l_{\max}$ consiste à trouver un sous-ensemble de portes du circuit tel que tout chemin de profondeur multiplicative $l_{\max} - 1$ ait avec celui-ci une intersection non vide. Cela permet de garantir la correction de l'évaluation du circuit, car le niveau de bruit ne dépasse pas le seuil $l_{\max}$. Afin d'améliorer l'efficacité, nous cherchons à obtenir un sous-ensemble de ce type de



(a) L'heuristique 1 donne une solution de 5 réamorçages.

(b) L'heuristique 2 donne une solution de 4 réamorçages.

(c) La solution optimale est de 3 réamorçages

Figure 5.1 – Limites des heuristiques pour la minimisation des réamorçages.

cardinal minimal. Ce problème peut ainsi s'écrire sous la forme d'un problème d'optimisation (voir définition 2.3) de la manière suivante.

Définition 5.1 (Problème $l_{\max}$-MB de la minimisation des réamorçages)**.** *Soit* Π *un schéma de chiffrement totalement homomorphe et* $l_{\max}$ *le niveau de bruit maximal qui y est associé.*

- *L'ensemble des instances* $\mathcal{I}$ *est un ensemble de circuits.*
- *Soit* $\mathcal{C} = (\mathcal{G}, \mathcal{W}) \in \mathcal{I}$*, une solution* $S \in \mathsf{Sol}(\mathcal{C})$ *est un sous-ensemble de* $\mathcal{G}$ *tel que tout chemin d'ordre multiplicatif* $l_{\max} - 1$ *ait au moins une porte dans* S*.*
- *La valeur de* S *est son cardinal :* $m(S, \mathcal{C}) = |S|$*.*
- $\mathsf{but} = \min$*.*

Nous notons D-$l_{\max}$-MB le problème de décision associé. Dans la suite de ce chapitre, nous prouvons la NP-complétude de D-$l_{\max}$-MB pour $l_{\max} \geq 3$. De plus, nous donnons une méthode permettant d'obtenir une solution optimale ou une bonne approximation de cette solution selon le circuit d'entrée. Les algorithmes naïfs pour résoudre ce problème donnent rarement des solutions optimales. Par exemple, les deux heuristiques suivantes ont été utilisées pour placer les réamorçages sur un circuit.

- *Heuristique 1* [GHS12] Cette méthode consiste à évaluer le niveau de bruit d'un chiffré et de le réamorcer dès que son niveau atteint $l_{\max}$. En pratique, cela revient à placer les réamorçages juste après les portes multiplicatives. Cette heuristique était la première utilisée dans les schémas de la première génération, quand une seule multiplication pouvait être évaluée entre deux réamorçages.
- *Heuristique 2* [GH11] Une autre stratégie consiste à attendre le plus longtemps possible après que le niveau ait atteint $l_{\max}$. Le chiffré est alors réamorcé juste avant la porte multiplicative suivante.

Comme nous pouvons le voir sur l'exemple simple donné figure 5.1, les solutions données par ces heuristiques ne sont pas toujours optimales.

Une méthode de résolution de ce problème a déjà été étudiée dans l'article de Lepoint et Paillier [LP13]. Ils définissent un modèle de croissance du bruit et utilisent le problème SAT pour résoudre des formules booléennes leur donnant l'emplacement des réamorçages. Leur étude se concentre essentiellement sur les schémas dit exponentiels (voir chapitre précédent) pour $l_{\max}$ petit (2,4), même si elle peut s'adapter pour des $l_{\max}$ plus grands. Pour s'adapter aux schémas linéaires, les auteurs modifient le circuit évalué afin d'appliquer leur algorithme comme une boîte noire. Aucune étude théorique de l'efficacité n'est donnée, et les évaluations pratiques ne sont menées que pour $l_{\max} = 2$ et $l_{\max} = 4$.

Nous proposons dans ce chapitre deux types de contributions. La première est théorique, en

donnant des résultats de complexité sur le problème de minimisation du nombre de réamorçages. Nous prouvons que celui-ci est polynomial pour $l_{\max} = 2$ et NP-complet pour $l_{\max} \geq 3$. La seconde est pratique en proposant une méthode pour déterminer le nombre et la place des réamorçages nécessaires pour un circuit donné. Notre méthode s'appuie sur la programmation linéaire et est plus flexible que celle proposée dans [LP13]. En effet, elle s'adapte directement à tous les types de schéma existants et prend en compte un grand nombre d'opérations homomorphes.

5.2 Complexité du problème dans le cas $l_{\max} = 2$

Ce cas concerne principalement les schémas à croissance de bruit exponentielle (voir partie 4.1.2). Nous présentons dans cette section un algorithme polynomial pour la résolution du problème 2-MB.

5.2.1 Un algorithme de résolution polynomial

Notre algorithme s'appuie sur le problème de calcul de séparateurs minimaux dans les graphes décrit à la section 2.2. Soit $\mathcal{C}$ un circuit arithmétique et $G = (V, E)$ le graphe orienté acyclique associé. Nous allons exploiter le fait qu'une et une seule multiplication peut avoir lieu entre deux réamorçages. Supposons que $\mathcal{C}$ ait une profondeur multiplicative de 1. Pour résoudre le problème du 2-MB dans un tel graphe, il suffit de résoudre le problème de connectivité du graphe. Dans un graphe avec une profondeur multiplicative plus importante, nous allons découper le graphe en composantes connexes ayant chacune une profondeur multiplicative de 1.

Pour cela, la première étape est de supprimer les arcs $(u, v) \in E$ tels que v est une porte multiplicative. Ceci nous donne un graphe G'. Les composantes connexes de G' sont aussi des graphes orientés acycliques, mais elles sont construites de telle manière que leurs circuits associés ont chacun une profondeur multiplicative de 1. En ajoutant un arc de la source s de G vers chaque porte multiplicative et un arc de chaque puits des composantes de G' vers t, chaque chemin de s à t passe par une et une seule porte multiplicative. Ainsi, pour évaluer correctement le circuit $\mathcal{C}$, il suffit d'effectuer un réamorçage par chemin : c'est exactement le problème du (s, t)-séparateur d'un graphe.

L'algorithme 5.1 décrit notre solution, et un exemple d'application est donné figure 5.2.

Algorithme 5.1 Calcul du nombre minimal de réamorçages pour un circuit avec $l_{\max} = 2$.

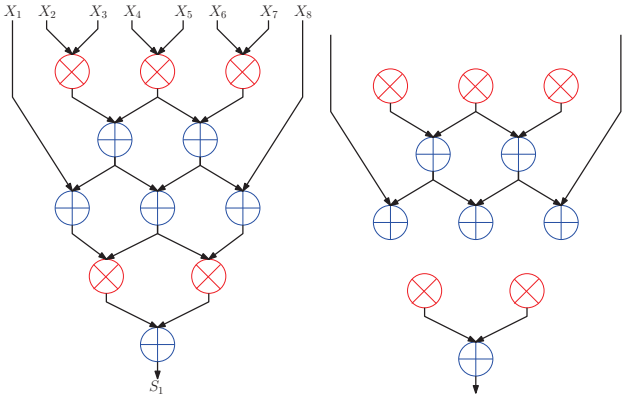
Entrée : $\mathcal{C}$ un circuit arithmétique et $G = (V, E)$ le graphe orienté acyclique associé.
Sortie : Un sous-ensemble minimal de portes dont il faut réamorcer les sorties.

1: Copier le graphe G dans un graphe $\tilde{G} = (\tilde{V}, \tilde{E})$
2: Ajouter un sommet source s et un sommet puits t à $\tilde{V}$
3: Supprimer tous les arcs (u, v) de $\tilde{E}$ tels que v soit une porte multiplicative
4: Pour toutes les portes multiplicatives v, ajouter un arc (s, v) à $\tilde{E}$
5: Pour tous les arcs (u, v) supprimés ligne 2, ajouter un arc (u, t) à $\tilde{E}$
6: Calculer un (s, t)-séparateur minimal S de $\tilde{G}$
7: **retourner** S

Théorème 5.2. *L'algorithme 5.1 est correct : il renvoie un ensemble minimal de réamorçages à effectuer pour une évaluation correcte d'un circuit $\mathcal{C}$ avec $l_{\max} = 2$.*

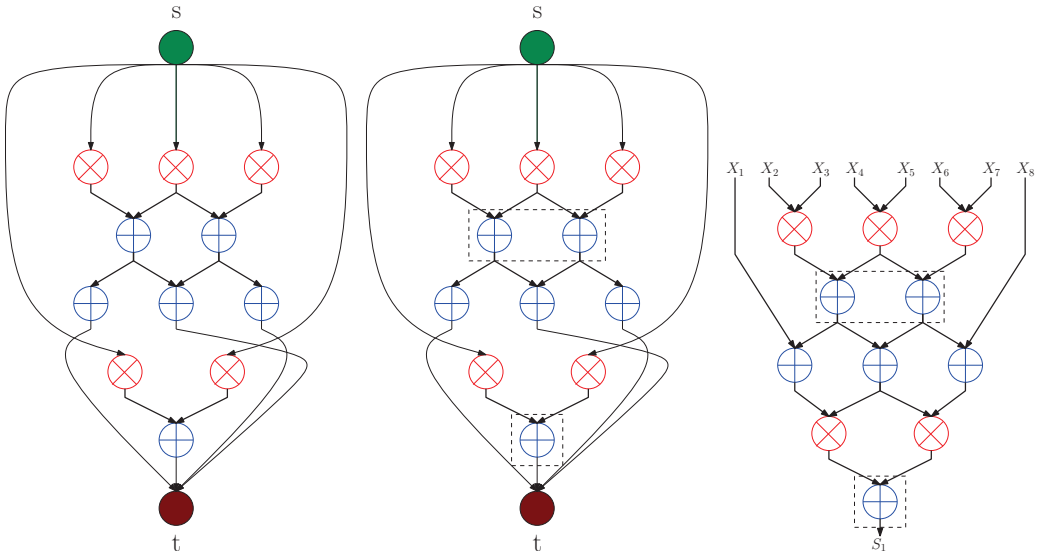
Démonstration. Soit $\mathcal{C} = (\mathcal{G}, \mathcal{W})$ un circuit arithmétique et soit $G = (V, E)$ le graphe orienté acyclique associé à $\mathcal{C}$. Nous allons d'abord prouver que l'algorithme renvoie bien une solution du 2-MB puis que cette solution est minimale.

Soit S une solution renvoyée par l'algorithme. Par définition, S est un (s, t)-séparateur de $\tilde{G}$



(a) Circuit initial $\mathcal{C}$.

(b) Suppression de tous les arcs entrant dans une porte multiplicative.



(c) Ajout d'un arc de s à toutes les portes multiplicatives et un arc de chaque puits du circuit vers le sommet t.

(d) Calcul du (s,t)-séparateur.

(e) On place les réamorçages à la sortie des portes composant le séparateur.

Figure 5.2 – Exemple d'exécution de l'algorithme 5.1 sur un circuit simple.

Supposons que S ne soit pas une solution du problème 2-MB. Il existe un chemin $\mathcal{P}$ dans le circuit $\mathcal{C}$ d'ordre multiplicatif 1 qui n'a pas de sommet dans S. Supposons que $\mathcal{P}$ soit composé de k additions en plus de la multiplication, on peut écrire :

$$\mathcal{P} = (A^{(1)}, \dots, A^{(i)}, M^{(1)}, A^{(i+1)}, \dots, A^{(k)}).$$

À l'étape 4 de l'algorithme, donc dans $\tilde{G}$, $\mathcal{P}$ donne le (s,t)-chemin suivant :

$$\mathcal{P}' = (s, M^{(1)}, A^{(i+1)}, \cdots, A^{(k)}, t).$$

$\mathcal{P}$ n'ayant pas de sommets dans S, $\mathcal{P}'$ n'en a pas non plus. S ne peut donc pas être un (s,t)-séparateur de $\tilde{G}$. Ainsi, S est bien une solution du problème 2-MB.

Montrons maintenant que S est minimale. Supposons qu'il existe un ensemble S' tel que $|S'| < |S|$. S et S' étant des solutions du 2-MB, il existe un chemin d'ordre multiplicatif 1 qui a au moins deux sommets dans S. Soit $\mathcal{P}$ un tel chemin. S étant construit en suivant l'algorithme 5.1, $\mathcal{P}$ a donc défini 2 (s,t)-chemins différents dans $\tilde{G}$.

Supposons qu'il existe u et u' différents sur $\mathcal{P}$ tels qu'on ait ajouté deux arcs (s,u) et (s,u') dans $\tilde{G}$. Cela n'est possible que si u et u' sont deux portes multiplicatives. Or $\mathcal{P}$ a un ordre multiplicatif de 1.

Supposons maintenant qu'il existe v et v' différents sur $\mathcal{P}$ tels qu'on ait ajouté deux arcs (v,t) et (v',t). Il y a ici plusieurs cas possibles. Un arc de la forme $(\cdot,t)$ n'est ajouté que pour un puits d'une composante connexe de G'. Un sommet est un puits d'une composante connexe de G' si :

— il est l'origine d'un arc dont la destination est une porte multiplicative ;
— il est un puits de G.

Étudions les combinaisons possibles.

— v et v' sont deux origines d'arcs ayant pour destinations des portes multiplicatives. Pour simplifier, considérons que v' est situé après v sur le chemin orienté $\mathcal{P}$ (c'est-à-dire qu'il existe un chemin orienté $\mathcal{P}'$ de v à v' inclus dans $\mathcal{P}$ d'ordre multiplicatif 1 - la porte multiplicative $\tilde{v}$ qui suit v). Si la porte multiplicative qui suit v' appartient à $\mathcal{P}$, il y a contradiction car $\mathcal{P}$ a alors un ordre multiplicatif d'au moins 2. Dans le cas où la porte multiplicative qui suit v' n'appartient pas à $\mathcal{P}$, considérons le chemin $\mathcal{P} \setminus \mathcal{P}'$. Celui-ci ne peut pas contenir de porte multiplicative (sinon $\mathcal{P}$ a un ordre multiplicatif d'au moins 2). Ainsi, à l'étape 3 de l'algorithme, aucun arc n'est ajouté entre s et $\mathcal{P} \setminus \mathcal{P}'$. Le seul (s,t)-chemin issu de P est donc $((s,\tilde{v}), \mathcal{P}' \setminus (v,\tilde{v}), (v',t))$.
— v et v' sont deux puits de G. Dans ce cas, ils sont confondus et sont l'extrémité de $\mathcal{P}$.
— v est l'origine d'un arc ayant pour destination une porte multiplicative et v' est un puits de G. Dans ce cas, comme ci-dessus, $\mathcal{P} \setminus \mathcal{P}'$ ne peut pas contenir de portes multiplicatives et donc ne génère pas de (s,t)-chemin.

Ainsi, l'algorithme 5.1 calcule bien une solution minimale du problème de minimisation du nombre de réamorçages pour $l_{\max} = 2$. □

Enfin, l'algorithme 5.1 s'exécute en temps polynomial.

Théorème 5.3. *L'algorithme 5.1 est polynomial, et a plus précisément une complexité en*

$$\mathcal{O}(|V||E|\log(|V|^2/|E|)).$$

Démonstration. La première et la troisième étape sont en $\mathcal{O}(|V|)$. La deuxième étape se fait en temps constant. La quatrième étape est en $\mathcal{O}(|E|)$. Le calcul d'un (s,t)-séparateur minimal se fait en $\mathcal{O}(|V||E|\log(|V|^2/|E|))$ (voir section 2.2). Ainsi la complexité de l'algorithme est $\mathcal{O}(|V||E|\log(|V|^2/|E|))$. □

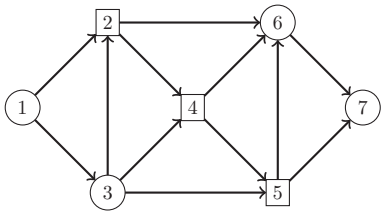


Figure 5.3 – Les sommets encadrés forment une solution du problème 3-PVCD.

Problèmes	VCD		$k-$PVCD
Instances	$D \in I_{\text{VCD}}$	$\overset{f}{\longmapsto}$	$f(D) \in I_{k-\text{PVCD}}$
Solutions	$g(D, S_{k-\text{PVCD}}) \in \text{sol}_{\text{VCD}}(D)$	$\overset{g}{\longmapsto}$	$S_{k-\text{PVCD}} \in \text{sol}_{k-\text{PVCD}}(f(D))$

Figure 5.4 – Schéma de la réduction du problème VCD vers le problème k-PVCD.

5.3 Complexité du problème pour le cas $l_{\max} \geq 3$

Dans cette section, nous prouvons que la version décisionnelle de $l_{\max}$-MB est NP-complète pour $l_{\max} \geq 3$. La preuve consiste en la réduction du problème de couverture par sommets d'un graphe orienté acyclique (connu pour être NP-complet) vers le problème de $l_{\max}$-MB. Pour obtenir cette réduction, nous utilisons un problème intermédiaire : le problème de couverture par sommets des chemins d'ordre k dans un graphe orienté acyclique, (k-PVCD, de l'anglais *k-path vertex cover in directed acyclic graphs*), qui consiste à trouver un sous-ensemble de sommets qui couvre tous les chemins orientés d'ordre k.

Ce dernier problème une version orientée du problème de couverture par sommets des chemins d'ordre k dans un graphe non-orienté (k-PVC, de l'anglais *k-path vertex cover*) introduit dans [BKKS11] où une preuve de NP-complétude est donnée. À notre connaissance aucun résultat n'existe sur la version orientée du problème de couverture par sommets des chemins d'ordre k. Nous commençons donc par donner une preuve de sa NP-complétude avant de s'attaquer au problème $l_{\max}$-MB pour $l_{\max} \geq 3$.

5.3.1 NP-complétude de k-PVCD

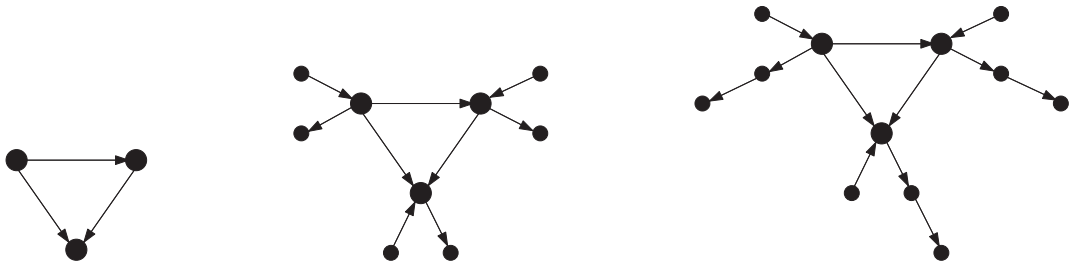
Commençons par définir formellement le problème de couverture par sommets des chemins d'ordre k dans un graphe orienté acyclique.

Définition 5.4 (k-PVCD)**.** *Le problème de couverture par sommets des chemins d'ordre k dans un graphe orienté acyclique est un quadruplet $(\mathcal{I}, \mathsf{Sol}, m, \mathsf{but})$ tel que :*

- *$\mathcal{I}$ est l'ensemble des graphes orientés acycliques,*
- *Soit $G = (V, E) \in \mathcal{I}$, une solution $S \in \mathsf{Sol}(G)$ est un sous-ensemble de sommets de V tel que tout chemin orienté d'ordre $k \geq 2$ ait au moins un sommet dans S,*
- *la valeur d'une solution S est son cardinal : $m(S, G) = |S|$,*
- *$\mathsf{but} = \min$.*

Nous notons D-k-PVCD le problème de décision associé. Un exemple de couverture par sommets des chemins d'ordre 4 dans un graphe orienté acyclique est présenté figure 5.3.

Théorème 5.5. *Le problème décisionnel de couverture par sommets des chemins d'ordre k d'un graphe orienté acyclique est NP-complet pour $k \geq 2$.*



(a) D le DAG originel. (b) $f(D)$ pour $k = 4$. (c) $f(D)$ pour $k = 5$.

Figure 5.5 – Exemple d'application de l'algorithme f sur un graphe orienté acyclique D.

Démonstration. Nous présentons une réduction du problème de couverture par sommets vers le problème de couverture par sommets des chemins d'ordre k. Cette réduction est schématisée figure 5.4.

Remarquons que D-2-PVCD est exactement D-VCD. Pour $k > 2$ nous allons construire une réduction (f, g) du problème D-VCD au problème D-k-PVCD.

Algorithme f. Soit $G = (V, E)$ un graphe orienté acyclique instance de D-VCD. Commençons par construire un algorithme f transformant un graphe orienté acyclique G, instance de D-VCD, en un graphe orienté acyclique $G' = f(G)$, instance du problème D-k-PVCD.

Pour tout sommet $u \in V$, nous ajoutons un chemin orienté de $\left\lfloor \frac{k}{2} \right\rfloor - 1$ sommets (en plus de u) tel que u soit le puits du chemin ; et un chemin orienté de $\left\lceil \frac{k}{2} \right\rceil - 1$ sommets (en plus de u) tel que u soit la source du chemin. Un exemple d'application est présenté sur la figure 5.5.

L'algorithme f parcourt tous les sommets de D une seule fois, donc l'algorithme f est polynomial avec une complexité $\mathcal{O}(|V|)$. Remarquons que le nombre de sommets de $f(D)$ est

$$|V| \cdot \left(\left\lfloor \frac{k}{2} \right\rfloor - 1 + \left\lceil \frac{k}{2} \right\rceil - 1 \right) = |V| \cdot (k - 2).$$

Dans la suite, nous appelons les sommets de D les *sommets initiaux*, et les sommets ajoutés par f les *nouveaux sommets*.

Maintenant, construisons un algorithme g transformant une solution $S_{k-\text{PVCD}}$ du problème k-PVCD dans $f(D)$ en une solution $g(D, S_{k-\text{PVCD}})$ du problème VCD dans D.

Algorithme g. Soit S une solution du problème de couverture par sommets des chemins d'ordre k dans $f(D)$. Supposons que S contienne un nouveau sommet u, c'est-à-dire qu'il existe $u \in S_{k-\text{PVCD}},\ u \notin V$. Sans prendre en compte l'orientation de graphe, on cherche le sommet initial $v \in V$ qui soit le plus proche de u. Comme le chemin entre u et v est par construction d'ordre inférieur à k, les chemins couverts par v le sont aussi par u : nous pouvons échanger u et v dans S. Nous répétons cette procédure pour tous les nouveaux sommets dans S. Soit S' l'ensemble obtenu : il ne contient que des sommets initiaux et $|S'| \leq |S|$. g retourne S'.

Montrons maintenant que S' est une solution de VCD dans G. Supposons qu'il existe un arc (x, y) dans E tel que $x \notin S'$ et $y \notin S'$. Dans $f(G)$, x est le puits d'un chemin d'ordre $\left\lfloor \frac{k}{2} \right\rfloor - 1$ (en plus de lui-même) et y est la source d'un chemin d'ordre $\left\lceil \frac{k}{2} \right\rceil - 1$ (en plus de lui-même). Ces deux chemins sont reliés par l'arc (x, y) : cela donne un chemin d'ordre k dans G'. Par construction de S', et comme ni x ni y sont dans S', ce chemin n'a pas de sommets dans S. Ceci est une contradiction avec la définition de S : S' est une couverture par sommets de G de taille au plus $|S|$.

Ainsi, le problème de couverture par sommets des chemins d'ordre k dans un graphe orienté acyclique est NP-difficile. Il reste à montrer que ce problème est effectivement dans la classe NP.

Problèmes	$k-\text{PVCD}$		$l_{\max}-\text{MB}$
Instances	$D \in I_{k-\text{PVCD}}$	$\overset{f}{\longmapsto}$	$f(D) \in I_{l_{\max}-\text{MB}}$
Solutions	$g(D, S_{l_{\max}-\text{MB}}) \in \text{sol}_{k-\text{PVCD}}(D)$	$\overset{g}{\longmapsto}$	$S_{l_{\max}-\text{MB}} \in \text{sol}_{l_{\max}-\text{MB}}(f(D))$

Figure 5.6 – Schéma de la réduction de D-k-PVCD vers D-$l_{\max}$-MB.

Pour cela il faut pouvoir vérifier qu'une solution renvoyée par un algorithme est bien une solution du problème en temps polynomial. Le nombre de chemins d'ordre k dans D est au plus $\mathcal{O}(|V|^k)$. Ainsi, toutes les solutions supposées du problème peuvent être vérifiées en temps polynomial, donc le problème D-k-PVCD appartient à la classe NP. Par conséquent, le problème de couverture par sommets des chemins d'ordre k dans un graphe orienté acyclique est NP-complet. □

5.3.2 NP-complétude de $l_{\max}$-MB

Nous présentons maintenant une réduction du problème de couverture par sommets des chemins d'ordre k sur un graphe orienté acyclique vers le problème de minimisation du nombre de réamorçages pour $l_{\max}$, avec $k = l_{\max} - 1$.

Théorème 5.6. *Le problème décisionnel de minimisation du nombre de réamorçages pour $l_{\max}$ dans un circuit arithmétique est NP-complet pour $l_{\max} \geq 3$.*

Démonstration. Soit $G = (V, E)$ un graphe orienté acyclique instance de D-$k-1$-PVCD. Nous explicitons la réduction présentée figure 5.6. Construisons d'abord l'algorithme f transformant G en une instance $f(G) = \mathcal{C}$ du problème D-$l_{\max}$-MB.

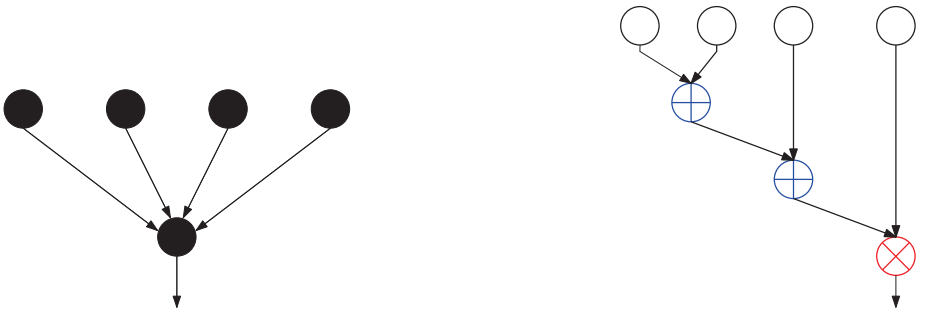
Algorithme f. Nous voulons construire un circuit arithmétique $\mathcal{C}$ à partir de G. Ce circuit n'est utile que dans le cadre de notre réduction, et il n'est pas requis qu'il évalue une fonction intéressante. En revanche, nous souhaitons que tous les chemins orientés P d'ordre $k-1$ dans G soient transformés en un chemin $\mathcal{P}$ d'ordre multiplicatif $l_{\max}$.

La principale différence entre un graphe et un circuit est la contrainte sur le degré entrant des portes d'un circuit : au plus deux. Ainsi, pour transformer les sommets de G en portes d'un circuit arithmétique $\mathcal{C} = (\mathcal{G}, \mathcal{W})$, nous allons distinguer plusieurs cas, selon leur degré entrant. Tout d'abord, tous les sommets de G de degré entrant 2 sont directement transformés en une porte multiplicative. Ensuite, tous les sommets de G de degré entrant 1 sont transformés en une porte multiplicative dont la deuxième entrée est une porte entrée contenant une constante quelconque. Enfin, il reste le cas des sommets de G de degré entrant d'au moins 3. Afin d'assurer la bijection entre sommets de G et portes multiplicatives de $\mathcal{C}$, ces sommets sont transformés en un sous-circuit en forme de *peigne* constitué de portes additives et d'une unique porte multiplicative en puits, comme montré sur la figure 5.7.

L'algorithme f parcourt tous les sommets de G une seule fois. Il a donc une complexité en $\mathcal{O}(|V|)$. Ainsi, f est bien un algorithme polynomial.

Algorithme g. Maintenant, construisons l'algorithme g transformant une solution $S_{l_{\max}-\text{MB}}$ du problème de minimisation du nombre de réamorçages dans $\mathcal{C}$ en une solution $g(D, S_{l_{\max}-\text{MB}})$ du problème de $(k-1)$-PVCD dans D.

Soit S une solution du problème $l_{\max}$-MB dans le circuit $\mathcal{C} = f(G)$. Chaque réamorçage placé sur une porte additive est déplacé sur l'unique porte multiplicative du sous-circuit en peigne auquel elle appartient. Tous les réamorçages sont maintenant sur des portes multiplicatives, ce qui donne un ensemble S'. Rappelons que f est une bijection entre les sommets de G et les



(a) Sommet de D.

(b) Sous-circuit correspondant dans $\mathcal{C} = f(D)$.

Figure 5.7 – Exemple de transformation d'un sommet de G de degré entrant 4 en un sous-circuit en forme de peigne dans $\mathcal{C}$ par l'algorithme f

portes multiplicatives de $\mathcal{C}$: en utilisant son application réciproque, nous pouvons construire $\tilde{S} = f^{-1}(S')$. g est l'algorithme qui transforme S en $\tilde{S}$. Remarquons que comme f est une bijection, $|g(G,S)| = |S_{l_{\max}-\text{MB}}|$. Enfin, l'algorithme g est polynomial : il parcourt toutes les portes de S une seule fois, sa complexité est $\mathcal{O}(|\mathcal{G}|)$.

Montrons que si S est bien une solution de $l_{\max}$-MB pour $\mathcal{C} = f(G)$, alors $\tilde{S} = g(G,S)$ est bien une solution du problème k-PVCD dans G pour $k = l_{\max} - 1$. Rappelons que f induit une bijection entre les portes multiplicatives et les sommets de G.

Par contraposée, supposons que $\tilde{S}$ ne soit pas une solution du problème k-PVCD dans G et donc qu'il existe un chemin P dans G d'ordre k dont l'intersection avec $\tilde{S} = g(G,S)$ est vide. Soit $\mathcal{P} = f(P)$ le chemin dans $\mathcal{C}$ retourné par l'algorithme f. P étant d'ordre k, il a $k+1 = l_{\max}$ sommets. Le chemin $\mathcal{P}$ a donc un ordre multiplicatif de $l_{\max}$. Par construction de $\tilde{S}$, l'intersection de $\mathcal{P}$ et S est vide. S ne peut donc pas être une solution du problème $l_{\max}$-MB dans $\mathcal{C}$. Ainsi, $g(G,S)$ est une solution du problème de couverture par sommets des chemins d'ordre $k = l_{\max} - 1$ dans G.

Cette réduction préserve la NP-difficulté : D-$l_{\max}$-MB est donc NP-difficile. Il reste à prouver que D-$l_{\max}$-MB est effectivement dans NP. Pour cela il faut pouvoir vérifier en temps polynomial qu'une solution renvoyée par un oracle est bien une solution de D-$l_{\max}$-MB.

Soit $\mathcal{C} = (\mathcal{G}, \mathcal{W})$ un circuit arithmétique. Les chemins d'ordre multiplicatif $l_{\max}$ peuvent être calculés en utilisant un parcours en largeur tronqué sur chaque sommet. Cela a une complexité $O(\mathcal{G}|\mathcal{G} + \mathcal{W}|)$. Ainsi, toutes les solutions supposées du problème peut être vérifiées en temps polynomial, donc le problème $l_{\max}$-MB appartient à la classe NP.

Par conséquent, le problème de minimisation du nombre de réamorçages est NP-complet. □

5.4 Recherche pratique de solutions

Dans cette section, nous présentons une méthode pour déterminer le nombre minimal de réamorçages nécessaire pour évaluer un circuit. Cette méthode ne fournit pas forcément un nombre minimal de réamorçages si la taille du circuit est importante. Néanmoins, elle permet d'obtenir une bonne approximation de ce nombre. Pour cela, nous commençons par reformuler le problème de $l_{\max}$-MB en terme de programmation linéaire en nombres entiers.

Nous commençons par introduire les variables associées aux portes du circuit. Ensuite nous réécrivons les règles de croissance du bruit en contraintes linéaires. Finalement, nous appliquons cette modélisation aux circuits arithmétiques fournis par [ST].

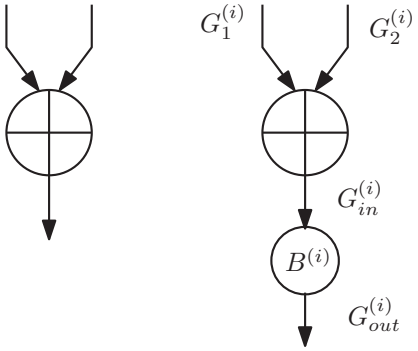


Figure 5.8 – Description des variables associées à une porte.

5.4.1 Définition des variables

Soit $\mathcal{C} = (\mathcal{G}, \mathcal{W})$ un circuit. Pour chaque porte $G \in \mathcal{G}$, on définit une variable booléenne qui prendra la valeur true si un réamorçage est nécessaire après cette porte et la valeur false sinon. Nous notons $B^{(i)}$ cette variable. L'objectif de notre programme linéaire est de minimiser le nombre de variables booléennes prenant la valeur true, donc la somme de ces variables.

Pour chaque porte $G^{(i)}$ de $\mathcal{C}$, nous notons $G_1^{(i)}$ et $G_2^{(i)}$ le niveau de bruit de chacune de ses deux entrées. La variable $B^{(i)}$ peut être vue comme une porte fictive de réamorçage, qui prend en entrée la sortie de $G^{(i)}$. Afin de simplifier les notations, $B^{(i)}$ désigne à la fois la variable booléenne associée à la porte $G^{(i)}$ et la porte-réamorçage qui est activée si $B^{(i)}$ vaut 1. Nous notons $G_{in}^{(i)}$ le niveau de bruit de l'entrée de $B^{(i)}$ (qui est donc égal au niveau de bruit de la sortie de $G^{(i)}$) et $G_{out}^{(i)}$ le niveau de bruit de la sortie de $B^{(i)}$. Notons que $G_{in}^{(i)} = G_{out}^{(i)}$ si $B^{(i)}$ vaut false. Une représentation de l'organisation de ces variables est donnée figure 5.8.

Rappelons que les variables modélisant le niveau de bruit ont comme borne inférieure 1 et comme borne supérieure $l_{\max}$ afin de pouvoir déchiffrer correctement les données à la sortie d'un calcul.

Minimiser le nombre de bootstrappings revient à minimiser le nombre de variables booléennes prenant la valeur true. La fonction objectif de notre programme linéaire est donc de minimiser la somme des variables booléennes

$$\sum_i B^{(i)}.$$

5.4.2 Définition des contraintes

Nous devons réécrire les relations entre les niveaux de bruit des différentes variables en contraintes linéaires. Dans cette section, nous décrivons ces contraintes pour les opérations homomorphes principales : addition, multiplication, multiplication par une constance, et bien sûr le réamorçage. Cependant le modèle peut facilement être adapté à d'autre types d'opérations homomorphes tant que l'évolution du niveau de bruit associé à ces opérations peut être exprimée en contraintes linéaires.

Réamorçage. Commençons par exprimer les contraintes associées à la gestion du niveau de bruit de l'opération de réamorçage que l'on a ajoutée à chaque porte du circuit. Rappelons que l'opération de réamorçage ne ramène pas le niveau de bruit à 1, mais plutôt à un niveau N qui est déterminé à la fois par le schéma considéré et par les paramètres choisis pour ce schéma. De plus, si l'opération de réamorçage n'est pas effectuée après la porte $G^{(i)}$, alors le niveau de bruit de la sortie de $G(i)$ n'est pas affecté et $G_{in}^{(i)} = G_{out}^{(i)}$. Ceci peut être modélisé par la contrainte quadratique suivante :

$$G_{out}^{(i)} = G_{in}^{(i)}.(1 - B^{(i)}) + N.B^{(i)}. \tag{5.1}$$

Pour reformuler cette contrainte quadratique sous la forme de contraintes linéaires, nous introduisons une constante auxiliaire X telle que $X \geq l_{\max}$. Le système d'inéquations linéaires suivant nous donne les propriétés voulues sur les niveaux de bruit :

$$\begin{cases} G_{out}^{(i)} \geq N \cdot B^{(i)} & (5.2) \\ G_{out}^{(i)} \leq N + (1 - B^{(i)}) \cdot X & (5.3) \\ G_{out}^{(i)} \geq G_{in}^{(i)} - X \cdot B^{(i)} & (5.4) \\ G_{out}^{(i)} \leq G_{in}^{(i)} + X \cdot B^{(i)}. & (5.5) \end{cases}$$

En effet, si la variable $B^{(i)}$ prend la valeur true, c'est-à-dire si un réamorçage est effectué à la sortie de la porte $G^{(i)}$, les équations 5.2 et 5.3 assurent l'égalité $G_{out}^{(i)} = N$. De plus, comme $X \geq l_{\max}$, les équations 5.4 et 5.5 restent vraies. D'autre part, si la variable $B^{(i)}$ prend la valeur false, donc si le réamorçage n'est pas effectué après la porte $G^{(i)}$, les équations 5.4 and 5.5 assurent $G_{out}^{(i)} = G_{in}^{(i)}$ et, de même que précédemment, les equations 5.2 et 5.3 restent vraies.

Addition. Soit c_1 et c_2 deux chiffrés avec un niveau de bruit respectif l_1 et l_2. Soit $c_3 = c_1 + c_2$ avec un niveau de bruit l_3. Nous voulons que $l_3 = \max(l_1, l_2)$. La fonction maximum n'est pas une fonction linéaire, donc nous ne pouvons pas l'utiliser directement dans la formulation de notre programme linéaire. Le maximum n'est rien d'autre que le minimum des majorants, si celui-ci existe. Or la définition d'un majorant, qui suit, peut s'exprimer uniquement avec des équations linéaires :

$$l_3 = \max(l_1, l_2) \iff \begin{cases} l_3 \geq l_1 \\ l_3 \geq l_2 \\ l_3 = l_1 \text{ ou } l_3 = l_2. \end{cases}$$

Nous cherchons à minimiser la valeur de l_3 : si le minimum des majorants de l'ensemble $\{l_1, l_2\}$ existe, c'est nécessairement le maximum de $\{l_1, l_2\}$. Or nous travaillons uniquement avec des niveaux de bruits entiers, donc, d'après le lemme de Zorn, ce minimum existe. En notant A une porte additive, nous obtenons le système d'équations linéaires suivant :

$$A_{in}^{(i)} = \max(A_1^{(i)}, A_2^{(i)}) \Longrightarrow \begin{cases} A_{in}^{(i)} \geq A_1^{(i)} \\ A_{in}^{(i)} \geq A_2^{(i)}, \end{cases}$$

dont les variables doivent respecter les bornes

$$1 \leq A_{in}^{(i)} \leq l_{\max} \text{ et } 1 \leq A_j^{(i)} \leq l_{\max}, \ \forall j.$$

Remarque 3. Si le nombre de portes additives dans le circuit est important, nous pouvons modifier les contraintes afin de prendre en compte le bruit logarithmique introduit par une addition. Soit $\varepsilon \in [0, 1]$ le bruit ajouté par une porte additive. Le niveau de bruit à la sortie de la porte est $l_3 = \max(l_1, l_2) + \varepsilon$. Dans ce cas, les contraintes deviennent

$$\begin{cases} A_{in}^{(i)} \geq A_1^{(i)} + \varepsilon \\ A_{in}^{(i)} \geq A_2^{(i)} + \varepsilon, \end{cases}$$

avec les même bornes supérieures et inférieures que pour l'addition.

Multiplication. Soit c_1 et c_2 deux chiffrés avec comme niveaux de bruits respectifs l_1 et l_2. Notons $c_3 = c_1 \cdot c_2$ avec un niveau de bruit l_3. Dans le modèle des schémas à croissance linéaire,

nous voulons $l_3 = \max(l_1, l_2) + 1$. De manière similaire à l'addition et en notant M la porte multiplicative, nous pouvons utiliser les contraintes suivantes :

$$M_{in}^{(i)} = \max(M_1^{(i)}, M_2^{(i)}) + 1 \Longrightarrow \begin{cases} M_{in}^{(i)} \geq M_1^{(i)} + 1 \\ M_{in}^{(i)} \geq M_2^{(i)} + 1, \end{cases}$$

dont les variables doivent respecter les bornes

$$1 \leq M_{in}^{(i)} \leq l_{\max} \text{ et } 1 \leq M_i^{(j)} \leq l_{\max}, \ \forall j.$$

Permutations, portes-NOT et addition par une constante. Dans la pratique, ces opérations ont peu d'influence sur le niveau de bruit d'un chiffré en comparaison avec la multiplication, donc nous supposons que le niveau de bruit en sortie d'une de ces opérations est le même que le niveau de bruit en entrée, ce qui nous donne la contrainte suivante :

$$G_{in}^{(i)} = G_1^{(i)}.$$

Nous montrons qu'une solution du programme linéaire ainsi défini est équivalent au problème $l_{\max}$-MB.

Théorème 5.7. *Soit $\mathcal{C}$ un circuit arithmétique. Le programme linéaire défini précédemment permet de résoudre le problème $l_{\max}$-MB dans $\mathcal{C}$.*

Démonstration. Les contraintes du programme linéaire ont été conçues pour qu'une solution au problème de minimisation du nombre de réamorçages soit une solution au programme linéaire.

Nous prouvons l'implication réciproque : toute solution du programme linéaire est une solution au problème de minimisation du nombre de réamorçages.

Soit S une solution du programme linéaire : S est l'ensemble des variables booléennes prenant la valeur true. Supposons que S n'est pas une solution du problème de minimisation du nombre de réamorçages. Alors il existe un chemin $\mathcal{P}$ dans $\mathcal{C}$ d'ordre multiplicatif $l_{\max}-1$ tel que $\mathcal{P} \cap S = \emptyset$. Le niveau de bruit d'un chiffré le long de $\mathcal{P}$ respecte les contraintes du programme linéaire et les bornes sur les variables. En particulier, le niveau de bruit d'un chiffré au début du chemin est d'au moins 1 et augmente de 1 à chaque porte multiplicative. Ainsi, son niveau de bruit à la fin du chemin est d'au moins $l_{\max}$. Ceci est en contradiction avec les bornes imposées sur les variables du programme linéaire, chaque variable étant majorée par $l_{\max} - 1$

Ainsi, S est une solution du problème de minimisation du nombre de réamorçages et toute solution du programme linéaire est une solution du problème $l_{\max}$-MB. □

5.5 Expérimentations

Dans cette section nous discutons des performances de notre modélisation. Nous testons la formulation sous forme de programmation linéaire du problème de minimisation du nombre de réamorçages sur les circuits fournis par Smart et Tillich [ST], ainsi que sur le circuit de l'AES utilisé dans [GHS12]. Les caractéristiques des différents circuits sont décrites dans le Tableau 5.1. Pour les tests, nous supposons que les chiffrés servant d'entrée au circuit possèdent un niveau de bruit de 1 et que le niveau de bruit des chiffrés en sortie du circuit est strictement inférieur à $l_{\max}$.

Nous avons utilisé le solveur *Gurobi Optimizer*[1] sur un MacBook pro avec un processeur Intel Core i7 à 2,3 GHz et de 16Go de RAM. Les résultats obtenus sont donnés dans le tableau 5.2. Nous avons concentré nos tests sur trois jeux de paramètres.

1. $(l_{\max} = 2, N = 1)$. Pour ces paramètres nous trouvons les mêmes solutions que dans [LP13].

1. http ://www.gurobi.com/

Circuits	portes multiplicatives	portes additives	Portes `NOT`	Profondeur multiplicative
Adder 32 bits	127	61	187	64
Adder 64 bits	265	115	379	128
Comparator 32 bits	150	0	150	23
Multiplier 32×32	5 926	1 069	5 379	128
AES (expanded key)	5 440	20 325	1 927	41
DES (expanded key)	18 175	1 351	10 875	262
MD5	29 084	14 150	34 627	2 973
SHA256	90 825	42 029	103 258	3 977

Table 5.1 – Caractéristiques des circuits arithmétiques utilisés pour les tests.

Circuits	l_{max}	N	Solution heuristique 1	Solution heuristique 2	Notre solution
Adder 32bits	2	1	127	127	127
Adder 32bits	4	1	31	32	31
Adder 32bits	20	9	5	5	4
Adder 64bits	2	1	265	267	265
Adder 64bits	4	1	63	74	63
Adder 64bits	20	9	10	12	10
Comparator	4	1	45	45	45
Comparator	20	9	1	1	1
Multiplier	2	1	6 350	5 926	5 924
Multiplier	4	1	3 200	3 110	1 219
Multiplier	20	9	105	116	69
AES	2	1	4 504	5 440	3 040
AES	20	9	736	1 600	220±20
DES	2	1	18 399	18 175	18 041
DES	20	9	4 435	4 006	440±20
MD5	2	1	29 084	34 496	28 896
SHA256	2	1	90 825	97 009	88 178

Table 5.2 – Nombre minimal de réamorçages pour les circuits décrits tableau 5.1.

2. $(l_{\max} = 4, N = 1)$. Pour ces paramètres nous trouvons les mêmes solutions que dans [LP13].
3. $(l_{\max} = 20, N = 9)$. Paramètres plus réalistes, similaires à ceux utilisés dans [HS15].

Pour les circuits simples, tels que `Adder` ou `Comparator`, les heuristiques trouvent une solution optimale ou une solution proche. Cependant, pour ces circuits, le calcul d'une solution optimale par notre méthode n'est pas plus lente que l'application des heuristiques.

Pour les circuits plus gros, le temps de calcul est difficile à prédire. Pour un $l_{\max}$ petit ou proche de la profondeur multiplicative du circuit, une solution optimale est trouvée en quelques minutes. Entre ces bornes, le solveur peut prendre plusieurs heures pour trouver une solution optimale. Cependant, le solveur trouve en quelques dizaines de minutes une bonne approximation qui est bien meilleure que les deux heuristiques. Il est possible de contrôler la finesse de ces approximations. Par exemple, sur le circuit DES, nous avons arrêté le calcul après 3h25min, quand le gap atteint 5% d'erreur. En comparaison avec la meilleure heuristique, cela permet d'éviter 3566 réamorçages : la lourdeur de ce précalcul est à relativiser en considérant le temps gagné lors de toutes les évaluations suivantes du circuit.

5.6 Conclusion

Dans ce chapitre, nous avons montré que le problème de minimisation des réamorçages est polynomial quand la profondeur multiplicative maximale de circuit admise est 2, et NP-complet quand cette dernière est plus grande que 3. Nous avons par ailleurs décrit une recherche de solutions, qui, même si elle ne trouve pas toujours le nombre minimal de réamorçages, donne dans tous les cas une approximation du minimum qui permet de diminuer grandement le temps de l'évaluation homomorphe. Par exemple, lors de l'évaluation d'un AES, nous passons de 736 réamorçages (en utilisant l'heuristique la plus performante) à 220, soit un gain de 516 réamorçages.

Nos résultats sur la complexité théorique du problème de minimisation des bootstrappings ont été récemment améliorés dans [BLMZ17]. Dans ce papier, les auteurs donnent ainsi un algorithme d'approximation d'une solution en temps polynomial.

Expérimentalement, nous avons constaté que les ensembles minimaux de réamorçages pour une profondeur multiplicative supérieure à 2 sont inclus dans ceux pour $l_{\max} = 2$. Cette conjecture, qui reste à prouver, permettrait certainement d'améliorer la recherche d'ensemble minimaux de réamorçages en diminuant le nombre d'entrées du programme linéaire.

Ces travaux montrent enfin que la conception des circuits joue un rôle important pour leur évaluation homomorphe. En effet, prendre en compte dès la conception des circuits les spécificités de l'évaluation homomorphe pour les circuits, et notamment leur profondeur multiplicative, améliorerait encore les performances de leur évaluation. Une première piste a été proposée par [CDS15], en utilisant un outil d'optimisation de circuits booléen, mais cet outil n'est pas dédié au chiffrement homomorphe.

Troisième partie

Conception de schémas *ad hoc*

Chapitre 6

Déduplication vérifiable de données chiffrées

Sommaire

L'usage de serveurs externes de stockage a explosé au cours de cette dernière décennie. Si ces services ont été vendus aux consommateurs avec des arguments d'économie de ressources, ces plateformes de stockages sont extrêmement coûteuses pour les fournisseurs. Afin de maîtriser l'expansion de ces plateformes, les fournisseurs ont recours à la déduplication : si plusieurs utilisateurs mettent le même fichier dans leur espace de stockage respectif, le serveur n'en garde qu'une seule copie et la renvoie indifféremment à l'utilisateur qui en fait la demande. Ce système n'est plus viable dès que les utilisateurs veulent chiffrer leurs données pour préserver leur vie privée ou la confidentialité de données d'entreprise. En effet, s'ils utilisent un chiffrement probabiliste, l'indistinguabilité des messages ne permet pas, par définition, la déduplication. De plus, même si le schéma retenu pour chiffrer les messages serait entièrement déterministe, les clés utilisées par deux clients sont choisies de manière indépendante, empêchant à nouveau la déduplication. C'est dans ce contexte qu'est née la notion de chiffrement verrouillé par le message (abrégé en MLE, de l'anglais *message-locked encryption*), permettant à la fois la détection de doublons (deux chiffrés sont liés au même clair) et la gestion des clés (n'importe quel utilisateur est capable de déchiffrer le fichier stocké sur le serveur). Notre contribution consiste en la définition d'une nouvelle notion de sécurité pour les schémas de type MLE ainsi que la conception d'un schéma atteignant cette notion de sécurité. Ces résultats ont été publiés à CANS 2016 [CLP16].

6.1 Chiffrement verrouillé par le message

6.1.1 Émergence d'un nouveau schéma

Les travaux sur la déduplication de données chiffrées ont été initiés par Douceur et al. en 2002 avec la proposition d'un schéma de chiffrement convergent [DAB+02]. Ce schéma est conçu pour que chaque utilisateur possédant le même message m obtienne la même clé k et le même chiffré c. Le protocole est le suivant. Soit H une fonction de hachage et SE un schéma de chiffrement symétrique déterministe. La clé utilisée pour chiffrer un message M est $k_M = H(M)$ et le chiffré est $c = \mathsf{SE.Encrypt}(M, k_M)$. Chaque chiffré c est accompagné d'un marqueur $T = H(C)$. Si deux marqueurs sont égaux, alors les deux chiffrés chiffrent le même message. De plus, la dérivation de clé étant une fonction déterministe du message, deux utilisateurs ayant le même message auront la même clé et pourront donc déchiffrer indifféremment des chiffrés du même message émis par n'importe quel utilisateur. Ce système de chiffrement convergent répond donc à la problématique de déduplication de données chiffrées.

Néanmoins, dans [BKR13], les auteurs soulignent le manque de modèle de sécurité formel du schéma de chiffrement convergent. Le schéma original a donné de nombreuses variantes, dont certaines sont sujettes à des attaques d'intégrité. Par exemple, la variante HCE1 utilise un haché de la clé comme marqueur à la place d'un haché du chiffré. Il est alors possible pour un attaquant qui veut remplacer le chiffré d'un message M_1 par un autre chiffré de M_2 de calculer la clé correspondant à M_1 et le marqueur approprié T_1, puis de chiffrer M_2 et d'y adjoindre le marqueur T_1. Le serveur va alors traiter les deux chiffrés comme chiffrant le même message, alors qu'il s'agit de messages différents.

L'article [BKR13] a donc apporté une certaine rigueur à ces nouveaux schémas. Il définit un modèle de sécurité pour les schémas existants, propose des preuves et/ou des attaques pour ces derniers et construit de nouveaux protocoles plus efficaces. Les auteurs proposent le nom de *message-locked encryption* (MLE) pour rassembler ces primitives. Ce nom insiste sur le fait que la clé utilisée pour chiffrer est une fonction du message, nous le traduirons pas *chiffrement verrouillé par le message*.

Abadi et al. [ABM+13] proposent le premier schéma totalement probabiliste. Le cas interactif a été étudié dans [BK15] et utilise du chiffrement totalement homomorphe.

Enfin, dans le cadre de la cryptographie à clé publique, des primitives peuvent sembler se rapprocher du MLE comme des schémas de chiffrement qui permettent le test d'égalité entre deux chiffrés [YTHW10]. D'autres propositions, comme le chiffrement cherchable [ABC+05, FP07] ou le chiffrement à test clair/chiffré [CFGL12] semblent aussi fournir des propriétés semblables à celles attendues pour un schéma de chiffrement verrouillé par le message. Néanmoins, ces schémas ne sont pas pertinents car dans tous ces modèles, un utilisateur A n'est pas en mesure de déchiffrer un fichier chiffré par un autre utilisateur B.

6.1.2 Définition formelle

Un schéma MLE est défini par les six algorithmes suivants : (PPGen, KD, $\mathsf{Encrypt}$, $\mathsf{Decrypt}$, EQ, Valid), s'exécutant sur un espace des messages $\mathcal{M}$, un espace des chiffrés $\mathcal{C}$ et un espace des clés $\mathcal{K}$. Le paramètre de sécurité est noté λ.

- L'algorithme de génération des paramètres PPGen prend en entrée 1^λ et retourne les paramètres publics $\mathsf{pp} \leftarrow \mathsf{PPGen}(1^\lambda)$.
- L'algorithme de dérivation des clés KD prend en entrée les paramètres publics pp et un message M. Il retourne une clé dérivée du message de manière déterministe $k_M \leftarrow \mathsf{KD}(\mathsf{pp}, M)$.
- L'algorithme de chiffrement $\mathsf{Encrypt}$ prend en entrée les paramètres pp, un message M et une clé dépendante de ce message k_M. Cet algorithme peut être déterministe ou probabiliste et retourne un chiffré $c \leftarrow \mathsf{Encrypt}(\mathsf{pp}, k_M, M)$.

— L'algorithme de déchiffrement $\mathsf{Decrypt}$ prend en entrée les paramètres publics pp, une clé k_M et un chiffré c. Il retourne un message M ou $\perp$: $\{M, \perp\} \leftarrow \mathsf{Decrypt}(\mathsf{pp}, k_M, c)$.
— L'algorithme de test d'égalité EQ prend en entrée les paramètres publics pp et deux chiffrés c_1 et c_2. Il retourne 1 si les deux chiffrés ont été générés à partir du même message, et 0 sinon : $\{0,1\} \leftarrow \mathsf{EQ}(\mathsf{pp}, c_1, c_2)$.
— L'algorithme de vérification de validité Valid prend en entrée les paramètres publics pp et un chiffré c. Il retourne 1 si c est un chiffré correctement formé, et 0 sinon : $\{0,1\} \leftarrow \mathsf{Valid}(\mathsf{pp}, c)$.

Un schéma MLE est dit *correct* si pour tout $\lambda \in \mathbb{N}$, pour tout $\mathsf{pp} \leftarrow \mathsf{PPGen}(1^\lambda)$, pour tout $M \in \mathcal{M}$ et pour tout $c \leftarrow \mathsf{Encrypt}(\mathsf{pp}, \mathsf{KD}(\mathsf{pp}, M), M)$, on a :
— $M = \mathsf{Decrypt}(\mathsf{pp}, \mathsf{KD}(\mathsf{pp}, M), c)$ et $\mathsf{Valid}(\mathsf{pp}, c) = 1$;
— $\mathsf{EQ}(\mathsf{pp}, \mathsf{Encrypt}(\mathsf{pp}, \mathsf{KD}(\mathsf{pp}, M), M; \omega), \mathsf{Encrypt}(\mathsf{pp}, \mathsf{KD}(\mathsf{pp}, M), M; \omega')) = 1$.

Cette dernière propriété est équivalente à la correction du tag de [BKR13] (mais nous avons explicitement écrit la partie aléatoire ω et ω' pour rappeler que le schéma peut être probabiliste).

Pour que ce type de schéma reste intéressant pour l'utilisateur, il est nécessaire que la clé k_M dérivée d'un message M soit significativement plus courte que M. Comme formalisé dans [BKR13], il doit exister des constantes $c, d < 1$ telle que la fonction prenant en entrée $\lambda \in \mathbb{N}$ et retournant $\max_{\mathsf{pp},M} (\mathtt{Pr}[|\mathsf{KD}(\mathsf{pp}, M)| > d|M|^c])$ soit négligeable ($\mathsf{pp} \in \mathsf{PPGen}(1^\lambda)$ et $M \in \mathcal{M}$).

6.1.3 Modèle de sécurité

Confidentialité

Du fait de la présence d'une procédure de test d'égalité entre deux chiffrés, un schéma de type MLE ne peut pas atteindre la notion classique d'indistinguabilité décrite dans le chapitre 3. En effet, dans le cadre par exemple du jeu IND-CPA, un adversaire polynomial a un avantage de 1 car il lui suffit d'exécuter la procédure de test d'égalité entre le chiffré proposé par le challengeur et le chiffré d'un des messages envoyé au challengeur.

Ainsi, la notion privilégiée pour qualifier la sécurité d'un schéma de type MLE est la confidentialité de messages non-prévisibles. L'imprévisibilité des messages est formalisée par l'entropie : il est supposé que les messages chiffrés avec ce type de schéma ont une haute entropie-min. Nous rappelons la définition de l'entropie-min.

Définition 6.1 (Entropie-min)**.** *L'entropie-min d'une variable aléatoire X est :*

$$H_\infty(X) = -\log(\max_x (\mathtt{Pr}[X = x])\,.$$

La notion d'entropie nous permet de définir la notion de source de blocs, qui caractérisent les messages pour lesquels la confidentialité est garantie avec un chiffrement MLE. Un adversaire qui a accès à un nombre infini de messages a un avantage 1 (à cause de la procédure de test d'égalité entre deux chiffrés). Ainsi, le nombre de messages est un paramètre de la sécurité du schéma [Bel98]. Nous commençons par définir une distribution de variables aléatoires à haute entropie-min.

Définition 6.2 ((T,μ)-source)**.** *Une variable aléatoire X telle que $H_\infty(X) \geq \mu$ est appelée une* μ-source*. Une* (T,μ)-source *est un T-uplet de variables aléatoires $\mathbf{X} = (X_1, \ldots, X_T)$ où chaque variable aléatoire X_i est une μ-source.*

Enfin, pour capturer la notion de distribution de messages à haute entropie imprévisibles, nous nous appuyons sur la définition suivante, où chaque message défini par la distribution ne dépend pas des messages qui le précèdent.

Définition 6.3 ((T,μ)-source de blocs)**.** *Une* (T,μ)-source de blocs *est un T-uplet de variables aléatoires $\mathbf{X} = (X_1, \ldots, X_T)$ où chaque variable aléatoire $X_i|_{X_1=x_1,\ldots,X_{i-1}=x_{i-1}}$ est une μ-source.*

Dans les travaux précédents sur le MLE, la confidentialité de messages imprévisibles a été déclinée en deux variantes : PRV$-CDA et PRV-CDA2, PRV venant de l'anglais *privacy* (confidentialité) et CDA signifiant attaque à distribution choisie (Chosen Distribution Attacks).

— La notion de PRV$-CDA [BKR13]. Dans cette expérience, l'adversaire doit distinguer le chiffré d'un vecteur de messages générés selon une distribution qu'il a choisi d'une suite de bits aléatoires.
— La notion de PRV-CDA2 [ABM+13] présente deux différences majeures avec celle de PRV$-CDA. La première est que l'adversaire a accès aux paramètres publics du schéma : la confidentialité doit être assurée aussi pour les distributions de messages qui pourraient dépendre de ces paramètres publics. De plus, l'expérience de sécurité est également modifée. Dans ce jeu, l'adversaire interagit avec un oracle réel ou fictif qui renvoit le chiffré d'un vecteur de messages générés soit suivant la distribution choisie par l'adversaire (mode réel), soit aléatoirement (mode fictif). L'adversaire doit finalement distinguer comment les messages du vecteur chiffré ont été générés.

Notre schéma atteint une propriété de confidentialité qui est une combinaison des deux expériences existantes. En effet, comme nous intégrons des NIZK à notre schéma pour garantir la vérifiabilité, une expérience de type PRV$-CDA [BKR13] n'est pas adaptée. Les chiffrés valides contenant une NIZK valide, ils sont aisément distinguables d'une suite de bits aléatoires. Notre expérience de confidentialité calque donc l'expérience PRV-CDA2 [ABM+13]. Néanmoins, nous utilisons le *Leftover Hash Lemma* afin de prouver que notre procédure de dérivation de clés KD retourne des clés avec une distribution proche de l'uniforme. Or, ce lemme ne vaut que pour des entrées indépendantes des paramètres du système. Ici, les entrées sont des messages : comme dans [BKR13], la confidentialité des messages n'est pas préservée lorsque la distribution des messages choisie par l'adversaire dépend des paramètres publics. Ainsi, nous appelons notre propriété de confidentialité PRV-piCDA, de l'anglais *Privacy under parameter independent Chosen Distribution Attack* pour souligner que celle-ci n'est atteinte que dans le cas de messages choisis indépendamment des paramètres publics. Comme dans la propriété PRV-CDA2 [ABM+13], nous nous appuyons sur un oracle de chiffrement réel ou fictif.

Définition 6.4 (Oracle de chiffrement réel ou fictif)**.** *L'oracle de chiffrement réel ou fictif prend en entrée le couple* $(\mathsf{mode}, \mathbb{M})$ *avec* $\mathsf{mode} \in \{\mathsf{réel}, \mathsf{fictif}\}$*, et* $\mathbb{M}$ *un circuit de taille polynomiale représentant une distribution sur des vecteurs de* T *messages de l'espace des messages* $\mathcal{M}$*. Si le mode est le mode réel (*$\mathsf{mode} = \mathsf{réel}$*) alors l'oracle tire le vecteur suivant la distribution choisie par l'adversaire* $(M_1, \ldots, M_T) \leftarrow \mathbb{M}$*. Si le mode est le mode fictif (*$\mathsf{mode} = \mathsf{fictif}$*) alors l'oracle tire* T *messages uniformes et indépendants dans l'espace* $\mathcal{M}$*. Enfin, l'oracle retourne un vecteur de chiffrés* $\mathbf{C} = (c_1, \ldots, c_T)$ *tel que pour tout* i*,* $k_{M_i} = \mathsf{KD}(\mathsf{pp}, M_i)$ *et* $c_i = \mathsf{Encrypt}(\mathsf{pp}, k_{M_i}, M_i)$*.*

Comme [RSV13], nous restreignons les requêtes des adversaires aux requêtes provenant de circuits de taille polynomiale dans le modèle de l'oracle aléatoire.

Définition 6.5 (Adversaire échantillonnant polynomialement)**.** *Soit une* (T, μ)*-source de blocs. Soit* $\mathcal{A}$ *un adversaire probabiliste en temps polynomial qui a accès à* λ *et à l'oracle de chiffrement réel ou fictif* $\mathsf{RoR}(\mathsf{mode}, \mathsf{pp}, \cdot)$*.* $\mathcal{A}$ *est un adversaire échantillonnant polynomialement* q *requêtes, si pour chaque appel* $\mathbf{M}$ *de* $\mathcal{A}$ *à l'oracle aléatoire,* $\mathbf{M}$ *est une* (T, μ)*-source de blocs échantillonnable par un circuit de taille polynomial qui utilise au plus* q *requêtes à l'oracle aléatoire.*

De manière informelle, la notion de confidentialité PRV-piCDA requiert que les chiffrés de messages aléatoires soient indistinguables de chiffrés de messages provenant de (T, μ)-source de blocs.

Définition 6.6 (Sécurité PRV-piCDA)**.** *Un schéma MLE* Λ *est* PRV-piCDA *sûr pour des messages issus de* (T, μ)*-source de blocs si pour tout adversaire probabiliste en temps polynomial* $\mathcal{A}$ *générant des* (T, μ)*-source de blocs, il existe une fonction négligeable* $\nu(\lambda)$ *telle que :*

$$\mathsf{Adv}^{\mathsf{PRV-piCDA}}_{\Lambda,\mathcal{A}}(\lambda) = \left| Pr\left[\mathbf{Exp}^{\mathsf{réel}}_{\Lambda,\mathcal{A}} = 1\right] - Pr\left[\mathbf{Exp}^{\mathsf{fictif}}_{\Lambda,\mathcal{A}} = 1\right] \right| \leq \nu(\lambda),$$

Expérience $\mathbf{Exp}^{\mathsf{mode}}_{\Lambda,\mathcal{A}}(\lambda)$
$\mathbb{M} \leftarrow \mathcal{A}_0(1^\lambda)$;
$\mathsf{pp} \leftarrow \mathsf{PPGen}(1^\lambda)$;
$\mathsf{mode}_\mathcal{A} \leftarrow \mathcal{A}_1^{\mathsf{RoR}(\mathsf{mode},\mathbb{M})}(1^\lambda, \mathsf{pp})$
Retourner $(\mathsf{mode}_\mathcal{A} == \mathsf{mode})$.

Figure 6.1 – Jeu PRV-piCDA : $\mathbf{Exp}^{\mathsf{mode}}_{\Lambda,\mathcal{A}}(\lambda)$

Expérience $\mathbf{Exp}^{TC}_{\Lambda,\mathcal{A}}(\lambda)$
$\mathsf{pp} \leftarrow \mathsf{PPGen}(1^\lambda)$;
$(M_0, c_1) \leftarrow \mathcal{A}(1^\lambda, \mathsf{pp})$;
Si $\mathsf{Valid}(\mathsf{pp}, c_1) = 0$ ou $M_0 = \perp$ retourner 0 ;
$k_M \leftarrow \mathsf{KD}(\mathsf{pp}, M_0)$;
$c_0 \leftarrow \mathsf{Encrypt}(\mathsf{pp}, k_M, M_0)$ et $M_1 \leftarrow \mathsf{Decrypt}(\mathsf{pp}, k_M, c_1)$;
Si $(\mathsf{EQ}(\mathsf{pp}, c_0, c_1) = 1) \wedge (M_0 \neq M_1) \wedge (M_1 \neq \perp)$ retourner 1 ;
Sinon, retourner 0.

Figure 6.2 – Jeu de la cohérence des marqueurs : $\mathbf{Exp}^{TC}_{\Lambda,\mathcal{A}}(\lambda)$

où le jeu $\mathbf{Exp}^{\mathsf{mode}}_{\Lambda,\mathcal{A}}(\lambda)$ *est défini Figure 6.1.*

Comme indiqué dans le théorème [ABM+13, Theorem 4.6], le cas où $\mathcal{A}$ fait de multiples requêtes à l'oracle de chiffrement est équivalent au cas où $\mathcal{A}$ ne lui fait qu'une unique requête.

Cohérence des marqueurs

La cohérence des marqueurs est une notion de sécurité évidente pour les sytèmes de MLE. En effet, elle assure que si deux chiffrés sont détectés comme identiques par le serveur, alors ils chiffrent effectivement le même message. Ainsi, le serveur a l'assurance que les chiffrés qu'il supprime sont effectivement des doublons, et que chaque utilisateur retrouvera bien les données qu'il a confiées au serveur. Pour formaliser cette notion, nous utilisons le jeu décrit par Abadi et al. dans [ABM+13].

Définition 6.7 (Cohérence des marqueurs (TC, tag consistency))**.** *Un schéma MLE* Λ *a des* marqueurs cohérents *si pour tout adversaire probabiliste s'exécutant en temps polynomial* $\mathcal{A}$*, il existe une fonction négligeable* $\nu(\lambda)$ *telle que :*

$$\mathsf{Adv}^{TC}_{\Lambda,\mathcal{A}}(\lambda) = Pr\left[\mathbf{Exp}^{TC}_{\Lambda,\mathcal{A}} = 1\right] \leq \nu(\lambda),$$

où l'expérience $\mathbf{Exp}^{TC}_{\Lambda,\mathcal{A}}(\lambda)$ *est décrite Figure 6.2.*

6.2 Vérifiabilité du MLE

Dans les travaux précédents, l'analyse de sécurité des schémas de chiffrement verrouillés par le message se concentre uniquement sur les aspects de confidentialité et de cohérence des marqueurs. Cette dernière propriété revient à prouver l'implication suivante : si $\mathsf{EQ}(c_1, c_2)$ retourne 1, alors c_1 et c_2 chiffrent le même message. Nous nous attachons ici à prouver également l'implication réciproque : si deux chiffrés c_1 et c_2 chiffrent le même message, alors le test d'égalité $\mathsf{EQ}(c_1, c_2)$ devrait retourner 1. Nous appelons cette propriété la *cohérence de la déduplication*. Cette notion est importante car elle permet de montrer que le schéma de chiffrement verrouillé par le message fournit *effectivement* la déduplication des données.

Expérience $\mathbf{Exp}^{DC}_{\Lambda,\mathcal{A}}(\lambda)$

$\mathsf{pp} \leftarrow \mathsf{PPGen}(1^\lambda)$;
$(M, c_0, c_1) \leftarrow \mathcal{A}(1^\lambda, \mathsf{pp})$;
Si $(\mathsf{Valid}(\mathsf{pp}, c_0) = 0) \vee (\mathsf{Valid}(\mathsf{pp}, c_1) = 0)$ alors retourner 0 ;
Si $\mathsf{EQ}(\mathsf{pp}, c_0, c_1) = 1$ alors retourner 0 ;
$k_M \leftarrow \mathsf{KD}(\mathsf{pp}, M)$;
$M_0 \leftarrow \mathsf{Decrypt}(\mathsf{pp}, k_M, c_0)$; $M_1 \leftarrow \mathsf{Decrypt}(\mathsf{pp}, k_M, c_1)$;
Si $M \neq M_0 \vee M \neq M_1$ alors retourner 0 ;
Retourner 1 ;

Figure 6.3 – Jeu de la cohérence de la déduplication : $\mathbf{Exp}^{DC}_{\Lambda,\mathcal{A}}(\lambda)$

6.2.1 Motivations

La nécessité de définir une nouvelle notion de sécurité pour les schémas de type MLE est apparue suite à un constat très simple : dans les principaux schémas de chiffrement convergent existants, l'utilisateur peut très facilement tricher et éviter le processus de déduplication. Le but de la propriété de cohérence de la déduplication est donc d'ajouter à ce type de schéma un caractère vérifiable. En effet, un serveur faisant usage d'un schéma de chiffrement de type convergent (et donc de son extension, le MLE) s'attend à ce que le processus de déduplication soit réellement effectué. Or, dans la plupart des schéma existant, le seul garant de la déduplication est l'utilisateur. Dans ces conditions, il est aisé de "faire dévier le schéma de sa fonctionnalité prescrite" [1]

En plus d'assurer un gain d'espace de stockage, la déduplication vérifiable a aussi un intérêt propre. Par exemple, aujourd'hui, un sujet épineux est le droit à l'oubli, et ce spécialement dans le cas du numérique. Comment un serveur peut prouver que certains fichiers chiffrant un contenu sensible ont bien été supprimés ? Le droit de la protection de la vie privée d'un utilisateur doit il prévaloir sur le droit à l'oubli d'un autre ? Si la décision de supprimer un fichier, comme par exemple un article diffamatoire ou une vidéo calomnieuse est ordonnée par un tribunal, alors le serveur devrait pouvoir prouver que la décision a été correctement appliquée. Avec un schéma de chiffrement verrouillé par le message vérifiable, il suffit de chiffrer le fichier litigieux avant de vérifier son existence dans la base de données. Si le test d'égalité retourne 1, supprimer l'exemplaire existant sur le serveur suffit pour prouver que plus aucun utilisateur n'aura accès à ce fichier.

6.2.2 Définition formelle

Nous donnons maintenant une définition formelle de notre nouvelle expérience $\mathbf{Exp}^{DC}_{\Lambda,\mathcal{A}}(1^\lambda)$.

Définition 6.8 (Cohérence de la déduplication (DC))**.** *Un schéma de chiffrement verrouillé par le message Λ a la propriété de cohérence de la déduplication si pour tout adversaire probabiliste s'exécutant en temps polynomial $\mathcal{A}$, il existe une fonction négligeable $\nu(\lambda)$ telle que :*

$$\mathsf{Adv}^{DC}_{\Lambda,\mathcal{A}}(\lambda) = Pr\left[\mathbf{Exp}^{DC}_{\Lambda,\mathcal{A}} = 1\right] \leq \nu(\lambda),$$

où l'expérience aléatoire $\mathbf{Exp}^{DC}_{\Lambda,\mathcal{A}}(1^\lambda)$ est décrite Figure 6.3.

6.2.3 La cohérence de la déduplication dans l'état de l'art

La cohérence de la déduplication n'a jamais été évoquée dans les travaux précédents, et encore moins formalisée. Les différentes solutions données par Bellare et al. [BKR13], que sont

1. Oded Goldreich, *The Foundations of Cryptography*, Préface, traduction personnelle.

CE, HCE1, HCE2, RCE, XHC and SXH n'atteignent pas cette propriété. Dans CE, le marqueur est $\tau = H(C)$, où $C = \mathcal{SE}(K, M)$ avec $K = H(M)$. Dans tous les autres schémas, le marqueur τ est l'image du message par une fonction $\mathcal{F}(\mathcal{F}(M))$, où $\mathcal{F}$ peut être soit une fonction de hachage (comme dans HCE1, HCE2, RCE), un extracteur (comme dans XHC) ou une projection (c'est le cas dans SXH). Quoi qu'il en soit, dans tous ces schémas, l'utilisateur peut utiliser une clé K aléatoire, qu'il conserve (au même titre qu'il aurait conservé la clé d'un schéma MLE) afin d'être capable de déchiffrer. La comparaison des marqueurs va alors toujours retourner 0, même si les messages chiffrés sont identiques.

Abadi et al. [ABM+13] ont proposé deux schémas : un schéma entièrement probabiliste et un schéma déterministe. Dans ce dernier, l'utilisateur peut tricher lors de la génération de clé comme précédemment. Le schéma n'atteint donc pas la propriété de cohérence de la déduplication. Leur schéma entièrement probabiliste, en revanche, introduit un lien entre les valeurs prises par la clé, les marqueurs et le chiffré en fonction du message sous la forme d'une preuve à divulgation nulle de connaissance. Cette NIZK force tous ces éléments à être cohérents les uns avec les autres : il semblerait donc que ce schéma atteigne la propriété de cohérence de la déduplication. On peut néanmoins souligner que l'utilisation d'une NIZK générique obère fortement l'efficacité du schéma.

Ainsi, nous avons introduit et formalisé la cohérence de la déduplication pour ne pas qu'un utilisateur puisse éviter le protocole de déduplication. Pour cela, le serveur doit pouvoir vérifier que chaque partie du chiffré, y compris les marqueurs et la clé utilisée pour chiffrer, doit être cohérente avec les autres et le message. Ainsi, le chiffré est bien dérivé *uniquement* du message, et non d'une source d'aléa externe, permettant la déduplication inter-utilisateurs. Un problème ouvert reste de prouver le lien formel entre l'existence dans le chiffré d'une telle preuve de cohérence et la cohérence de la déduplication.

6.2.4 Une construction générique

Dans cette section, nous donnons une contruction permettant de transformer n'importe quel schéma de chiffrement verrouillé par le message Λ confidentiel avec des marqueurs cohérents en un schéma Λ' atteignant de surcroît la cohérence de la déduplication. Pour cela, nous utilisons un schéma de mise en gage Γ ainsi qu'une preuve à divulgation nulle de connaissance Π. Notre schéma se compose des algorithmes suivants.

— PPGen. L'algorithme de génération des paramètres exécute Λ.PPGen qui renvoie Λ.pp, Γ.Setup qui renvoie Γ.pp et enfin la génération d'une chaîne commune de référence R pour la NIZK. Il retourne le triplet $\Lambda'.\mathsf{pp} = (\Lambda.\mathsf{pp}, \Gamma.\mathsf{pp}, R)$.

— KD. La dérivation des clés prend pour entrée $\Lambda'.\mathsf{pp}$ et un message M. Il exécute Λ.KD, et retourne la clé correspondante à M $k_M = \Lambda.\mathsf{KD}(\Lambda.\mathsf{pp}, M)$.

— Encrypt. Le chiffrement se fait en trois étapes. Premièrement, l'algorithme exécute la procédure $\Lambda.\mathsf{Encrypt}(\Lambda.\mathsf{pp}, k_M, M)$ qui renvoie c. Ensuite, il calcule une mise en gage de M, qui donne le couple $(C, r) = \Gamma.\mathsf{Commit}(\Gamma.\mathsf{pp}, M)$. Enfin, il fournit la preuve à divulgation nulle de connaissance suivante :

$$\pi = \mathsf{NIZK}\Big(M, r : c = \Lambda.\mathsf{Encrypt}(\Lambda.\mathsf{pp}, \Lambda.\mathsf{KD}(\Lambda.\mathsf{pp}, M), M) \wedge (C, r) = \Gamma.\mathsf{Commit}(\Gamma.\mathsf{pp}, M)\Big).$$

Le chiffré est le triplet $c' = (c, C, \pi)$.

— Decrypt. L'algorithme de déchiffrement prend en entrée un chiffré $c' = (c, C, \pi)$ et la clé k_M, et il exécute $\Lambda.\mathsf{Decrypt}(\Lambda.\mathsf{pp}, k_M, c)$ pour obtenir M.

— EQ. Le test d'égalité prend en entrée deux chiffrés $c'_1 = (c_1, C_1, \pi_1)$ et $c'_2 = (c_2, C_2, \pi_2)$, et exécute $\Lambda.\mathsf{EQ}(\Lambda.\mathsf{pp}, c_1, c_2)$.

— Valid. L'algorithme de vérification prend en entrée un chiffré $c' = (c, C, \pi)$. Il exécute $\Lambda.\mathsf{Valid}(\Lambda.\mathsf{pp}, c)$ puis vérifie la correction de la preuve NIZK π.

Sécurité. Le schéma Λ' atteint les mêmes propriétés de confidentialité et de cohérence des marqueurs que le schéma Λ. En effet, la perte dans les preuves de sécurité est négligeable et est due à deux facteurs. D'une part, une perte due à l'avantage de l'adversaire contre la confidentialité du schéma de mise en gage, si cette dernière est calculatoire. D'autre part, la perte liée à la propriété de non-divulgation des NIZK s'il est nécessaire de les simuler lors de la preuve de sécurité.

Le théorème 6.9 prouve la cohérence de la déduplication dans le schéma Λ'.

Théorème 6.9. *Si Λ est un schéma de chiffrement verrouillé par le message avec des marqueurs cohérents, Γ un schéma de mise en gage contraignant et Π un système de preuve NIZK valide, alors le schéma Λ' décrit ci-dessus atteint la cohérence de la déduplication.*

Démonstration. Nous allons procéder par réduction en réduisant la propriété de contrainte du schéma de mise en gage à la cohérence de la déduplication. Soit $\mathcal{A}$ un adversaire contre l'expérience de contrainte de Γ (voir section 3.3.4). $\mathcal{A}$ a accès aux paramètres publics $\Gamma.\mathsf{pp}$ du schéma Γ. Son but est de produire une mise en gage C ainsi que deux paires (M, r) et (M', r') telles que $\mathsf{Open}(C, M, r) = \mathsf{Open}(C, M', r') = 1$. $\mathcal{A}$ a aussi accès à un adversaire $\mathcal{B}$ qui a un avantage non-négligeable contre la cohérence de la déduplication de notre schéma.

Génération des paramètres. $\mathcal{A}$ génère d'abord les paramètres de Λ' et les envoie à $\mathcal{B}$.

Réponse de $\mathcal{B}$. $\mathcal{B}$ renvoie à $\mathcal{A}$ un triplet (M, c'_0, c'_1), avec $c'_0 = (c_0, C_0, \pi_0)$ et $c'_1 = (c_1, C_1, \pi_1)$, tel que les conditions de l'expérience de la cohérence de la déduplication soient vérifiées. En particulier, nous avons $\Lambda'.\mathsf{Valid}(\mathsf{pp}, c'_0) = 1$, $\Lambda'.\mathsf{Valid}(\mathsf{pp}, c'_1) = 1$ et $\Lambda.\mathsf{EQ}(\mathsf{pp}, c_0, c_1) = 0$.

Réponse au challenge de la mise en gage. En utilisant la propriété de validité de la NIZK, $\mathcal{A}$ peut extraire de π_0 et de π_1 les messages M'_0 et M'_1 correspondant respectivement à c_0 et à c_1. De plus, comme $\Lambda.\mathsf{EQ}(\mathsf{pp}, c_0, c_1) = 0$, et comme Λ a des marqueurs cohérents, nous avons $M'_0 \neq M'_1$.

De plus $\mathcal{B}$ réussissant l'expérience DC, $\mathcal{A}$ peut utiliser M, c_0 et c_1 pour calculer respectivement $k_M = \Lambda.\mathsf{KD}(\mathsf{pp}, M)$, $M_0 = \Lambda.\mathsf{Decrypt}(\mathsf{pp}, k_M, c_0)$ et $M_1 = \Lambda.\mathsf{Decrypt}(\mathsf{pp}, k_M, c_1)$ tels que $M = M_0 = M_1$.

Ainsi, pour au moins un $i \in \{0, 1\}$, on a $M_i \neq M'_i$. $\mathcal{A}$ peut extraire de la preuve NIZK r'_i et envoyer $C_i, (M_i, r_i), (M'_i, r'_i)$ au challengeur sur la contrainte de la mise en gage. Si (M, c'_0, c'_1) est un triplet valide contre la DC, $C_i, (M_i, r_i), (M'_i, r'_i)$ est un triplet valide contre la contrainte de la mise en gage : $\mathcal{A}$ a donc un avantage contre cette dernière propriété au moins égal à celui de $\mathcal{B}$ contre la DC. □

6.3 Une construction vérifiable basée sur ElGamal

6.3.1 Idée générale

Notre construction diffère des précédentes principalement sur la procédure de dérivation de clés. Contrairement à [ABM+13], nous utilisons une fonction de hachage avec des propriétés algébriques. Ceci nous évite d'utiliser des NIZK génériques [Gro10], gagnant ainsi en efficacité.

Nous commençons d'abord par diviser le message M en ℓ petits blocs $(m_1 \| \dots \| m_\ell)$ de ρ bits. La clé est le produit $k_M = \prod_{i=1}^{\ell} a_i^{m_i} \mod p$ avec des a_i publics. En utilisant le Théorème 6.15, nous prouvons que si les messages sont générés par une source avec une entropie min suffisante, la clé k_M est indistinguable d'une clé uniforme.

Chaque bloc m_i du message est ensuite chiffré avec la clé k_M en utilisant une variante du chiffrement ElGamal [Gam84] où les messages sont en exposant :

$$T_{1,i} = g_i^{r_i} \text{ et } T_{2,i} = h^{m_i} \cdot g_i^{r_i \cdot k_M}. \tag{6.1}$$

Notons que lorsqu'on déchiffre cette version d'ElGamal, nous obtenons g^{m_i}. Les blocs m_i doivent être choisis suffisamment petits pour être déchiffrés relativement aux attaques existantes sur le logarithme discret (voir section 3.2.1).

Pour générer un marqueur, nous utilisons la même technique que [ABM+13]. Nous créons donc une paire $(\tau_1 = t_1^u, \tau_2 = t_2^{u \cdot k_M})$ qui permet de détecter les duplicatas d'un fichier en utilisant un calcul de couplages. Pour vérifier que le marqueur a été calculé de façon correcte par l'utilisateur, ce dernier doit prouver la validité des équations suivantes :

$$\tau_1 = t_1^u \text{ et} \tag{6.2}$$

$$e(\tau_1, t_2)^{k_M} = e(t_1, \tau_2). \tag{6.3}$$

Pour atteindre la propriété de la cohérence de la déduplication, l'utilisateur fournit finalement une NIZK prouvant que le chiffré a été généré correctement. Il doit donc prouver que la clé a bien été dérivée du message, que les marqueurs utilisent la bonne clé et que les chiffrés ElGamal chiffrent le bon message avec la bonne clé. À cet effet, l'utilisateur ajoute au chiffré une mise en gage C sur les m_i :

$$C = \prod_{i=1}^{\ell} a_i^{m_i} \cdot x^s = k_M \cdot x^s \mod p. \tag{6.4}$$

Un élément crucial de notre NIZK est de prouver que l'élément k_M (vu comme un exposant dans les groupes $\mathbb{G}_1$ et $\mathbb{G}_T$) impliqué dans le marqueur, la mise en gage et les chiffrés est le même que le k_M (vu comme un élément du groupe $\mathbb{Z}_p^*$ d'ordre p) dérivé du message. Dans le marqueur et les chiffrés ElGamal (équations (6.1) à (6.3)), la clé k_M est un exposant et donc nous pouvons utiliser des preuves à divulgation nulle de connaissance standards et efficaces de type Schnorr [Sch89], et les rendre non interactives avec l'heuristique de Fiat Shamir [FS86].

Comme $C = k_M \cdot x^s = \prod_{i=1}^{\ell} a_i^{m_i} \cdot x^s \mod p$, nous pouvons aussi prouver de manière efficace la correction de la mise en gage. Il faut enfin prouver le lien entre la clé et le message. Pour cela, nous utilisons les équations (6.3) et (6.4). En effet, l'équation (6.3) peut être réécrite comme suit :

$$e(\tau_1, t_2)^C = e(t_1, \tau_2)^{x^s}. \tag{6.5}$$

Prouver la validité de l'équation (6.5) implique de prouver la validité d'un double logarithme discret. Nous utilisons les techniques de [Sta96, NS00] décrites dans le chapitre 3, ce qui affecte les performances de notre construction.

6.3.2 Construction

Nous décrivons maintenant notre schéma de MLE vérifiable Λ.

— PPGen. Soit λ le paramètre de sécurité et ℓ, ρ deux entiers. La génération des paramètres commence par la génération d'un environnement bilinéaire $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, e)$ avec p un entier premier de λ bits, $\mathbb{G}_1$, $\mathbb{G}_2$ et $\mathbb{G}_T$ trois groupes multiplicatifs de même ordre p et $e : \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_T$ un couplage bilinéaire asymétrique. Soit $t_1, \{g_i\}_{i=1,\ldots,\ell}, h$ des générateurs de $\mathbb{G}_1$ et t_2 un générateur de $\mathbb{G}_2$. Enfin, nous avons besoin de $\ell+1$ générateurs publics $x, a_1, \ldots, a_\ell$ de $\mathbb{Z}_p^*$.

$$\mathsf{pp} = \{p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, t_1, \{g_i\}_{i=1,\ldots,\ell}, h, t_2, x, \{a_i\}_{i=1,\ldots,\ell}\}.$$

— KD. La dérivation des clés prend en entrée les paramètres publics pp et un message M. On commence par diviser M en ℓ blocs $M = (m_1 \| \ldots \| m_\ell)$ de ρ bits.
La clé k_M est calculée comme le produit scalaire des m_i et des a_i.

$$k_M = \prod_{i=1}^{\ell} a_i^{m_i} \mod p.$$

— Encrypt. L'algorithme de chiffrement prend en entrée les paramètres publics pp, un message $M = (m_1 \| \dots \| m_\ell)$ et une clé k_M. Le chiffré est construit comme suit :

1. choisir $u \in \mathbb{Z}_p^*$ uniformément, et calculer $\tau_1 = t_1^u$ and $\tau_2 = t_2^{u \cdot k_M}$, and set $\tau = (\tau_1, \tau_2)$;
2. choisir $s \in \mathbb{Z}_p^*$ uniformément. En utilisant le schéma de Pedersen, mettre en gage les éléments m_i : $C = k_M \cdot x^s = \prod_{i=1}^{\ell} a_i^{m_i} \cdot x^s \mod p$;
3. pour tout $1 \leq i \leq \ell$, choisir des $r_i \in \mathbb{Z}_p^*$ uniformes et indépendants, puis calculer un chiffré ElGamal $T_{1,i} = g_i^{r_i}$ et $T_{2,i} = h^{m_i} \cdot g_i^{r_i \cdot k_M}$;
4. calculer la preuve non interactive à divulgation nulle de connaissance suivante.

$$\begin{aligned}
\pi \;=\; & \mathsf{NIZK}\Big(u, \{r_i\}_{i=1,\dots,\ell}, M, k_M, s : \\
& \tau_1 = t_1^u \wedge e(\tau_1, t_2)^{k_M} = e(t_1, \tau_2) \\
\wedge\;\; & T_{1,1} = g_1^r \wedge \dots \wedge T_{1,\ell} = g_\ell^r \\
\wedge\;\; & T_{2,1} = T_{1,1}^{k_M} g^{m_1} \wedge \dots \wedge T_{2,\ell} = T_{1,l}^{k_M} g^{m_\ell} \\
\wedge\;\; & C = \prod_{i=1}^{\ell} a_i^{m_i} \cdot x^s \\
\wedge\;\; & e(\tau_1, t_2)^C = e(t_1, \tau_2)^{x^s}\Big).
\end{aligned}$$

Le chiffré est composé de tous les éléments précédents :

$$c = (\tau, \{T_{1,i}, T_{2,i}\}_{i=1,\dots,\ell}, C, \pi).$$

— Valid. L'algorithme de vérification de validité prend en entrée un chiffré c de la forme $(\tau, \{T_{1,i}, T_{2,i}\}_i, C, \pi)$, et retourne 1 si la preuve π est correcte.

— Decrypt. L'algorithme de déchiffrement prend en entrée les paramètres pp, une clé k_M et un chiffré $c = (\tau, \{T_{1,i}, T_{2,i}\}_i, C, \pi)$, et s'exécute comme suit.

1. Pour tout $i \in \{1, \dots, \ell\}$, retrouver $h^{m_i} = T_{2,i} / T_{1,i}^{k_M}$ en exécutant le déchiffrement standard du schéma ElGamal ;
2. Pour tout $i \in \{1, \dots, \ell\}$, rechercher m_i avec une recherche de logarithme discret. Cette étape est rendue possible par le choix de blocs de ρ bits.
3. Retourner $M = (m_1 \| \dots \| m_\ell)$.

— EQ. L'algorithme de test d'égalité prend en entrée les paramètres publics pp et deux chiffrés valides $c = (\tau, \{T_{1,i}, T_{2,i}\}_i, C, \pi)$ et $\tilde{c} = (\tilde{\tau}, \{\tilde{T}_{1,i}, \tilde{T}_{2,i}\}, \tilde{C}, \tilde{\pi})$. On décompose τ en $\tau_1 = t_1^u$ et $\tau_2 = t_2^{u \cdot k}$ et $\tilde{\tau}$ en $\tilde{\tau}_1 = t_1^{\tilde{u}}$ et $\tilde{\tau}_2 = t_2^{\tilde{u} \cdot k}$. Cet algorithme retourne 1 si et seulement si $e(\tau_1, \tilde{\tau}_2) = e(\tilde{\tau}_1, \tau_2)$.

Remarque 4. L'algorithme Valid doit être exécuté avant les procédures de déchiffrement Decrypt et de test d'égalité EQ afin d'être sûr que ces dernières prennent bien en entrée des chiffrés correctement formés.

6.4 Preuves de sécurité

Dans cette section, nous fournissons les preuves de sécurité de notre construction. L'hypothèse de sécurité sous-tendant notre construction se réduit à l'hypothèse de Diffie Hellman tripartie décisionnelle, et nous en fournissant une généralisation afin de simplifier les preuves de sécurité. Ensuite, nous donnons les preuves concernant les trois expériences de sécurité PRV CDA, la cohérence des marqueurs et la cohérence de la déduplication dans le modèle de l'oracle aléatoire. Cette dernière preuve est uniquement une instanciation de notre preuve générale fournie section 6.2.

6.4.1 Hypothèse de sécurité

Dans tout ce qui suit, nous nous placerons dans un environnement bilinéaire $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, e)$. Pour des raisons d'efficacité, nous instancions le système avec e est un couplage de type 3, c'est-à-dire qu'il n'y a pas d'homomorphisme calculable efficacement entre $\mathbb{G}_1$ et $\mathbb{G}_2$. La confidentialité de notre schéma repose sur l'hypothèse suivante :

Hypothèse 6.10 (bl-DDH (Blinded-DDH))**.** *Soit $(t_1, t_1^u, g_1, g_1^r, g_1^z) \in \mathbb{G}_1^5$ et $(t_2, t_2^{u\cdot k}) \in \mathbb{G}_2^2$ pour un triplet aléatoire $(u, r, k) \in (\mathbb{Z}_p^*)^3$. Le problème* bl-DDH *consiste à décider si $z = r \cdot k$ ou si z est un scalaire aléatoire $\mathbb{Z}_p^*$. Nous notons $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{bl\text{-}DDH}}$ l'avantage d'un adversaire $\mathcal{A}$ s'exécutant en temps polynomial contre* bl-DDH*.*

Nous allons prouver que cette hypothèse se réduit dans le cas des couplages symétriques à l'hypothèse Diffie-Hellman Tripartite Décisionnel D3DH [BSW06]. Nous rappelons que cette hypothèse est plus forte dans le cas des couplages symétriques que dans le cas des couplages asymétriques. La réduction fournie implique donc la sécurité de notre schéma en présence d'un environnement bilinéaire de type 3.

Hypothèse 6.11 (Diffie-Hellman Tripartite Décisionnel D3DH)**.** *Soit $g, g^a, g^b, g^c, g^z \in \mathbb{G}$, le problème consiste à décider si $z = abc$ ou z aléatoire.*

Théorème 6.12. *Dans un environnement bilinéaire symétrique, l'hypothèse* D3DH *se réduit à l'hypothèse* bl-DDH*.*

Démonstration. Posons v tel que $t_1 = g_1^v$ et effectuons les changements de variables suivants : $v' = v \cdot u$ et $k' = k \cdot v/v'$. L'hypothèse bl-DDH se réécrit alors comme suit : pour $g_1, g_1^v, g_1^{v'}, g_1^r, g_1^z \in \mathbb{G}_1$ et $\tilde{g}, \tilde{g}^{k'} \in \mathbb{G}_2$, décider si $z = (k'rv)/v'$ ou si z est aléatoire.

Soit $\mathcal{A}$ un adversaire ayant un avantage non négligeable contre l'hypothèse bl-DDH.

Soit $\mathcal{B}$ un adversaire contre D3DH. Remarquons que comme nous sommes dans un environnement bilinéaire de type 1, il est possible de poser $g = g_1 = \tilde{g}$. $\mathcal{B}$ exécute alors l'algorithme 6.1.

Algorithme 6.1 Réduction de D3DH à bl-DDH

1: Soit $\left(g, g^a, g^b, g^c, g^z\right)$ une instance de D3DH
2: $g^v \leftarrow g^a$, $g^r \leftarrow g^b$, $g^{k'} \leftarrow g^c$, et $v' \leftarrow 1$
3: $\mathcal{A} \leftarrow \left(g^v, g^{v'}, g^r, g^{k'}, g^z\right)$
4: $\mathcal{A}$ retourne un bit $b = 1$ si $z = (k'rv)/v'$ et $b = 0$ sinon
5: $\mathcal{B}$ **retourner** b

Comme $z = (k'rv)/v'$ si $z = abc$, $\mathsf{Adv}_{\mathcal{B}}^{\mathsf{D3DH}}(\lambda) \geq \mathsf{Adv}_{\mathcal{A}}^{\mathsf{bl\text{-}DDH}}(\lambda)$. □

Pour simplifier l'écriture de la preuve de sécurité concernant la confidentialité du schéma, nous utilisons la généralisation suivante de l'hypothèse 6.10, que nous appellerons l'hypothèse $(T, \ell)-$bl-DDH. Nous prouvons que l'hypothèse bl-DDH se réduit à l'hypothèse $(T, \ell)-$bl-DDH.

Hypothèse 6.13 ($(T, \ell)-$bl-DDH ((T, ℓ)-blinded-DDH))**.** *Soit T et ℓ deux entiers. Soient $\left[u_h, \{r_{h,i}\}_{i=1}^{\ell}, k_h\right]_{h=1}^{T}$ aléatoires dans $(\mathbb{Z}_p^*)^{T(\ell+2)}$. Étant donné $(t_1, g_1, \ldots, g_\ell) \in \mathbb{G}_1^{\ell+1}$, et $t_2 \in \mathbb{G}_2$, $\left[t_1^{u_h}, \{g_i^{r_{h,i}}\}_{i=1}^{\ell}, \{g_i^{z_{h,i}}\}_{i=1}^{\ell}\right]_{h=1}^{T} \in \mathbb{G}_1^{T(2\ell+1)}$ et $\left[t_2^{u_h \cdot k_h}\right]_{h=1}^{T} \in \mathbb{G}_2^T$, il est difficile de décider si $z_{h,i} = r_{h,i} \cdot k_h$ ou si z est un élément aléatoire de $\mathbb{Z}_p^*$ pour tout $h = 1, \ldots, T$ et pour tout $i = 1, \ldots, \ell$. Nous notons $\mathsf{Adv}_{\mathcal{A}}^{(T,\ell)-\mathsf{bl\text{-}DDH}}(\lambda)$ l'avantage d'un adversaire $\mathcal{A}$ s'exécutant en temps polynomial contre $(T, \ell)-$*bl-DDH*.*

Remarquons que l'hypothèse $(1, 1)$-bl-DDH est exactement l'hypothèse bl-DDH. Nous donnons maintenant la preuve de la réduction de l'hypothèse 6.13 à l'hypothèse 6.10

Théorème 6.14. *Les hypothèses* bl-DDH *et* DDH *se réduisent polynomialement à l'hypothèse* $(T,\ell)-$bl-DDH.

Démonstration. Nous allons procéder en deux temps. Tout d'abord, nous allons montrer qu'un adversaire $\mathcal{A}$ contre l'hypothèse $(T,\ell)-$bl-DDH a le même avantage qu'un adversaire contre l'hypothèse $(1,\ell)-$bl-DDH. Par souci de clarté, nous exposons la démonstration dans le cas $\ell = 1$, mais elle se généralise aisément.

Dans l'hypothèse $(T,1)$-bl-DDH l'adversaire a un avantage non négligeable si pour un h, il arrive à distinguer si $z_{h,1} = r_{h,1} \cdot k_h$ ou si $z_{h,1}$ est aléatoire.

Nous pouvons réécrire une instance de $(T,1)$-bl-DDH comme T instances de bl-DDH comme suit :

$$\begin{cases} \left(t_1, t_1^{u_1}, g_1, g_1^{r_{1,1}}, g_1^{z_{1,1}}\right) & \left(t_2, t_2^{u_1 \cdot k_1}\right) \\ & \vdots \\ \left(t_1, t_1^{u_T}, g_1, g_1^{r_{T,1}}, g_1^{z_{T,1}}\right) & \left(t_2, t_2^{u_T \cdot k_T}\right) \end{cases}$$

Ces instances sont indépendantes les unes des autres : en casser au moins une implique que l'avantage de $\mathcal{A}$ soit majoré ainsi $\mathsf{Adv}_{\mathcal{A}}^{(T,1)-\mathsf{bl\text{-}DDH}} \leq T \cdot \mathsf{Adv}_{\mathcal{A}}^{\mathsf{bl\text{-}DDH}}$.

Maintenant nous nous intéressons à l'hypothèse $(1,\ell)$-bl-DDH.

L'adversaire a un avantage non négligeable si pour un i, il arrive à distinguer si $z_{1,i} = r_{1,i} \cdot k_1$.

Nous pouvons réécrire une instance de $(1,\ell)$-bl-DDH comme ℓ instances de bl-DDH comme suit :

$$\begin{cases} \left(t_1, t_1^{u_1}, g_1, g_1^{r_{1,1}}, g_1^{z_{1,1}}\right) & \left(t_2, t_2^{u_1 \cdot k_1}\right) \\ & \vdots \\ \left(t_1, t_1^{u_1}, g_1, g_1^{r_{1,\ell}}, g_1^{z_{1,\ell}}\right) & \left(t_2, t_2^{u_1 \cdot k_1}\right). \end{cases}$$

Contrairement à $(T,1)$-bl-DDH, ces instances ne sont pas indépendantes les unes des autres. Nous allons écrire la preuve pour le cas $\ell = 2$, la généralisation se fait aisément grâce à un raisonnement par récurrence. Soit $\mathcal{A}$ avec un avantage non négligeable contre l'hypothèse $(1,2)-$bl-DDH. $\mathcal{A}$ peut distinguer avec une probabilité non négligeable $z_1 = r_{1,1} \cdot k_1$ de l'aléatoire ou $z_2 = r_{1,2} \cdot k_1$ de l'aléatoire. Pour alléger les notations, dans la suite de la démonstration, nous ne noterons plus les indices relatifs à $T = 1$. De plus, nous noterons désormais h_i l'élément $g_1^{r_i}$.

Notre instance se réécrit ainsi :

$$\begin{cases} (t_1, t_1^{u_1}, g_1, h_1, h_1^{z_1}) & \left(t_2, t_2^{u_1 \cdot k}\right) \\ (t_1, t_1^{u_1}, g_1, h_2, h_2^{z_2}) & \left(t_2, t_2^{u_1 \cdot k}\right). \end{cases}$$

Le but de l'adversaire est de distinguer $z_1 = k$ de l'aléatoire ou $z_2 = k$ de l'aléatoire. Sans perte de généralité, nous pouvons supposer que nous sommes dans le cas où $\mathcal{A}$ peut distinguer z_1, l'autre cas étant symétrique. Nous avons :

$$\mathsf{Adv}_{\mathcal{A}}^{(1,2)-\mathsf{bl\text{-}DDH}} = |\mathtt{Pr}(\mathcal{A}(\cdots, h_1^{z_1}, h_2^{z_2} | z_1 = k) - \mathtt{Pr}(\mathcal{A}(\cdots, h_1^{z_1}, h_2^{z_2})|.$$

En vertu de l'inégalité triangulaire, nous pouvons alors écrire :

$$\mathsf{Adv}_{\mathcal{A}}^{(1,2)-\mathsf{bl\text{-}DDH}} \leq \mathsf{Adv}_1 + \mathsf{Adv}_2,$$

avec

$$\begin{aligned} \mathsf{Adv}_1 &= |\mathtt{Pr}(\mathcal{A}(\cdots, h_1^{z_1}, h_2^{z_2} | z_1 = k) - \mathtt{Pr}(\mathcal{A}(\cdots, h_1^{z_1}, h_2^{z_1})| \text{ et} \\ \mathsf{Adv}_2 &= |\mathtt{Pr}(\mathcal{A}(\cdots, h_1^{z_1}, h_2^{z_1}) - \mathtt{Pr}(\mathcal{A}(\cdots, h_1^{z_1}, h_2^{z_2})|. \end{aligned}$$

Par hypothèse sur l'adversaire $\mathcal{A}$, au moins l'une des deux valeurs Adv_1 ou Adv_2 doit être non-négligeable.

Supposons que Adv_1 soit non négligeable. Une instance de bl-DDH peut être étendue à l'hypothèse évaluée par Adv_1 en lui ajoutant h_2 : $\mathcal{A}$ a donc un avantage non négligeable contre bl-DDH. Maintenant, supposons que Adv_2 soit non négligeable. Soit $(h_1, h_1^{z_1}, h_2, h_2^{z_2})$ une instance DDH (en réécrivant $h_2 = h_1^x$, déterminer si $z_2 = z_1$ revient à déterminer si $z_2 = x \cdot z_1$). $\mathcal{A}$ peut générer les autres éléments de l'instance évaluée par Adv_2 avec des scalaires connus, et utiliser son avantage non négligeable contre Adv_2 pour résoudre l'hypothèse DDH. □

6.4.2 Confidentialité

De manière informelle, la notion de PRV-piCDA signifie que des chiffrés d'éléments aléatoires et de vrais messages sont indistinguables, tant qu'il y a suffisamment d'entropie dans l'espace des messages. Nous prouvons le théorème 6.16 avec une preuve par jeux [Sho04]. Dans cette preuve, nous utilisons que la distribution des clés issues de la procédure KD est proche de l'uniforme. Pour cela, nous nous appuyons sur le théorème 6.15 qui est une variante du Leftover Hash Lemma proposée par [CV08].

Théorème 6.15. *[CV08, Theorem 3.5] Soit $H : \{1,\dots,2^n\} \to \{1,\dots,2^m\}$ une fonction de hachage tirée aléatoirement dans une famille 2-universelle de fonction de hachage $\mathcal{H}$. Soit $\mathbf{X} = (X_1,\dots,X_T)$ un (T,μ)-source de blocs sur $\{1,\dots,2^n\}^T$. Pour tout $\varepsilon > 0$ tel que $\mu > m + \log(T) + 2\log(1/\varepsilon)$, la suite de hachés $(H, \mathbf{Y}) = (H, H(X_1),\dots,H(X_T))$ est à distance ε de la distribution uniforme sur $\mathcal{H} \times \{1,\dots,2^m\}^T$.*

Théorème 6.16. *Soit ε et μ deux réels non nuls, p un entier premier et T,ℓ deux entiers tels que $\mu > \log p + \log(T) + 2\log(1/\varepsilon)$. Le schéma Λ de la Section 6.3.2 atteint la sécurité* PRV-piCDA *pour (T,μ)-block sources sous l'hypothèse $(T,\ell)-$*bl-DDH *dans le modèle de l'oracle aléatoire.*

Démonstration. Les clés issues de la procédure KD s'écrivent de la manière suivante : $k_M = \prod_{i=1}^{\ell} a_i^{m_i} \mod p$, où les a_i génèrent $\mathbb{Z}_p^*$ pour $i \in \{1,\dots,\ell\}$. Chaque a_i peut être réécrit comme b^{α_i}, avec b un générateur de $\mathbb{Z}_p^*$, et $\boldsymbol{\alpha} = (\alpha_1,\dots,\alpha_\ell)$. Le produit scalaire $\langle\cdot,\cdot\rangle : \left(\mathbb{Z}_p^\ell\right)^T \to \mathbb{Z}_p$ est une fonction de hachage 2-universelle (voir section 3.3.3).

En appliquant le théorème 6.15, on obtient que les clés générées par les messages choisis par l'adversaire suivant une variable aléatoire $\mathbf{M} = (M_1,\dots,M_T)$ sont indistinguables de l'uniforme si $\mathbf{M}$ est une (T,μ)-source de blocs. Plus précisément, si $\mu \geq \log(p) + \log(T) + 2\log(1/\varepsilon)$, alors la distribution $(\boldsymbol{\alpha}, \langle\boldsymbol{\alpha}, M_1\rangle),\dots,\langle\boldsymbol{\alpha}, M_T\rangle)$ est à distance ε de la distribution uniforme sur $\mathbb{Z}_p^\ell \times \mathbb{Z}_p^T$. Ainsi, $((a_1,\dots,a_\ell), k_{M_1},\dots,k_{M_\ell})$ est également à distance ε de la distribution uniforme sur $\mathbb{Z}_p^\ell \times \mathbb{Z}_p^\ell$.

Nous construisons un simulateur $\mathcal{S}$ de l'oracle de chiffrement réel ou fictif contre lequel un adversaire $\mathcal{A}$ échantillonnant polynomialement des (T,μ)-sources de blocs avec au plus q requêtes à l'oracle aléatoire pour le jeu PRV-piCDA a un avantage exactement $\frac{1}{2}$.

Jeu G_0. C'est le jeu original. Soit un adversaire $\mathcal{A}$ capable de casser la sécurité PRV-piCDA de notre schéma. Dans ce jeu, $\mathcal{A}$ choisit une distribution $\mathbb{M}$ sur l'espace des messages M. Il envoie ensuite une requête à l'oracle de chiffrement réel ou fictif avec cette distribution. C'est seulement après sa requête que les paramètres publics du schémas sont générés.

L'oracle choisit une valeur b (réel ou fictif) et produit un vecteur de T messages suivant la distribution $\mathbb{M}$ de l'adversaire $\mathcal{A}$ (mode réel) ou la distribution uniforme sur $\mathcal{M}$ (mode fictif) et le chiffre avant de le renvoyer à l'adversaire. $\mathcal{A}$ a donc accès à un vecteur de T chiffrés :

$$\left[(\tau_h, \{T_{1,i,h}, T_{2,i,h}\}_{i=1}^{\ell}, C_h, \pi_h)_h\right]_{h=1}^{T}.$$

$\mathcal{A}$ doit alors retourner une valeur b' (réel ou fictif), correspondant à la manière dont les messages ont été chiffrés par l'oracle. On note S_i l'événement tel que $b = b'$ dans le jeu G_i.

On a ainsi :

$$\mathsf{Adv}^{\mathsf{PRV-piCDA}}_{\Lambda,\mathcal{A}}(\lambda) = \left|\Pr\left[\mathbf{Exp}^{\mathsf{réel}}_{\Lambda,\mathcal{A}} = 1\right] - \Pr\left[\mathbf{Exp}^{\mathsf{fictif}}_{\Lambda,\mathcal{A}} = 1\right]\right| = 2\left|\mathtt{Pr}(S_0) - \frac{1}{2}\right|.$$

Jeu G_1. Dans ce jeu, $\mathcal{S}$ simule les T preuves non-interactives à divulgation nulle de connaissance présentes dans le vecteur de chiffrés produit par l'oracle de chiffrement réel ou fictif au lieu de les calculer. L'avantage de $\mathcal{A}$ contre la propriété de non divulgation de la NIZK pour la preuve de connaissance du double logarithme discret est noté $\mathsf{Adv}^{\mathsf{zk}}_{\Pi,\mathcal{A}}(\lambda)$. Dans le modèle de l'oracle aléatoire, ces simulations sont indistinguables de preuves réelles pour $\mathcal{A}$, tant qu'aucune collision n'apparaît dans les fonctions de hachage. Les requêtes que $\mathcal{A}$ fait à l'oracle aléatoire dépendent des éléments $\{M_h\}_{i=1}^{T}$ qui eux-mêmes se décomposent en ℓ blocs de ρ bits, et des éléments $\{u_h\}_{h=1}^{T}, \{\{r_{i,h}\}_{i=1}^{\ell}\}_{h=1}^{T}, \{s_h\}_{h=1}^{T}$. On note q_H le nombre de requêtes que $\mathcal{A}$ fait à l'oracle aléatoire et on obtient la probabilité de collisions suivante :

$$|\mathtt{Pr}(S_0) - \mathtt{Pr}(S_1)| \leq \mathsf{Adv}^{\mathsf{zk}}_{\Pi,\mathcal{A}}(\lambda) + \frac{q_H}{2^{\rho\ell T} p^{T(2+\ell)}}.$$

Jeu G_2. Nous nous attaquons maintenant à la génération des clés. Au lieu de calculer les clés conformément à la procédure KD, $\mathcal{S}$ tire des clés aléatoires dans $\mathbb{Z}_p$. Ainsi, à partir de maintenant, les clés utilisées par $\mathcal{S}$ ne dépendent plus des messages qu'il chiffre. D'après le Théorème 6.15, $(k_{M_1}, \ldots, k_{M_T})$ est à distance ε de la distribution uniforme sur $\mathbb{Z}_p^T$, indépendamment de la manière dont les messages ont été générés. On a donc :

$$|\mathtt{Pr}(S_1) - \mathtt{Pr}(S_2)| \leq \varepsilon.$$

Jeu G_3. Dans les parties liées aux marqueurs, l'adversaire a accès aux valeurs $\{t_1^{u_h}\}_{h=1}^{T}$ et $\{t_2^{u_h \cdot k_h}\}_{h_1}^{T}$ qui ne dépendent plus du message chiffré. Dans les parties des T chiffrés se rapportant au chiffrement ElGamal, l'adversaire a accès aux informations suivantes : $T_{1,i} = g_i^{r_{h,i}}$ et $T_{2,i} = g_i^{r_{h,i} \cdot k_h} \cdot g^{m_i}$. Pour chaque chiffré, nous choisissons $z_{h,i}$ de manière aléatoire et uniforme dans $\mathbb{Z}_p$ et remplaçons $T_{2,i} = g_i^{r_{h,i} \cdot k_h} \cdot g^{m_{h,i}}$ par $g_i^{z_{h,i}} \cdot g^{m_{h,i}}$. Ceci est exactement une instance de l'hypothèse $(T,\ell)-\mathsf{bl\text{-}DDH}$, ces deux éléments sont indistinguables. Soit $\mathcal{B}$ un adversaire contre $(T,\ell)-\mathsf{bl\text{-}DDH}$, alors, pour tout $\mathcal{B}$, on a :

$$|\mathtt{Pr}(S_2) - \mathtt{Pr}(S_3)| \leq \mathsf{Adv}^{(T,\ell)-\mathsf{bl\text{-}DDH}}_{\mathcal{B}}(\lambda).$$

Jeu G_4. Dans ce jeu, le simulateur $\mathcal{S}$ se comporte exactement comme l'oracle de chiffrement réel ou fictif, et calcule une mise en gage de Pedersen des valeurs m_i. On a donc :

$$\mathtt{Pr}(S_3) = \mathtt{Pr}(S_4).$$

De plus, l'avantage de $\mathcal{A}$ pour casser l'indistinguabilité du schéma de mise en gage de Pedersen est exactement $\frac{1}{2}$, car ce schéma est parfaitement confidentiel. Ainsi, nous pouvons calculer l'avantage exact de $\mathcal{A}$ contre la sécurité $\mathsf{PRV\text{-}piCDA}$ de notre schéma Λ :

$$\left|\mathtt{Pr}(S_0) - \frac{1}{2}\right| \leq \mathsf{Adv}^{\mathsf{zk}}_{\Pi,\mathcal{A}}(\lambda) + \frac{q_H}{2^{\rho\ell T} p^{T(2+\ell)}} + \varepsilon + \mathsf{Adv}^{(T,\ell)-\mathsf{bl\text{-}DDH}}_{\mathcal{B}}(\lambda),$$

et cet avantage est négligeable. □

6.4.3 Cohérence des marqueurs

Théorème 6.17. *Le schéma Λ décrit dans la Section 6.3.2 a des marqueurs cohérents à condition que la fonction de génération des clés soit résistante aux collisions.*

Démonstration. Soit $\mathcal{A}$ un adversaire ayant un avantage non négligeable dans l'expérience de cohérence des marqueurs.

$$\mathbf{Exp}^{TC}_{\Lambda,\mathcal{A}}(\lambda) = \Pr\left[\mathbf{Exp}^{TC}_{\Lambda,\mathcal{A}} = 1\right] \leq \nu(\lambda).$$

Soit M et c_1 le message et le chiffré fournis par $\mathcal{A}$ dans une expérience gagnante : en particulier, c_1 doit être un chiffré valide. Le challengeur dérive k_M à partir de M et calcule le chiffré $c_0 = \mathsf{Encrypt}(k_M, M)$ et le message $M' = \mathsf{Decrypt}(k_M, c_1)$. De plus, on a $e(t_1^{u_0}, t_2^{u_1 \cdot k_{M'}}) = e(t_2^{u_1}, t_1^{u_0 \cdot k_M})$, avec $k_{M'}$ la clé utilisée pour calculer c_1. Cela implique successivement :

$$\begin{aligned}(e(t_1,t_2)^{u_0 \cdot u_1})^{k_{M'}} &= (e(t_1,t_2)^{u_0 \cdot u_1})^{k_M}\,,\\ k_{M'} &= k_M \mod p,\end{aligned}$$

avec probabilité plus grande que $\nu(\lambda)$.

Par conséquent, $\mathcal{A}$ est capable de trouver des collisions sur la fonction KD avec une probabilité non négligeable, ce qui est impossible car le produit scalaire est une fonction de hachage 2-universelle. □

6.4.4 Cohérence de la déduplication

Théorème 6.18. *Comme le schéma de mise en gage de Pedersen est calculatoirement contraignant, le schéma Λ décrit Section 6.3.2 atteint la propriété de cohérence de la déduplication dans le modèle de l'oracle aléatoire.*

Démonstration. Soit $\mathcal{A}$ un adversaire ayant un avantage non négligeable dans l'expérience de cohérence de la déduplication.

Soit M, c_0 et c_1 le message et les deux chiffrés fournis par l'adversaire $\mathcal{A}$ lors d'une expérience gagnante. On note :

$$c_0 = (\tau_0, \{T^{(0)}_{1,i}, T^{(0)}_{2,i}\}_i, C_0, \pi_0) \text{ et } c_1 = (\tau_1, \{T^{(1)}_{1,i}, T^{(1)}_{2,i}\}_i, C_1, \pi_1).$$

Les deux chiffrés doivent être valides, et donc la preuve NIZK de chacun des chiffrés doit être valide. Notons α et β les messages clairs utilisés par A pour calculer c_0 et c_1. La preuve NIZK assure que la clé utilisée pour calculer c_0 est bien $k_\alpha = \prod_{i=1}^{\ell} a_i^{\alpha_i}$ et que la clé utilisée pour calculer c_1 est bien $k_\beta = \prod_{i=1}^{\ell} a_i^{\beta_i}$. D'autre part, $\mathsf{EQ}(c_0, c_1) = 0$ implique que :

$$(e(t_1,t_2)^{u_0 \cdot u_1})^{k_\alpha} \neq (e(t_1,t_2)^{u_0 \cdot u_1})^{k_\beta}\,.$$

Ainsi, nous avons $k_\alpha \neq k_\beta$, et donc $\alpha \neq \beta$.

Ensuite, le challengeur calcule $k_M = \mathsf{KD}(pp, M)$. Ici, k_M est calculée de manière honnête et donc on a bien $k_M = \prod_{i=1}^{\ell} a_i^{m_i}$. Le challengeur utilise ensuite cette clé k_M pour déchiffrer c_0 et c_1, obtenant m_0 et m_1.

Nous avons supposé que $\mathcal{A}$ gagne l'expérience de la DC, donc que $M = M_0$ et que $M = M_1$. En particulier, c_0 est maintenant un chiffré valide à la fois pour le message M_0 et pour le message α et, de même, c_1 est un chiffré valide à la fois pour le message M_1 et pour le message β. Or, chaque chiffré comporte une mise en gage de Pedersen C_i sur le texte clair. Ainsi, pour gagner l'expérience de la DC, C_0 doit être une mise en gage de Pedersen correcte pour M et pour α et, de même, C_1doit être une mise en gage de Pedersen correcte pour β et pour M avec M différent soit de α, soit de β (sinon, α serait égal à β).

Comme le système de mise en gage de Pedersen est calculatoirement contraignant, cela signifie que $\mathcal{A}$ a un avantage non négligeable pour casser le problème du logarithme discret, ce qui est impossible. □

Algorithme	$(\mathbb{Z}_p)^{\times}$	$\mathbb{G}_1$	$[\times](\mathbb{G}_1)$	$[\times](\mathbb{G}_2)$	$(\mathbb{G}_T)^{\times}$	Pairing
Chiffrement	$\ell+4$	–	$6\ell+2$	1	$2\lambda+1$	1
Déchiffrement	$\ell+4$	$\ell\sqrt{2^\rho}$	$3\ell+4$	–	$2\lambda+3$	2
Test d'égalité	–	–	–	–	–	$2N$

Table 6.1 – Nombre d'opérations requises pour chaque étape, où N est le nombre de chiffrés dans la base de données, $[\times](\mathbb{G})$ est la multiplication scalaire dans le groupe $\mathbb{G}$ et $(\mathbb{G})^{\times}$ l'exponentiation modulaire dans $\mathbb{G}$.

6.4.5 Efficacité

Comme [ABM+13] et [BK15], nous améliorons la sécurité du chiffrement convergent, ce qui entraîne une perte d'efficacité. Nous sommes plus efficaces que [ABM+13] et [BK15], qui utilisent des NIZK génériques et du chiffrement totalement homomorphe, mais la comparaison formelle reste difficile car les trois schémas atteignent des propriétés de sécurité différentes.

Les étapes de notre schéma les plus gourmandes en temps et en mémoire sont les suivantes ;

— Le déchiffrement des ℓ chiffrés ElGamal $(T_{1,i}, T_{2,i})$, $1 \leq i \leq \ell$, qui nécessite de calculer ℓ logarithme discrets pour retrouver chaque m_i.

— Le calcul de la NIZK sur le double logarithme discret $e(\tau_1, t_2)^{C_1} = e(t_1, \tau_2)^{x^s}$, qui nécessite λ exponentiations dans $\mathbb{G}_T$ et λ exponentiations dans $\mathbb{Z}_p$.

Le tableau 6.1 résume le nombre d'opérations nécessaires dans chaque groupe pour chaque étape de notre schéma.

Choix du paramètre ℓ. Le paramètre ℓ correspond au nombre de blocs qui résultent de la division du message. Ainsi, il détermine le nombre de chiffrés ElGamal et le nombre d'exponentiations dans la NIZK de la mise en gage et donc ne doit pas être trop important. Néanmoins, plus ℓ est petit, plus ρ, le nombre de bits de chaque bloc, est important, rendant le calcul du logarithme discret au déchiffrement plus lent.

En effet, au déchiffrement ElGamal, nous obtenons $h_i = h^{m_i}$ ce qui nécessite le calcul de $m_i = \log_h h_i$ pour retrouver m_i. Ce calcul n'est possible efficacement que si m_i est relativement petit. Plus précisément, avec un algorithme pas de bébé, pas de géant, nous pouvons retrouver un $m - i$ de taille ρ en $\mathcal{O}(\sqrt{2^\rho})$ opérations de groupe (qui se font dans notre cas dans $\mathbb{G}_1$).

Ainsi, nous cherchons un compromis sur ℓ qui ne doit être ni trop petit pour permettre des NIZKs et le chiffrement marquant ElGamal, mais suffisamment grand pour que la taille ρ de chaque m_i soit relativement petite pour le calcul du logarithme discret. Nous estimons qu'une exponentiation prend environ $\frac{3}{2}\lambda$ opérations de groupe. Ainsi, si nous voulons fixer le temps de calcul du logarithme discret à au plus t fois le temps d'une exponentiation modulaire, il résulte que nous devons prendre

$$\rho \approx \log_2 3t^2\lambda^2 - 2.$$

Par exemple, pour $t = 10$ et $\lambda = 128$, nous pouvons prendre $\rho = 21$.

6.5 Conclusion

Dans ce chapitre, nous avons formalisé la cohérence de la déduplication comme nouvelle propriété de sécurité à atteindre pour un schéma verrouillé par le message. Cette propriété apporte au serveur une manière de vérifier que le chiffrement a été fait correctement par les utilisateurs. Cela lui assure que le processus de déduplication s'effectuera sans heurts et que chaque copie gardée sur le serveur est effectivement unique.

Les perspectives pour ce travail sont multiples. La sécurité de notre schéma n'est pour l'instant prouvée que dans le modèle de l'oracle aléatoire. La conception d'un schéma MLE vérifiable dans le modèle standard reste ainsi un problème ouvert. De plus, nous pouvons nous poser la

question de la combinaison des différentes propriétés de sécurité existantes pour les schémas MLE. Par exemple, est-il possible d'obtenir une instanciation de MLE atteignant la cohérence de la déduplication, assurant également la sécurité pour des messages dépendant des paramètres publics du schéma et qui ne repose pas sur des NIZK génériques ? Enfin, il est également important de travailler sur l'efficacité de ces primitives cryptographiques car les protocoles proposés actuellement restent trop lourds pour espérer un déploiement effectif.

Chapitre 7

Détection d'intrusions sur du trafic chiffré

Sommaire

Le but de ce chapitre est de décrire des protocoles cryptographiques permettant la détection d'intrusions sur du traffic chiffré. Ceci répond à une problématique actuelle majeure. En effet, le déploiement du protocole HTTPS et du chiffrement des communications est vital pour répondre aux problèmes de confidentialité des échanges sensibles sur internet. Cette confidentialité est maintenant devenue nécessaire dans la vie quotidienne des internautes : connexion à des sites sensibles comme sa banque, paiement en ligne... Mais dans le même temps, le protocole HTTPS empêche le travail d'applications de sécurité fondamentales : en effet, un trafic sain est indistinguable d'un trafic corrompu. Certains systèmes ont même recours à une attaque d'homme-au-milieu, générant de faux certificats SSL au niveau du proxy chargé de réaliser l'analyse de sécurité. Il déchiffre alors tout le trafic afin de détecter d'éventuelles intrusions malveillantes [HREJ14, Jar12]. Une première solution plus adaptée à ce problème a été proposée

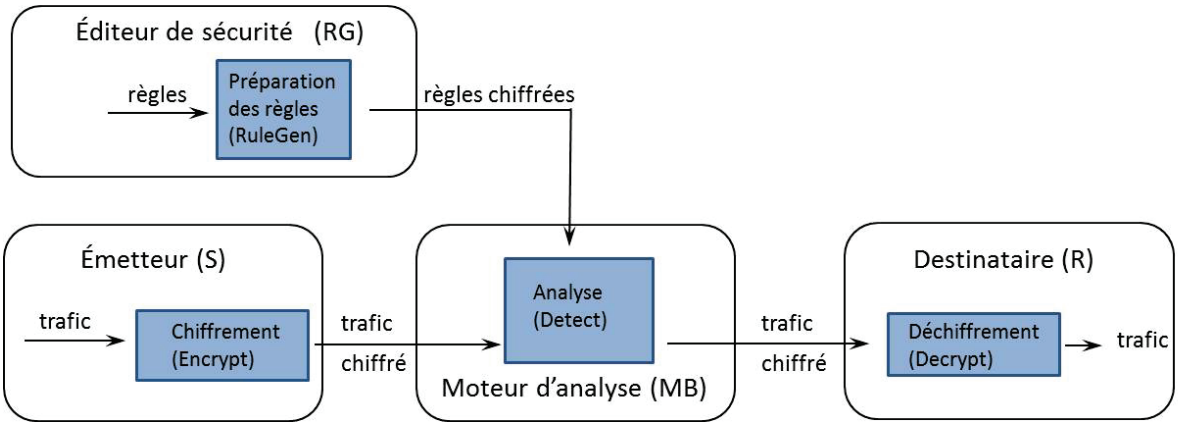


Figure 7.1 – Architecture de BlindIDS

par [SLPR15], s'appuyant sur des outils de calcul multi-parties comme le transfert en aveugle ou les circuits embrouillés de Yao (voir section 4.4). Dans ce chapitre, nous présentons cette solution et les limitations que nous lui avons trouvées. Ensuite, nous présentons deux protocoles répondant à ces limitations, l'un basé sur du chiffrement cherchable et l'autre basé sur le chiffrement complètement homomorphe. Ces résultats ont donné lieu à deux brevets français et ont été publiés à AsiaCCS 2017 [CDKP16b, CDKP16a, CDK+17].

7.1 Architecture de la détection d'intrusions

7.1.1 Acteurs considérés

Le service de détection d'intrusions (IDS) s'intègre à un réseau de communication sur lequel un émetteur (S) envoie un message à un destinataire (R). Le service de détection d'intrusions est lui-même divisé en deux entités. Le moteur d'analyse (MB) intercepte les messages échangés sur le réseau et utilise des règles générées par un éditeur de règles (RG) pour chercher une éventuelle attaque. En cas d'attaque, le moteur d'analyse peut soit renvoyer une alerte avec le message, ce qui est le cas le plus courant, soit décider d'interrompre complètement la transmission. Ces acteurs sont représentés figure 7.1.

7.1.2 Règles considérées

Les règles générées par l'éditeur de sécurité sont de plusieurs types selon leurs fonctionnalités : domaines de logiciels malveillants, ensemble d'URL pour le contrôle parental, les règles Yara et Snort pour l'inspection proprement dite des paquets.

Les listes noires de maliciels [Mal] et les listes noires d'URL pour le contrôle parental [URL] ont le même mode de fonctionnement bien qu'elles couvrent des cas d'usages différents. Ces sortes de listes se prêtent parfaitement bien à la recherche de correspondances et sont entièrement couvertes par notre première solution à base de chiffrement cherchable, garantissant un taux de détection de 100% avec celle-ci. Les règles Yara [Yar] et Snort [Sno] incluent en plus de ces mots-clés des expressions rationnelles à évaluer sur le trafic (voir chapitre 2). Néanmoins, 77.3% des règles Yara et 75% des règles Snort ne contiennent pas d'expressions rationnelles et sont donc intégralement couvertes pas notre première solution. C'est pour évaluer les règles restantes que nous présentons une solution théorique basée sur le chiffrement homomorphe. Ces résultats sont résumés figure 7.2.

Exemple 1 : Un exemple de règle est donné figure 7.3. Il s'agit ici de reconnaître une injection SQL basique. Le scénario de l'attaque est le suivant. Lorsqu'on lui demande son identifiant et son

Ensemble	Nbre d'entrées	Entrées supportées par DSE	%
Liste noire de maliciels [Mal]	1,250	1,250	100%
Liste noire d'URL [URL]	4,546,341	4,546,341	100%
Règles Yara [Yar]	256	198	77.3%
Règles Snort [Sno]	3,467	2,606	75%

FIGURE 7.2 – Ensembles de règles évalués par le proxy. La troisième colonne correspond au taux de détection de notre première solution. Il est à noter que notre deuxième solution supporte 100% des règles, quelque soit l'ensemble.

```
alert tcp any any -> $HOME_NET $PORT_HTTP (msg: "SQL Injection Attempt - or 1=1"; content:
    "GET"; http_method; uricontent: "or 1=1"; pcre:"/(\%27)|(\')|(\-\-)|(%23)|(#)/i"; nocase;
    classtype:web-application-attack; sid:3000001; rev:1;)
```

FIGURE 7.3 – Règle Snort [Thr]

mot de passe, l'attaquant rentre dans le champ mot de passe une chaîne de type "motdepasse or 1=1". Le serveur auquel il essaie de se connecter vérifie alors la requête SQL suivante : "motdepasse = motdepassevéritable or 1=1". Ce prédicat prenant toujours la valeur "vrai", il est ainsi possible de contourner la procédure d'authentification. Le champ "pcre" contient une expression rationnelle à évaluer sur le trafic. Ici, cette expression vise à détecter que la chaîne de caractères 1=1 est suivie d'une apostrophe, qui permet de fermer la requête SQL, et d'une ou plusieurs instructions (- -, #) visant à commenter tout code SQL qui pourrait suivre la requête frauduleuse. La détection de ce type d'attaque conduit à la génération de l'alerte "SQL Injection Attempt - or 1=1".

7.1.3 Interactions

Un schéma de détection d'intrusions sur du trafic chiffré qui implique un émetteur S, un destinataire R, un moteur d'analyse MB et un éditeur de sécurité RG est composé des cinq procédures suivantes.

- La mise en place Setup, qui prend en entrée le paramètre de sécurité λ, et génère les paramètres publics pp du système, et les clés des acteurs.
- La génération des règles $\mathsf{RuleGen}$, qui prend en entrée les paramètres publics pp, la clé secrète $\mathsf{sk}_{\mathsf{RG}}$ de l'éditeur de sécurité RG et un ensemble de règles $\mathcal{R}$ à évaluer sur le trafic. Elle renvoie un ensemble de trappes qui sont transmises au moteur d'analyse.
- L'envoi $\mathsf{Encrypt}$ prend en entrée les paramètres pp, la clé publique du destinatire pk_{R} et un trafic T. Elle renvoie un un trafic chiffré à l'intention du destinataire R.
- La procédure de détection Detect qui prend en entrée pp, un trafic chiffré et l'ensemble des trappes générées par le RG et renvoie un bit, 0 si le trafic est sain et 1 s'il est malveillant. Elle peut aussi retourner une information auxiliaire aux. Enfin, elle peut retourner un message d'erreur $\perp$.
- La réception $\mathsf{Decrypt}$ prend en entrée les paramètres pp, la clé secrète du destinataire sk_{R} et un trafic chiffré. Elle renvoie le trafic clair ou un message d'erreur $\perp$.

7.2 La solution BlindBox

Dans [SLPR15], les auteurs proposent la première solution de détection d'intrusions sur du trafic chiffré. Ils concilient deux propriétés qui paraissaient jusqu'alors antagonistes : la confidentialité du trafic et la surveillance par des applications de sécurité. Nous présentons d'abord

l'architecture de leurs protocoles, puis décrivons brièvement les trois protocoles proposés, avant de nous intéresser au modèle de sécurité considéré.

7.2.1 Protocoles

Dans [SLPR15], les auteurs proposent trois protocoles ayant différentes capacités de détection. Le premier protocole considère uniquement la recherche de mots-clés, le second étend ce premier protocole aux combinaisons de mots-clés et le troisième a les mêmes capacités qu'un IDS sur du trafic en clair. Comme les tests d'égalité sur les données chiffrées ne fonctionnent que lorsque le mot à chercher et le mot à tester ont la même longueur, la première étape du protocole est de découper le trafic en unités qui seront analysées par le moteur d'analyse.

Découpe du trafic

Deux approches sont proposées dans [SLPR15].

— Découpe *à taille fixe*. Il s'agit de fixer la taille des unités et de créer à chaque caractère dans le trafic une unité de cette taille. Par exemple, la chaîne "ceci est un virus" sera découpée en "ceci", "eci ", "ci e", etc. Si un mot-clé dépasse cette taille fixée, cette méthode impose de le découper également. Par exemple, le mot-clé virus peut être découpé en "viru" et "irus".

— Découpe *selon les délimiteurs*. Les auteurs remarquent que dans du trafic fortement structuré, comme HTTP, les mots à rechercher ont peu de chances d'être distribués aléatoirement dans le trafic. Ils sont en fait encadrés par des délimiteurs, comme des signes spéciaux de ponctuation, ou des balises, et la découpe peut se faire en suivant ces délimiteurs.

Cette étape de découpe est fondamentale lorqu'on s'intéresse à la recherche de correspondance. En effet, les protocoles de chiffrement ne sont capables de supporter la recherche d'égalité qu'entre deux chiffrés ayant exactement le même message clair. Ainsi, les taux donnés figure 7.2 correspondent à une détection théorique. La détection des attaques dépend de la méthode de découpe du trafic utilisée. En pratique, la méthode de découpe à taille fixe crée une unité à chaque caractère du trafic, permettant une détection maximale. Néanmoins, elle génère un surcoût important. En effet, le trafic est chiffré de multiples fois. La découpe selon les délimiteurs est plus intéressante car le trafic n'est chiffré qu'une seule fois. Son taux de détection, en revanche, n'est pas garanti et doit être déterminé expérimentalement. En effectuant des expériences similaires à celles menées par BlindBox sur un ensemble fourni par l'ICTF [Vig] lors d'un concours de capture de drapeau, incluant des attaques variées, nous obtenons des résultats de détection semblables aux leurs, soit 96.5% des mots-clés détectés et 98.3% des règles qui auraient été détectées par Snort, ce qui valide l'efficacité de la méthode de découpe selon les délimiteurs.

Schémas

Le premier protocole BlindBox I utilise les techniques de calcul multi-parties décrites section 4.4. Celui-ci ne permet que de chercher dans le trafic un mot-clé r donné. L'émetteur S découpe le trafic en unités t, puis encode chaque unité de manière probabiliste avec une clé de session k. L'aléa utilisé est connu du moteur d'analyse MB. Nous parlons d'encodage et non de chiffrement car ces encodages ne sont pas réversibles. En plus de ces encodages, S envoie donc le trafic chiffré avec la clé de session k.

Le moteur d'analyse se met en coupure entre l'émetteur et le destinataire. Lorsqu'il intercepte le trafic, il engage un calcul de circuit embrouillé avec S et R afin d'obtenir un circuit $\mathcal{C}$ permettant d'encoder une règle r sous la clé k sans apprendre la valeur de cette dernière. Le calcul du circuit doit être effectué par les deux extrémités du réseau afin que le moteur d'analyse puisse vérifier qu'il a été effectué honnêtement en comparant les deux résultats. Enfin, pour chaque règle r, le moteur engage un protocole de transfert inconscient afin de générer les entrées brouillées de

$\mathcal{C}$. De même, ce protocole est effectué avec les deux extrémités du circuit. Grâce à $\mathcal{C}$ et les entrées brouillées obtenues, le moteur d'analyse peut calculer les encodages des règles et tester la présence de mots-clés suspects dans le trafic échangé par S et R. Il transmet ensuite le trafic chiffré et les encodages des unités à R, éventuellement avec une alerte.

Le destinataire R déchiffre le trafic et reconstitue à partir de ce dernier les encodages de chaque unité. Il vérifie la correspondance des encodages reconstitués avec ceux transmis par le moteur d'analyse, afin d'avoir l'assurance que le trafic a été effectivement analysé.

Le deuxième protocole BlindBox II repose sur le même principe. La position de chaque unité est maintenant attachée à celle-ci. Le moteur d'analyse peut ainsi gérer des règles précisant la position à laquelle un mot-clé doit apparaître pour que le trafic soit considéré comme malveillant.

Enfin, le troisième protocole BlindBox III permet l'exécution d'un IDS complet sur le trafic. La particularité de ce protocole est de permettre au moteur d'analyse de déchiffrer *intégralement* le trafic en cas de présence de mots-clés suspects. À cette fin, la clé de connexion SSL est intégrée dans les règles chiffrées et peut être récupérée en cas de correspondance.

7.2.2 Propriétés de confidentialité

Dans cette section, nous présentons tout d'abord les types d'attaques auxquels le protocole peut être confronté puis le modèle de confidentialité garanti par BlindBox.

Tout d'abord, les auteurs font l'hypothèse essentielle que soit l'émetteur S, soit le destinataire R est honnête. Cette hypothèse peut paraître restrictive. En effet, une attaque très répandue aujourd'hui est la prise de contrôle par un logiciel malveillant du terminal émetteur. Ce dernier est alors en capacité de faire fuir des données potentiellement confidentielles vers l'attaquant distant qui l'a déployé. Symantec pointe dans son rapport 2015 [Sym15] sur les menaces qu'il est de plus en plus courant pour les attaquants d'injecter du code malveillant dans des mises à jour de logiciels inoffensifs ou encore d'avoir recours à des emails de phishing pour inciter des utilisateurs à télécharger et exécuter du contenu malveillant. Aussi inquiétantes que soient ces menaces, il est toujours possible pour ces attaquants de convenir d'une clé de chiffrement avec le terminal infecté qu'il contrôle via le logiciel malveillant. Ce dernier peut alors chiffrer symétriquement les données qu'il cherche à exfiltrer, échappant ainsi à toute technique de détection d'attaque, que celle-ci ait lieu sur du trafic en clair ou chiffré. Ainsi, bien qu'elle soit assez forte, l'hypothèse selon laquelle au moins une des deux extrémités du réseau est honnête est donc inhérente à tout système de détection d'intrusions, même sur du trafic clair. Elle est donc indispensable et nous la reprendrons dans notre modèle.

Concernant le modèle de confidentialité, deux modèles sont proposés dans [SLPR15]. Le premier modèle est la "confidentialité sauf correspondance" (*exact match privacy*). Dans ce modèle, le trafic reste confidentiel à condition qu'aucune unité ne corresponde pas à un des mots-clés identifié par l'éditeur de règles comme indicateur d'une possible attaque. Dans ce cas, et seulement dans ce cas, le moteur d'analyse apprend la présence de ce mot-clé dans le trafic, mais ne connaît rien du reste du trafic (sinon qu'il ne correspond pas aux mots-clés recherchés). Ce modèle peut être problématique dans le cas de règles de détection avec des mots-clés multiples. Si la présence d'une partie des mots-clés seulement n'est pas distinctive d'une attaque, le moteur d'analyse apprend la présence de certains mots-clés dans du trafic sain, portant ainsi atteinte à la confidentialité d'utilisateurs honnêtes.

Le deuxième modèle est la "confidentialité sous réserve de cause probable" (*probable cause privacy*). Ce modèle concerne le protocole III, et comme nous l'avons vu dans la description de celui-ci, la présence de certains mots-clés permet le déchiffrement du trafic entier pour une inspection plus complète. Ce déchiffrement ne contribue donc pas à détecter des intrusions en plus de celles identifiables par simple recherche de motifs, mais seulement à affiner la détection de faux positifs. Or, les faux positifs ne présentent pas une menace suffisante pour le réseau pour justifier un déchiffrement complet du trafic. En effet, leur seule conséquence est de générer

des alertes supplémentaires qui doivent être traitées par l'utilisateur. Cela entraîne certes une charge de travail supplémentaire pour le récepteur, mais n'est pas un danger pour ce dernier.

Ainsi, ces deux modèles de confidentialité ne nous paraissent pas entièrement satisfaisants. Dans la suite de ce chapitre, nous proposons de nouveaux modèles protégeant mieux la confidentialité du trafic. De plus, les schémas présentés dans [SLPR15] souffrent d'autres limitations.

7.2.3 Limitations des protocoles BlindBox

Bien que le protocole proposé par [SLPR15] permette un grand pas dans la détection d'intrusions sur du trafic chiffré, les limitations de leurs schémas sont nombreuses. Nous avons notamment identifié les quatre suivantes.

Limitations de performance. Dans le protocole de [SLPR15], l'étape de chiffrement des règles est entièrement dépendant des clés choisies par l'émetteur et le récepteur. Cela signifie qu'à chaque nouvel établissement de connexion sécurisée, l'ensemble des règles de détection doit être rechiffré par le moteur d'analyse. Cela nécessite le chiffrement de centaines, voire de milliers de règles à chaque établissement d'une nouvelle connexion, en faisant appel à des techniques complexes comme le transfert inconscient ou les circuits embrouillés. De plus, ces règles chiffrées doivent être gardées en mémoire tout au long de la connexion SSL. L'espace mémoire consommé par le moteur d'analyse croît donc polynomialement avec le nombre de règles utilisées et le nombre de connexions simultanées, ce qui pose la question de la mise à l'échelle du protocole.

Limitations de confidentialité. Dans le schéma de [SLPR15], le moteur d'analyse a accès aux règles produites par l'éditeur de sécurité en clair. Il est assez peu réaliste de considérer que des éditeurs de sécurité soient prêts à donner leurs règles en clair à des tierces personnes [MM15, HBB15]. Celles-ci constituent en effet le cœur de métier de l'éditeur. Le modèle économique de ces éditeurs est d'ailleurs fondé sur la valeur de leurs règles : c'est un des critères principaux d'évaluation de leur travail [SPV+16]. Pour les détections sur du traffic en clair, ces règles sont d'ailleurs implantées de manière très bas niveau dans les moteurs d'analyse, afin de protéger le travail des éditeurs : un lourd travail de rétro-ingénierie est nécessaire si l'on veut les retrouver. Ainsi, l'approche proposée par [SLPR15] semble peu compatible avec les intérêts des éditeurs de sécurité actuels.

Limitations de sécurité. Le modèle proposé dans [SLPR15] considère qu'au moins l'une des deux extrémités du trafic reste honnête. Comme nous l'avons vu précédemment, ceci n'est pas restreignant dans le sens où deux extrémités malhonnêtes peuvent collaborer pour éviter le processus de détection, même en clair. Néanmoins, dans le mode de réalisation de [SLPR15] le trafic est envoyé deux fois : une fois en entier, et une fois divisé par unités lexicales à des fins d'analyse. Cette redondance est délicate à gérer dans le cas d'un émetteur malhonnête. En effet, c'est au destinataire que revient la tâche de reconstituer le trafic et de vérifier que les unités inspectées correspondent bien au trafic entier. Il y a donc un risque pour le destinataire honnête de recevoir un trafic malveillant sans alerte. La redondance entre le trafic chiffré et le trafic inspectée doit donc être si possible limitée.

Limitations de fonctionnalité. Si le troisième protocole proposé par [SLPR15] permet bien d'exécuter un full IDS, il le fait au prix d'un déchiffrement complet du trafic. Comme mentionné précédemment, cela n'élargit pas le champ de détection des attaques, mais permet uniquement le repérage de faux-positifs. Or, les faux-négatifs, c'est-à-dire des attaques non détectées, sont beaucoup plus nuisibles pour les utilisateurs et BlindBox ne propose pas de solution pour les limiter.

Dans la suite de ce chapitre, nous proposons deux protocoles répondant à tout ou partie de ces préoccupations. En particulier, nous cherchons à atteindre le modèle de sécurité décrit dans la section suivante.

7.3 Modèle de sécurité

Dans cette section, nous présentons les modèles de sécurité atteints par nos schémas. Nous considérons deux types d'adversaire qui ont des buts distincts. La première expérience de sécurité modélise le cas où l'une des extrémités du réseau est malhonnête et cherche à éviter le processus de détection d'attaque. La deuxième concerne l'indistinguabilité du trafic vis-à-vis du moteur d'analyse.

Hypothèses

Dans cette section, nous détaillons les hypothèses faites sur le comportement des parties, et notamment les informations auxquelles elles tentent d'accéder.

Moteur d'analyse. Le moteur d'analyse suit le modèle *honnête-mais-curieux*. Il effectue la détection correctement sur le trafic chiffré, en utilisant les signatures fournies par l'éditeur de sécurité. En revanche, il va essayer d'obtenir des informations supplémentaires à la fois sur le trafic chiffré et sur les mots-clés cachés derrière les signatures envoyées par l'éditeur de sécurité.

Éditeur de sécurité. Ce rôle est également tenu par une entité honnête-mais-curieuse. Les signatures fournies au moteur d'analyse sont effectivement des signatures conçues pour détecter un trafic malveillant et non pour obtenir des informations supplémentaires sur le trafic. Cette hypothèse semble raisonnable : dans un marché hautement concurrentiel, un éditeur de sécurité ne va pas risquer sa réputation en générant des signatures ne correspondant pas à de réelles attaques. En revanche, l'éditeur de sécurité peut être curieux, c'est-à-dire qu'il peut écouter le trafic chiffré à la recherche d'informations sur le texte en clair.

Collusion entre le MB et le RG. Dans notre modèle, nous supposons qu'il n'existe pas de collusion entre ces parties. En effet, une telle collusion pourrait leur permettre d'obtenir de nombreuses informations sur le trafic, notamment si le moteur d'analyse demande à l'éditeur de sécurité de générer des trappes en fonction du trafic inspecté. L'entente, selon le protocole utilisé, peut aussi mener à un déchiffrement total du trafic. Cette hypothèse nous semble aussi raisonnable, pour la même raison que la précédente.

Émetteur et destinataire. Nous faisons l'hypothèse que l'une des deux extrémités du réseau est honnête. Comme détaillé section 7.2.2, cette hypothèse, bien que forte, est inhérente à la détection d'intrusions. De plus, pour simplifier l'exposé, nous supposons que c'est l'émetteur qui est malhonnête, bien que nos protocoles et nos preuves s'adaptent au cas inverse.

7.3.1 Détection

Sous l'hypothèse qu'au moins l'une des deux extrémités du réseau est honnête, notre schéma atteint l'expérience de sécurité décrite figure 7.4. De manière informelle, cette expérience permet de s'assurer que tous les trafics malveillants détectés par l'IDS en clair le sont également en chiffré. Dans la description de l'expérience, nous utiliserons la notation Detect pour désigner l'étape correspondant à la détection d'intrusions. Selon l'entrée de Detect, il sera implicite que cette détection se fait sur du trafic chiffré, en suivant notre protocole, ou sur du trafic clair, en utilisant les mêmes règles que pour le trafic chiffré. Le but pour $\mathcal{A}$ est de produire un trafic chiffré E tel que $\mathsf{Detect}(\mathsf{pp}, E, \mathcal{B}) = 0$ (donc, que le moteur d'analyse déclare sain sous l'ensemble

Expérience $\mathbf{Exp}^{det}_{\mathcal{A}}(\lambda)$
$(\mathsf{pp}, \mathsf{sk}_{\mathsf{RG}}, \mathsf{sk}_{\mathsf{R}}) \leftarrow \mathsf{Setup}(1^\lambda)$;
$\mathcal{B} \leftarrow \mathsf{RuleGen}(\mathsf{pp}, \mathsf{sk}_{\mathsf{RG}}, \mathcal{R})$;
$E \leftarrow \mathcal{A}(1^\lambda, \mathsf{pp})$;
Si $\mathsf{Detect}(\mathsf{pp}, E, \mathcal{B}) = 1$, retourner 0 ;
$T \leftarrow \mathsf{Decrypt}(\mathsf{pp}, \mathsf{sk}_{\mathsf{R}}, E)$;
Si $\mathsf{Detect}(T, \mathcal{R}) = 0$, retourner 0 ;
retourner 1 ;

Figure 7.4 – Jeu de la détection : $\mathbf{Exp}^{det}_{\mathcal{A}}(\lambda)$

Expérience $\mathbf{Exp}^{\mathsf{T-IND}}_{\mathcal{A}}(\lambda)$
$(\mathsf{pp}, \mathsf{pk}) \leftarrow \mathsf{Setup}(1^\lambda)$;
$\mathcal{B} \leftarrow \mathsf{RuleGen}(\mathsf{pp}, \mathsf{sk}_{\mathsf{RG}}, \mathbb{R})$;
$T_0, T_1 \leftarrow \mathcal{A}(1^\lambda, \mathsf{pp})$;
Si $\mathsf{Detect}(\mathsf{pp}, \mathsf{Encrypt}(T_0, \mathsf{pk}_{\mathsf{R}}), \mathcal{R}) \neq \mathsf{Detect}(\mathsf{pp}, \mathsf{Encrypt}(T_1, \mathsf{pk}_{\mathsf{R}}), \mathcal{B})$ retourner 0 ;
$b \xleftarrow{\$} \{0, 1\}$;
$c_b \leftarrow \mathsf{Encrypt}(T_b, \mathsf{pk}_{\mathsf{R}})$;
$b' \leftarrow \mathcal{A}(c_b)$;
Si $b' \neq b$, retourner 0 ;
Retourner 1.

Figure 7.5 – Jeu de l'indistinguabilité du trafic : $\mathbf{Exp}^{\mathsf{T-IND}}_{\mathcal{A}}(\lambda)$

des règles chiffrées $\mathcal{B}$) alors que la version déchiffrée T de E est malveillante avec les règles non chiffrée $\mathcal{R}$ correspondantes ($\mathsf{Detect}(T, \mathcal{R}) = 1$).

Ainsi, nous disons qu'un protocole de détection d'intrusions sur du trafic chiffré *détecte* (sous-entendu, correctement) le trafic malveillant si pour tout adversaire probabiliste en temps polynomial $\mathcal{A}$, il existe une fonction négligeable $\nu(\lambda)$ telle que :

$$\mathsf{Adv}^{det}_{\mathcal{A}}(\lambda) = \Pr\left[\mathsf{Exp}^{det}_{\pi,\mathcal{A}} = 1\right] \leq \nu(\lambda).$$

7.3.2 Indistinguabilité du trafic

Avec cette expérience de sécurité, nous souhaitons nous assurer que la seule information obtenue par le moteur d'analyse sur les chiffrés interceptés soit la présence ou l'absence d'attaque (et comme nous allons voir, éventuellement une information auxiliaire). L'expérience pour notre schéma diffère quelque peu de l'indistinguabilité classique, car le moteur d'analyse a accès à un ensemble de trappes générées par l'éditeur de règles. Or, la procédure de détection peut retourner une information auxiliaire en plus de 0 ou 1. Cette information peut être la trappe qui a déclenché l'alerte. Ceci implique que l'indistinguabilité de deux chiffrées n'a de sens qu'entre des chiffrés de même *type* : soit ils correspondent tous les deux à une attaque, et dans ce cas, celle-ci doit générer la même information auxiliaire sinon l'adversaire peut distinguer les chiffrés, soit ils sont tous les deux sains. Dans le cas où l'un correspond à une attaque et l'autre non, l'indistinguabilité est en effet trivialement non atteinte. Nous décrivons notre expérience figure 7.5.

Ainsi, nous disons qu'un protocole de détection d'intrusions est *trafic-indistinguable* si pour tout adversaire probabiliste en temps polynomial $\mathcal{A}$, il existe une fonction négligeable $\nu(\lambda)$ telle que :

$$\mathsf{Adv}^{\mathsf{T-IND}}_{\pi,\mathcal{A}}(\lambda) = \left|2 \cdot \Pr\left[\mathsf{Exp}^{\mathsf{T-IND}}_{\pi,A} = 1\right] - 1\right| \leq \nu(\lambda).$$

$\boxed{\text{Experiment } \mathsf{Exp}^{\mathsf{R-IND}}_{\mathcal{A}}(\lambda)}$
$b \leftarrow \{0,1\}$;
$(\mathsf{pp}, \mathsf{sk}_{\mathsf{RG}}, \mathsf{sk}_{\mathsf{R}}) \leftarrow \mathsf{Setup}(1^\lambda)$;
$r_0, r_1 \leftarrow \mathcal{A}_f(1^\lambda, \mathsf{pp})$;
$\mathcal{B}_b \leftarrow \mathsf{RuleGen}(\mathsf{pp}, \mathsf{sk}_{\mathsf{RG}}, r_b)$;
$b' \leftarrow \mathcal{A}_g(\mathsf{sk}_{\mathsf{R}}, \mathcal{B}_b)$;
Retourner $(b = b')$;

Figure 7.6 – Jeu de l'indistinguabilité de règles : $\mathsf{Exp}^{\mathsf{R-IND}}_{\mathcal{A}}(\lambda)$

7.3.3 Indistinguabilité des règles

Enfin, nous souhaitons modéliser le fait que le moteur d'analyse n'apprend pas d'information sur les règles : c'est ce que nous appelons l'indistinguabilité des règles. Comme pour l'indistinguabilité du trafic, le moteur d'analyse peut créer des trafics et les tester à l'aide des trappes qu'il a obtenu de l'éditeur de règle. Il peut donc mener une attaque par brute force sur l'ensemble des règles. Comme dans le modèle de sécurité du chapitre 6, nous nous limitons donc aux règles qui présentent suffisamment d'entropie-min pour résister à cette attaque.

L'expérience est décrite 7.6, pour un adversaire $\mathcal{A} = (\mathcal{A}_f, \mathcal{A}_g)$ tel que $\mathcal{A}_f$ et $\mathcal{A}_g$ ne puissent pas communiquer. $\mathcal{A}_f$ choisit deux règles r_0 et r_1, et l'une des deux est utilisée dans la procédure $\mathsf{RuleGen}$. La sortie de $\mathsf{RuleGen}$, $\mathcal{B}_b$, est donnée à $\mathcal{A}_g$, qui doit sortir le bit b.

Ainsi, nous disons qu'un protocole de détection d'intrusions est *règle-indistinguable* si pour tout adversaire probabiliste en temps polynomial $\mathcal{A} = (\mathcal{A}_f, \mathcal{A}_g)$ il existe une fonction négligeable $\nu(\lambda)$ telle que :

$$\mathsf{Adv}^{\mathsf{R-IND}}_{\mathcal{A}}(\lambda) = \left| 2 \cdot \Pr\left[\mathsf{Exp}^{\mathsf{R-IND}}_{A} = 1\right] - 1 \right| \leq \nu(\lambda).$$

7.4 Un protocole basé sur le chiffrement cherchable

Le premier protocole proposé utilise comme brique de base le chiffrement cherchable. Ce protocole répond aux trois premières limitations identifiées à la section précédente.

— Les règles de détection sont chiffrées de manière indépendante des clés de déchiffrement : celles-ci peuvent être utilisées pour l'ensemble des connexions sur le réseau.
— Le moteur d'analyse n'a pas accès aux règles de détection en clair, respectant ainsi les intérêts économiques des éditeurs de sécurité.
— Le trafic n'est pas dupliqué : dans le cas d'un destinataire honnête, celui-ci a la garantie que le trafic reçu est celui qui a été inspecté. De plus, la validité de la découpe peut se faire unité par unité, évitant à R de déchiffrer intégralement un trafic malveillant.

De plus, la confidentialité du trafic vis-à-vis du moteur d'analyse est renforcée par rapport à BlindBox. En effet, n'ayant pas les mots-clés à chercher en clair, le résultat des tests effectués par le moteur d'analyse ne lui apprend que la présence ou l'absence d'attaques, et non la présence ou l'absence de mots-clés.

7.4.1 Chiffrement cherchable déchiffrable

Le concept de chiffrement cherchable déchiffrable a été proposé par Fuhr et Paillier [FP07]. il s'agit une extension du chiffrement cherchable à clé publique proposé par [BCOP04], avec la propriété supplémentaire que les clés de dérivation de trappe et de déchiffrement son complètement indépendantes l'une de l'autre. Ce schéma s'applique parfaitement dans notre contexte, car il permet à l'éditeur de sécurité de générer des trappes tandis que le trafic est déchiffré par le destinataire, sans que les trappes ne dépendent de la clé secrète du destinataire.

Un schéma de chiffrement cherchable déchiffrable DSE (de l'anglais *decryptable searchable encryption*) est composé des cinq algorithmes suivants.

— KeyGen est l'algorithme de génération des clés. Il prend en entrée le paramètre de sécurité λ et génère une clé de dérivation de trappe tk, une clé publique pk et une clé privée sk.
— Encrypt est l'algorithme de chiffrement qui prend en entrée un message M et la clé publique pk. Il retourne un chiffré c.
— TrapGen est l'algorithme de génération de trappe. Il prend en entrée un mot-clé w et la clé de génération de trappe tk et calcule une trappe $T(w)$ qui permet la recherche du mot-clé w.
— Test est l'algorithme de test. Il prend en entrée un chiffré c et une trappe $T(w)$ pour un mot w et retourne 1 si c chiffre le mot w et 0 sinon.
— Decrypt est la procédure de déchiffrement. Il prend en entrée un chiffré c et la clé privée sk, et retourne un message M.

Une instanciation d'un tel protocole est donné dans [FP07] et se base sur le chiffrement ElGamal et les couplages bilinéaires. C'est cette construction, modifiée afin que S n'accède pas aux mots-clés même en cas de correspondance, que nous utilisons dans l'implantation de notre protocole BlindIDS-DSE, présentée section 7.4.3. En particulier, l'étape de détection ne permet pas, en cas de correspondance, de remonter à l'aléa utilisé pour chiffrer le message, mais seulement à une chaîne publique.

7.4.2 Syntaxe du protocole BlindIDS-DSE

Nous décrivons maintenant notre protocole pour la détection d'intrusion sur du trafic chiffré basé sur le DSE. Soit DSE : (DSE.KeyGen, DSE.Encrypt, DSE.TrapGen, DSE.Test, DSE.Decrypt) un schéma de chiffrement cherchable déchiffrable

1. *Mise en place* (Setup). Afin de lancer le système, l'éditeur de règles RG génère une clé de dérivation de trappe tk avec l'algorithme DSE.KeyGen. Dans le même temps, et avec le même algorithme, chaque destinataire R dans le réseau génère sa clé privée sk_R et sa clé publique pk_R.
2. *Préparation des règles* (RuleGen). Le RG utilise l'algorithme DSE.TrapGen pour générer les trappes $T(r_i)$ pour chaque mot-clé r_i à chercher dans le trafic. L'ensemble des trappes $\mathcal{T} = \{T(r_1), \cdots, T(r_\ell)\}$ est envoyé au moteur d'analyse.
3. *Envoi d'un message* (Encrypt). L'émetteur S chiffre chaque mot $w_j, j \in [1, k]$ du message à envoyer avec la procédure DSE.Encrypt sous la clé publique pk_R du destinataire R. Il obtient l'ensemble de messages chiffrés suivants $\mathcal{C} = \{c_1, \ldots, c_k\}$ qu'il envoie à R, en passant par le moteur d'analyse.
4. *Détection d'intrusion* (Detect). Grâce à l'ensemble de trappes $\mathcal{T} = \{T(t_1), \ldots, T(r_\ell)\}$ et à celui de chiffrés $\mathcal{C} = \{c_1, \ldots, c_k\}$ le moteur d'analyse peut vérifier si un mot chiffré correspond à une attaque ou non à l'aide de l'algorithme DSE.Test. Si une correspondance est trouvée, le moteur d'analyse bloque le message ou envoie une alerte au destinataire, selon son mode de fonctionnement. Si aucune correspondance n'est trouvée, le moteur d'analyse transmet le message au destinataire.
5. *Réception* (Decrypt). Le destinataire retrouve le message en déchiffrant chaque mot avec sa clé secrète sk_R.

Tout comme BlindBox, ce protocole repose sur une découpe du trafic en unités pouvant être analysées. Nous privilégions l'approche de découpe par délimiteurs pour deux raisons. Tout d'abord, son surcoût est moindre par rapport à l'approche par taille fixe. De plus, elle a l'avantage de ne pas introduire de redondance, dont l'attaquant pourrait tirer partie pour dissimuler son attaque.

Au moment du déchiffrement, le destinataire peut vérifier au fur et à mesure la validité des unités envoyées par S, et une alerte peut être générée avant le déchiffrement complet en

cas de non-respect du protocole par l'émetteur. Cela répond à la troisième limitation identifiée section 7.2.3, évitant au destinataire de déchiffrer en totalité un trafic malveillant.

Enfin, il y a la possibilité d'ajouter à chaque chiffré la position de l'unité dans le trafic et à chaque trappe les positions auxquelles le mot-clé considéré doit être recherché pour être considéré comme suspect. Ainsi, notre protocole a les mêmes fonctionnalités que le protocole BlindBox II, permettant notamment la détection de mot-clé à un emplacement donné par la règle.

7.4.3 Instanciation

Dans cette section, nous présentons le schéma tel qu'il a été implémenté. Cette construction suit la proposition de DSE basé sur ElGamal faite par [FP07], en la modifiant la procédure de test d'égalité afin que le moteur d'analyse n'ait pas accès au trafic même en cas de correspondance.

Soit $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, q)$ un environnement bilinéaire tel que défini à la section 3.1.2. Nous notons $\mathtt{w}$ la longueur maximale des mots clés. Soit $F : \{0,1\}^{\mathtt{w}} \to \mathbb{G}_2, G : \mathbb{G}_1 \to \{0,1\}^{\mathtt{w}} \times \mathbb{Z}_q$ et $H : \mathbb{G}_T \to \mathbb{Z}_q$ trois fonctions de hachage.

— *Mise en place* (Setup). La mise en place du système et le calcul des clés se fait en deux étapes indépendantes.
  — RG exécute $\mathsf{DSE.KeyGen}(1^\lambda)$ et conserve uniquement la clé de dérivation de trappe $\mathsf{tk} = x' \leftarrow \mathbb{Z}_q$. Il génère également une chaîne S aléatoire de longueur $\mathtt{w}$. Il publie $\mathsf{pk_{RG}} = g_1^{x'}$ et S.
  — Le destinataire R exécute également $\mathsf{DSE.KeyGen}(1^\lambda)$ pour obtenir sa clé privée $\mathsf{sk}_R = x \leftarrow \mathbb{Z}_q$. Il publie sa clé publique associée $\widetilde{\mathsf{pk_R}} = g_1^x$.
  — La clé publique pour un destinataire R est la paire $\mathsf{pk_R} = (\mathsf{pk_{RG}}, \widetilde{\mathsf{pk_R}})$.
— *Préparation des règles* (RuleGen). Pour chaque mot-clé w_i à chercher dans le trafic, RG utilise la procédure $\mathsf{DSE.TrapGen}(w_i, \mathsf{tk})$. Il obtient $T_i = F(w_i)^{x'}$. L'éditeur envoie l'ensemble des trappes $\mathcal{T} = \{T_1, ..T_l\}$ au moteur d'analyse MB.
— *Envoi d'un message* (Encrypt).
  — S découpe le trafic en se basant sur les délimiteurs et obtient les unités $t_1, \ldots, t_n$.
  — Pour chaque unité t_i ainsi obtenue, S tire r_i aléatoirement dans $\mathbb{Z}_q$ et exécute la procédure $\mathsf{DSE.Encrypt}(t_i, \mathsf{pk_R})$. Il obtient successivement : $c_{1,i} = g_1^{r_i}$, $(s_1, s_2)_i = G(\widetilde{\mathsf{pk_R}}^{r_i})$, $c_{2,i} = s_{1,i} \oplus t_i$, $c_{3,i} = g_1^{s_{2,i}}$, $u_i = e(\mathsf{pk_{RG}}^{s_{2,i}}, F(t_i))$ et $c_{4,i} = H(u_i) + S \mod q$. Le chiffré d'une unité t_i est ainsi le quadruplet $c_i = (c_{1,i}, c_{2,i}, c_{3,i}, c_{4,i})$.
  — S envoie chaque unité chiffrée à R.
— *Détection d'intrusions* (Detect). À réception d'un message, le moteur d'analyse exécute la procédure $\mathsf{DSE.Test}(c_i, T_j)$ pour rechercher des correspondances. Pour cela, il exécute les étapes suivantes.
  — Le moteur d'analyse retrouve la valeur $u_i = e(c_{3,i}, T_j)$, puis $S' = c_{4,i} - H(u_i) \mod q$.
  — Si $S' \neq S$, il retourne 0 (cela implique que les valeurs r_j et t_i sont différentes).
  — Sinon, il retourne 1.

  Si le moteur d'analyse retourne 1, cela signifie qu'une correspondance a été trouvée et le moteur génère une alerte, ou peut décider d'interrompre l'échange. Dans le cas contraire, les chiffrés sont transmis à leur destinataire R.
— *Réception* (Decrypt). À réception des chiffrés, R exécute pour chaque c_i la procédure $\mathsf{DSE.Decrypt}(c_i, \mathsf{sk_R})$ composée des étapes suivantes.
  — R calcule $s_i = c_{1,i}^x$ puis en dérive $(s_{1,i}, s_{2_i})$ à l'aide de la fonction G. Enfin, il calcule la valeur $t_i = c_{2,i} \oplus s_{1,i}$.
  — Si $c_{3,i} \neq g_1^{s_{2,i}}$, la procédure retourne $\perp$.
  — Sinon, il calcule $u_i = e(\mathsf{pk_{RG}}^{s_{2,i}}, F(t_i))$ et $S' = c_{4,i} - H(u_i) \mod q$.
  — Si $S' \neq S$, il retourne $\perp$.
  — Enfin, il retourne t_i.

7.4.4 Preuves de sécurité

Dans cette section, nous donnons les théorèmes de sécurité de BlindIDS-DSE relatifs au modèle décrit section 7.3, ainsi que les preuves correspondantes.

Détection

Théorème 7.1. *Notre schéma BlindIDS-DSE détecte correctement les intrusions sous réserve qu'il n'y ait pas de collisions dans la fonction de génération de trappes.*

Démonstration. Supposons que notre schéma ne détecte pas correctement les intrusions. Il existe donc un mot-clé w^* tel que :

1. le RG a envoyé à MB la trappe $T^* = F(w^*)^{x'}$ liée à w^* ;
2. l'émetteur S a produit un chiffré valide pour un trafic t, $c^* = (c_1^*, c_2^*, c_3^*, c_4^*)$;
3. la sortie de Detect est 0 ;
4. le déchiffrement de c^* donne bien $t = w^*$.

En particulier, le troisième point implique que :

$$S \neq c_4^* - H(e(c_3^*, T^*)) \mod q.$$

Or, lors de la procédure de déchiffrement, le destinataire doit obtenir w^*, donc :

$$\begin{aligned} w^* &= c_2^* \oplus s_1^*, \\ S &= c_4^* - H(e(\mathsf{pk}_{\mathsf{RG}}^{s_2^*}, F(w^*))) \mod q. \end{aligned}$$

Ainsi, $e(c_3^*, T^*) \neq e(\mathsf{pk}_{\mathsf{RG}}^{s_2^*}, F(w^*))$, ce qui implique :

$$\begin{aligned} e(g_1^{s_2^*}, F(w^*)^{x'}) &\neq e(g_1^{x' \cdot s_2^*}, F(w^*)), \\ F(w^*) &\neq F(w^*). \end{aligned}$$

Ce qui est absurde.

Nous évaluons enfin la probabilité d'un faux positif, c'est-à-dire qu'une unité de trafic t légitime corresponde à une trappe issue par le RG pour un mot-clé w avec $w \neq t$. En particulier, cela signifie que :

$$\begin{aligned} e(\mathsf{pk}_{\mathsf{SE}}^{s_2}, F(t)) &= e(c_3, F(w)^{x'}), \\ e(g_1^{x' \cdot s_2}, F(t)) &= e(g_1^{s_2}, F(w)^{x'}), \\ e(g_1, F(t))^{x' \cdot s_2} &= e(g_1, F(w))^{s_2 \cdot x'}. \end{aligned}$$

Ceci implique que $F(w) = F(t)$, et donc qu'il y a une collision dans la fonction de hachage F, ce qui arrive avec une probabilité négligeable. □

Indistinguabilité du trafic

Nous prouvons l'insdistinguabilité du trafic sous deux hypothèses : CDH et GDDHE, introduite par [BBG05]. Nous donnons ici une version informelle de GDDHE et renvoyons le lecteur à [BBG05] pour plus de détails.

Hypothèse 7.2 (Diffie-Hellman Calculatoire CDH)**.** *Soit $g, g^a, g^b \in G$, le problème consiste à calculer g^{ab}.*

Hypothèse 7.3 ($((P, Q, f) - \mathsf{GDDHE})$)**.** *Soit s, n des entiers positifs et $P, Q \in \mathbb{F}_p[X_1, \ldots, X_n]^s$ deux s-uplets de polynômes à n variables sur $\mathbb{F}_p$. Soit $f \in \mathbb{F}_p[X_1, \ldots, X_n]$ un polynôme qui soit linéairement indépendant de P et Q. Soit $H(x_1, \ldots, x_n) = (g^{P(x_1,\ldots,x_n)}, \tilde{g}^{Q(x_1,\ldots,x_n)}) \in \mathbb{G}_1^s \times \mathbb{G}_2^s$ et $T = g_T^{f(x_1,\ldots,x_n)}$, calculer f.*

Théorème 7.4. *Notre schéma BlindIDS-DSE est trafic-indistinguable sous les hypothèses* CDH *et* GDDHE *dans le modèle de l'oracle aléatoire.*

Démonstration. Nous prouvons le théorème 7.4 avec un argument hybride. L'objectif est de construire un simulateur $\mathcal{S}$ pour la procédure de chiffrement contre lequel un adversaire $\mathcal{A}$ contre l'expérience $\mathsf{T}-\mathsf{IND}$ a un avantage d'exactement 1/2. Pour cela, il faut que la sortie de $\mathcal{S}$ soit parfaitement indistinguable d'un aléa.
Jeu G_0. Ce jeu est le jeu original : dans celui-ci $\mathcal{S}$ suit la procédure de chiffrement Encrypt. L'adversaire $\mathcal{A}$ choisit T_0 et T_1 tels que $\mathsf{Detect}(\mathsf{Encrypt}(T_0)) = \mathsf{Detect}(\mathsf{Encrypt}(T_1))$ et le challenger chiffre l'un des deux. La vue de l'adversaire est le trafic chiffré suivant, où b est le bit choisit par le challenger.

$$\begin{aligned} c_{1,b} &= g_1^{r_b}; \\ c_{2,b} &= s_{1,b} \oplus T_b; \\ c_{3,b} &= g_1^{s_{2,b}}; \\ c_{4,b} &= H(u_b) + S \mod q. \end{aligned}$$

Durant la suite de jeu, nous notons S_i la probabilité que le bit b' produit par $\mathcal{A}$ soit b dans le jeu G_i. Nous avons :

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{T-IND}}(\lambda) = \left| 2 \cdot \Pr\left[\mathsf{Exp}_{\mathcal{A}}^{\mathsf{T-IND}} = 1\right] - 1 \right| = |2 \cdot S_0 - 1| .$$

Notons que c_1 ne dépend pas de T_b est totalement aléatoire car r_b est un élément aléatoire dans $\mathbb{Z}_q$.
Jeu G_1. La deuxième composante c_2 est également aléatoire et uniforme dans $\mathbb{Z}_q$ dans le modèle de l'oracle aléatoire. En effet, s_1 est la sortie de la fonction G modélisée ici par un oracle aléatoire. Néanmoins, l'adversaire peut retrouver la valeur de s_1 s'il arrive à calculer $g_1^{x \cdot r_b}$ à l'aide de $c_1 = g_1^{r_b}$ et $\widetilde{\mathsf{pk}_\mathsf{R}} = g_1^x$. C'est exactement le jeu de l'hypothèse CDH. Soit $\mathcal{B}$ un adversaire contre CDH. Néanmoins, un adversaire contre CDH ne peut pas utiliser directement $\mathcal{A}$ pour réussir : comme nous utilisons des couplages asymétriques, il ne peut pas vérifier laquelle des requêtes de $\mathcal{A}$ à l'oracle aléatoire est effectivement la réponse au défi CDH. Soit $\mathcal{B}$ un adversaire contre CDH, alors, en notant q_G le nombre de requêtes de $\mathcal{A}$ à l'oracle aléatoire, pour tout $\mathcal{B}$ nous avons :

$$|\mathtt{Pr}(S_0) - \mathtt{Pr}(S_1)| \leq \frac{1}{q_G} \mathsf{Adv}_{\mathcal{B}}^{\mathsf{CDH}}(\lambda).$$

Jeu G_2**.** Nous faisons le même raisonnement sur c_3, et pour tout adversaire $\mathcal{B}$ contre CDH, nous avons :

$$|\mathtt{Pr}(S_1) - \mathtt{Pr}(S_2)| \leq \frac{1}{q_G} \mathsf{Adv}_{\mathcal{B}}^{\mathsf{CDH}}(\lambda).$$

Jeu G_3**.** De même, la dernière composante c_4 est aléatoire sauf si $\mathcal{A}$ trouve la valeur de u. $\mathcal{A}$ a accès à des éléments de la forme $g_1^{P(r_b, s_2, x', x)}$ en combinant $c_1, c_2, \mathsf{pk}_\mathsf{RG}$ et $\widetilde{\mathsf{pk}_\mathsf{R}}$. Le polynôme P ainsi obtenu a des monômes de degré maximal 1, car les exposants r_1, s_2, x and x' sont secrets. $\mathcal{A}$ a aussi accès à des éléments de type $g_2^{Q(\alpha_i, x')}$, en notant α_i les logarithmes discrets (inconnus) des $F(t_i)$ que $\mathcal{A}$ peut calculer pour les unités de son choix. Les monômes de Q sont de degré au plus deux, correspondant aux trappes de la forme $F(w_i)^{x'} = g_2^{\alpha_i \cdot x'}$ fournies par le RG. Or $\mathcal{A}$ doit calculer un élément de $\mathbb{G}_t$ de la forme $g_t^{s_2 \cdot x' \cdot \alpha_b}$. L'exposant est un monôme de degré 3, donc linéairement indépendant de P et Q. Soit $\mathcal{C}$ un adversaire contre l'hypothèse GDDHE, de même que précédemment, il ne peut vérifier laquelle des requêtes de $\mathcal{A}$ correspond effectivement à un succès pour GDDHE. Ainsi, pour tout $\mathcal{C}$, et en notant q_H le nombre de requêtes de $\mathcal{A}$ à l'oracle aléatoire, nous avons :

$$|S_2 - S_3| \leq \frac{1}{q_H}\mathsf{Adv}_{\mathcal{C}}^{\mathsf{GDDHE}}(\lambda).$$

Une fois que u est remplacé par une valeur aléatoire, la sortie de $\mathcal{S}$ est parfaitement indistinguable de l'uniforme et $S_3 = 1/2$.

Ainsi,

$$|2 \cdot S_0 - 1| \leq \frac{2}{q_G}\mathsf{Adv}_{\mathcal{B}}^{\mathsf{CDH}}(\lambda) + \frac{1}{q_H}\mathsf{Adv}_{\mathcal{C}}^{\mathsf{GDDHE}}(\lambda),$$

et l'avantage de l'adversaire est négligeable. □

Indistinguabilité des règles

Théorème 7.5. *Notre schéma BlindIDS-DSE est règle-indistinguable pour tout ensemble de règle avec une haute min-entropie dans le modèle de l'oracle aléatoire.*

Démonstration. Nous considérons un adversaire en deux parties $\mathcal{A} = (\mathcal{A}_f, \mathcal{A}_g)$. La première partie de l'adversaire $\mathcal{A}_f$ génère deux mots clés w_0 et w_1 et exécute $\mathsf{RuleGen}$ sur w_b pour un bit secret $b \in \{0, 1\}$. Cette procédure renvoie $T_b = F(w_b)^{x'}$. La trappe résultante T_b est donnée à la seconde partie de l'adversaire, $\mathcal{A}_g$. Rappelons que les deux parties de l'adversaire $\mathcal{A}_f$ et $\mathcal{A}_g$ ne peuvent pas communiquer entre elles.

Dans le modèle de l'oracle aléatoire, T_b est entièrement déterminée par w_b. $\mathcal{A}_g$ peut créer autant de trafic chiffré sur des mots de son choix w, et faire appel à l'oracle aléatoire pour obtenir $F(w)$. Comme les règles viennent d'une distribution ayant une haute entropie-min, cela n'arrive qu'avec une probabilité supérieure à $2^{-\mu(\lambda)}$, avec $\mu(\lambda) \in \omega(\log \lambda)$, ce qui est négligeable. □

7.5 Implantation et performances

7.5.1 Plate-forme de test

Ce protocole a été implanté en Java 8, en utilisant l'API pour les couplages fournie par la bibliothèque [MIT13], écrite en C++. Nous avons utilisé la courbe fournie par défaut par [MIT13], qui est une courbe Barreto-Naehrig 254 bits. Le couplage utilisé est Optimal Ate, et la fonction de hachage sha 256. Les nombres aléatoires ont été générés avec la classe Java SecureRandom. Les tests ont été exécutés sur un Intel(R) Xeon(R) avec un E5-1620 CPU 3.70GHz à 4 cœurs sous Linux 64 bits.

7.5.2 Performances

Nous décrivons dans cette section les résultats synthétisés figure 7.7. Pour faciliter la comparaison, nous reprenons les scénarios de test de BlindBox II, qui est le seul protocole BlindBox implanté dans [SLPR15] : des unités de 128 bits et des paquets réseau de 1500 octets. Comme une règle contient 3 à 4 mots-clés, 3k règles correspondent à 10k mots-clés.

Performances en temps

La principale conséquence de l'utilisation d'un schéma à clé publique est la réduction du temps de mise en place de la connexion, déplaçant le surcoût induit par la détectabilité d'attaque du client au moteur d'analyse. En utilisant Blind-IDS DSE, un utilisateur met en moyenne 2.3 secondes pour chiffrer, charger et envoyer au moteur d'analyse une page web de type CNN, alors que ce temps est de 97 secondes avec BlindBox. D'autres temps de chargement pour des pages populaires sont disponibles figure 7.8. Ce gain de temps se fait au prix d'une détection plus lente au niveau du moteur d'analyse. Néanmoins, le temps total requis avec notre protocole pour chiffrer un paquet typique de 1500 octets et d'effectuer dessus une détection d'attaque est

Rôle	Description	Inspection SSL	BlindBox II	Notre solution
Émetteur/ Destinataire	Mise en place (1 mot-clé)	73ms	588ms	73ms
	Mise en place (3K règles)	73ms	97s	73ms
	Chiffrement (128 bits)	13ns	69ns	729μs
	Chiffrement (1500 bytes)	3μs	90μs	27ms
Moteur d'analyse *(Temps de détection)*	1 règle, 1 unité (128 bits)	Pas applicable	20ns	691μs
	1 règle, 1 paquet (1500 octets)	Pas applicable	5μs	41.3ms
	3K règles, 1 unité (128 bits)	Pas applicable	137ns	1.1s
	3K règles, 1 paquet (1500 octets)	Pas applicable	33μs	84s
Moteur d'analyse *(Utilisation RAM)*	1 règle, 1 connexion	Pas applicable	1.75Mo	0.2ko
	3K règles, 1 connexion	Pas applicable	5.71Go	0.6Mo
	1 règle, 100 connexions	Pas applicable	175Mo	0.2ko
	3K règles, 100 connexions	Pas applicable	571Go	0.6Mo

Figure 7.7 – Performances de notre solution BlindIDS-DSE pour l'établissement de la connexion et la détection, en comparaison avec une connexion SSL et la solution BlindBox II.

Site internet	Taille	HTTPS	BlindBox II	Notre solution
CNN	131KB	0.073	97.008	2.373
Facebook	74KB	0.073	97.004	1.073
Twitter	284KB	0.073	97.017	5.073
BBC	196KB	0.073	97.011	3.573

Figure 7.8 – Temps de chargement (en secondes) de divers sites populaires

inférieur au seul coût de mise en place de la connexion par BlindBox. De plus, les calculs lourds sont maintenant faits au niveau du moteur d'analyse, alors que le protocole BlindBox requiert un grand nombre de calcul de circuits brouillés par les utilisateurs. Or, le moteur d'analyse peut être un large serveur avec des capacités de calcul et de parallélisation bien supérieures à celle d'un ordinateur personnel.

Performances en mémoire

Enfin, nous améliorons considérablement les performances mémoire de BlindBox. En effet, les trappes utilisées pour détecter des attaques ne dépendent pas de la clé secrète du destinataire. L'espace mémoire nécessaire pour les stocker est donc linéaire en le nombre de règles utilisées. En revanche, BlindBox doit stocker un circuit de Yao par mot-clé et par destinataire, ce qui augmente considérablement l'espace de stockage requis en cas de connexions parallèles. Ce gain est d'autant plus significatif que nos trappes consistent en un élément de $\mathbb{G}_2$, soit 508 bits alors que les circuits embrouillés utilisés par BlindBox font 599 kilo-octets. Ainsi, pour 100 connexions parallèles et 3000 règles de détection, l'espace requis par BlindBox est évalué à 512Go, alors que notre solution ne nécessite que 580ko.

7.6 Un protocole basé sur le chiffrement homomorphe

Dans cette section, nous nous intéressons surtout à la quatrième limitation identifiée de BlindBox (bien que notre protocole réponde également aux autres limitations). Il s'agit d'effectuer les mêmes fonctionnalités qu'un IDS sans avoir à déchiffrer le trafic. De plus, avec notre protocole, nous élargissons le spectre des attaques détectées : en effet, le protocole BlindBox III n'effectue la détection d'attaques complète qu'en cas de correspondance de motifs suspects. Ce protocole a avant un tout un intérêt théorique, en réussissant à répondre à toutes les limitations identifiées sur le protocole BlindBox. Il nous permet aussi d'étudier la possibilité d'évaluer des expressions rationnelles sur un message chiffré.

7.6.1 Évaluation d'expression rationnelles sur des données chiffrées

L'évaluation d'expression rationnelles sur les données chiffrées est une opération assez délicate. En effet, cette opération est susceptible de révéler une certaine quantité d'informations sur les

messages clairs. Ainsi, comme dans le cas du tri étudié chapitre 4, il est vital que le temps d'exécution ne fournisse aucune information sur le message évalué. Comme toute expression rationnelle peut être évaluée avec un automate fini 2, il suffit de pouvoir évaluer ces dernier sur du trafic chiffré.

Soit $\mathcal{A} = (\Sigma, Q, T, q_0, F)$ un automate fini à évaluer. Notre but est de savoir si une chaîne γ composée des caractères $\gamma_1 \parallel \cdots \parallel \gamma_n$ est reconnue par $\mathcal{A}$. L'alphabet Σ peut être considéré comme étant $\{0,1\}^*$ grâce à l'encodage ASCII des caractères. Ainsi, pour simplifier les formules, nous considérons que γ est une suite de bits.

Nous utilisons les instructions conditionnelles pour évaluer les états successifs dans lesquels se trouve l'automate pendant l'évaluation de γ. Nous cherchons à évaluer si l'état résultant de la fonction suivante appartient à un état final F :

$$T(\gamma_n, T(\gamma_{n-1}, \dots T(\gamma_2, T(\gamma_1, q_0)).$$

Enfin, la valeur (chiffrée) de $T(\gamma_i, q_j)$ est donnée en évaluant de manière homomorphe

$$(\gamma_i == 0) \times T(0, q_j) + (\gamma_i == 1) \times T(1, q_j).$$

Ceci peut se faire avec un bruit qui croît linéairement en la taille de γ [GINX16, CGGI16].

7.6.2 Le protocole BlindIDS-FHE

Ce protocole utilise un schéma de chiffrement totalement homomorphe FHE : ($\mathsf{FHE.KeyGen}$, $\mathsf{FHE.Encrypt}$, $\mathsf{FHE.Decrypt}$, $\mathsf{FHE.Eval}$), ainsi qu'un schéma de chiffrement symétrique Sym : ($\mathsf{Sym.KeyGen}$, $\mathsf{Sym.Encrypt}$, $\mathsf{Sym.Decrypt}$). Afin d'éviter d'envoyer des quantités de données trop importantes sur le réseau, l'émetteur chiffre ses données en utilisant Sym et une clé secrète K et ne chiffre que cette dernière avec le protocole FHE.

- *Mise en place* (Setup). L'éditeur de règles exécute $\mathsf{FHE.KeyGen}$ et obtient sa paire clé privée, clé publique $(\mathsf{sk_{RG}}, \mathsf{pk_{RG}})$.
- *Préparation des règles.* ($\mathsf{RuleGen}$). L'éditeur de règles transforme les expressions régulières de chaque règle i en un algorithme $\mathcal{A}_i$ évaluable par $\mathsf{FHE.Valid}$. Notons que les entrées de $\mathcal{A}_i$ qui dépendent de la règles à évaluer (mots-clés, caractères des expressions rationnelles) sont chiffrées en utilisant $\mathsf{FHE.Encrypt}$ et la clé publique du RG. Chaque $\mathcal{A}_i$ (avec ses entrées chiffrées) est ensuite envoyé au moteur d'analyse.
- *Envoi d'un message* ($\mathsf{Encrypt}$). Le moteur d'analyse choisit une valeur x_0 et l'envoie à S. S et R choisissent une clé de session K qui sera utilisée par le schéma de chiffrement symétrique Sym. Pour envoyer un message M, S calcule les valeurs suivantes :

$$T = \mathsf{Sym.Encrypt}((x_0 \parallel M), K),$$
$$S = (\mathsf{FHE.Encrypt}(K, \mathsf{pk_{RG}}).$$

  Le chiffré (T, S) est envoyé à R via le moteur d'analyse.
- *Détection d'intrusion* (Detect). Le moteur d'analyse effectue les étapes suivantes.
  - Calcul de $T' = \mathsf{FHE.Encrypt}(T, \mathsf{pk_{RG}})$
  - Utilisation de la procédure $\mathsf{FHE.FHE.Eval}$ pour évaluer la fonction $\mathsf{Sym.Decrypt}$ et donc retirer la couche de chiffrement interne. On obtient :

$$\begin{aligned} T'' &= \mathsf{FHE.Eval}(T', S, \mathsf{Sym.Decrypt}) \\ &= \mathsf{FHE.Enc}(\mathsf{Sym.Decrypt}(\mathsf{Sym.Encrypt}((x_0 \parallel M), K), K), \mathsf{pk_{RG}}) \\ &= \mathsf{FHE.Enc}((x_0 \parallel M), \mathsf{pk_{RG}}). \end{aligned}$$

  - Décomposer T'' de manière homomorphe pour obtenir

$$T''_1 = \mathsf{FHE.Enc}(x_0, \mathsf{pk_{RG}}) \text{ et } T''_2 = \mathsf{FHE.Enc}(M, \mathsf{pk_{RG}}).$$

— Pour chaque $\mathcal{A}_i$ à évaluer sur le trafic M, le moteur d'analyse utilise la procédure FHE.Eval pour obtenir

$$\begin{aligned} B &= \mathsf{FHE.Eval}(T_2'', \mathcal{A}_i) \\ &= \mathsf{FHE.Enc}(\mathcal{A}_i(M), \mathsf{pk_{RG}}). \end{aligned}$$

— Le moteur d'analyse envoie $(T_1'', B) = (\mathsf{FHE.Enc}(x_0, \mathsf{pk_{RG}}), \mathsf{FHE.Enc}(\mathcal{A}_i(M), \mathsf{pk_{RG}})$ avec x_0 à l'éditeur de règles.
— L'éditeur de règles déchiffre T_1'' et vérifie que le résultat est égal à x_0, et si c'est le cas, il déchiffre B, obtient un bit b et le renvoie au moteur d'analyse.
— Si $b = 1$, le moteur d'analyse génère une alerte et sinon, il transfère le trafic T au destinataire.

— *Réception* (Decrypt). Le destinataire R utilise la clé de session K pour déchiffrer T.

À première vue, ce protocole souffre de deux inconvénients. Le premier est l'étape interactive entre le moteur d'analyse et l'éditeur de règles au moment de la détection d'attaques. Il se trouve que cette architecture est compatible avec ce qui se fait déjà dans le domaine de la détection d'intrusions. Par exemple, dans le système commercialisé par BlueCoat [Blu16], un échange se produit entre le moteur d'analyse et l'éditeur de règle à chaque inspection de trafic.

Le deuxième inconvénient est que ce protocole fait l'hypothèse que l'éditeur de règles est honnête. En effet, la clé de session étant chiffrée sous la clé publique de l'éditeur de règles, celui-ci peut surveiller et déchiffrer tout le trafic. Cela peut sembler étrange d'accuser un éditeur de sécurité, censé protéger ses utilisateurs, de malveillance à l'égard de ceux-ci, mais de récents scandales impliquant les éditeurs de sécurité dans la surveillance de masse nécessite de se prémunir aussi contre ce genre d'attaques [Fro16]. On peut pour cela avoir recours à un proxy de rechiffrement compatible avec le chiffrement homomorphe [Gen09a]. Cela permet de retrouver l'hypothèse classique de confidentialité sous réserve de non collusion entre le moteur d'analyse et l'éditeur de règles.

Nous décrivons d'abord une instanciation du proxy de rechiffrement compatible avec le chiffrement totalement homomorphe puis la transformation du protocole BlindIDS-FHE.

Variante en utilisant un proxy de rechiffrement.

Nous rappelons ici une construction générique de proxy de rechiffrement à partir d'un schéma totalement homomorphe proposée par Gentry dans sa thèse [Gen09a]. À partir d'un schéma de chiffrement totalement homomorphe FHE : (FHE.PPGen, FHE.KeyGen, FHE.Encrypt, FHE.Decrypt, FHE.Eval), nous construisons un proxy de rechiffrement de A vers B constitué des procédures $\mathsf{PRE}_{A\to B}$: (PRE.PPGen, PRE.KeyGen, PRE.ReKeyGen, PRE.Encrypt, PRE.ReEnc, PRE.Decrypt).

— PRE.PPGen. Les paramètres du proxy de rechiffrement sont générés à partir de la procédure FHE.PPGen.
— PRE.KeyGen. Les deux parties A et B génèrent leur paire de clé à l'aide de FHE.KeyGen. Ils obtiennent respectivement $(\mathsf{pk}_A, \mathsf{sk}_A)$ et $(\mathsf{pk}_B, \mathsf{sk}_B)$.
— PRE.ReKeyGen. A calcule sa clé de rechiffrement $\mathsf{rk}_{A\to B}$ vers B comme la paire $(k_1, k_2) = (pk_B, \mathsf{FHE.Enc}(sk_A, \mathsf{pk}_B))$. Cette clé est transmise au proxy.
— PRE.Encrypt. L'algorithme de chiffrement prend en entrée les paramètres publics pp, la clé publique de A et un message M et produit un chiffré à l'intention de A ainsi : $c_A = \mathsf{FHE.Enc}(M, \mathsf{pk}_A)$. Ce dernier va pouvoir (éventuellement) être rechiffré pour B.
— PRE.ReEnc. À réception d'un chiffré c_A, le proxy le rechiffre pour B de la manière suivante.
  — Il calcule $c' = \mathsf{FHE.Enc}(c_A, \mathsf{pk}_B)$.
  — La clé privée de A et le message M étant maintenant tous les deux chiffrés avec la clé publique de B, il exécute la procédure $\mathsf{FHE.Eval}(\mathsf{FHE.Decrypt}(k_2, c'))$. Elle retourne le message M chiffré avec la clé publique de B : $c_B = \mathsf{FHE.Enc}(M, \mathsf{pk}_B)$
— PRE.Decrypt. La procédure de déchiffrement est FHE.Decrypt.

Avec ce proxy, on peut adapter le protocole BlindIDS-FHE de la manière suivante.

— Comme précédemment, l'éditeur de règles exécute la procédure $\mathsf{FHE.KeyGen}$ et obtient sa paire de clés $(\mathsf{pk}_{\mathsf{RG}}, \mathsf{sk}_{\mathsf{RG}})$. Il chiffre les règles avec sa clé publique $\mathsf{pk}_{\mathsf{RG}}$ et les envoie au moteur d'analyse.
— Le destinataire R calcule une clé de rechiffrement de lui-même vers l'éditeur de règles RG. Cette clé de rechiffrement est la paire $(k_1, k_2) = (\mathsf{pk}_{\mathsf{R}}, \mathsf{FHE.Encrypt}(\mathsf{sk}_{\mathsf{R}}, \mathsf{pk}_{\mathsf{RG}}))$. Il envoie (k_1, k_2) au moteur d'analyse.
— L'émetteur S chiffre les clés de session en utilisant la clé publique du destinataire R.
— À réception d'une clé de session chiffré pour R et du trafic associé, le moteur d'analyse utilise (k_1, k_2) et la rechiffre pour la clé publique de l'éditeur de règle RG.
— Le moteur d'analyse réalise ensuite la détection d'attaque comme précédemment, et reçoit de l'éditeur de règle le bit b valant 0 ou 1 correspondant au résultat de l'analyse.
— Selon la valeur de b, le moteur d'analyse transfère le trafic au destinataire ou génère une alerte.

7.6.3 Arguments de sécurité

Dans cette section, nous donnons les théorèmes de sécurité de BlindIDS-FHE relatifs au modèle décrit section 7.3. Pour chaque théorème, nous donnons l'idée de la preuve correspondante.

Détection

Théorème 7.6. *Notre schéma BlindIDS-FHE détecte correctement les intrusions pour tout schéma FHE correct.*

Démonstration. Pour simplifier, nous donnons la preuve dans le schéma sans proxy de rechiffrement, mais celle-ci s'adapte facilement à la variante avec proxy de rechiffrement.

Dans le protocole FHE, les règles à détecter sont données sous forme de circuits et de variables chiffrées. De même que pour le DSE, supposons qu'un adversaire réussisse dans l'expérience de détection avec un avantage non-négligeable.

Il existe donc une règle $\mathcal{R}^*$ et un message M tel que $\mathcal{R}^*(M) = 1$ et :

1. le RG a envoyé à MB la trappe $\mathcal{A}^*$ liée à $\mathcal{R}^*$;
2. l'émetteur S a produit un chiffré valide c^* :
$$T^* = \mathsf{Sym.Encrypt}((x_0 \parallel M), K),$$
$$S^* = (\mathsf{FHE.Encrypt}(K, \mathsf{pk}_{\mathsf{RG}});$$
3. la sortie de Detect est 0 ;
4. le déchiffrement de c^* donne bien M.

Avec ce chiffré, lors de la procédure de détection, le moteur d'analyse commence par récupérer $T_2^* = \mathsf{FHE.Encrypt}(M, \mathsf{pk}_{\mathsf{RG}})$, puis calcule :

$$\begin{aligned} B &= \mathsf{FHE.Eval}(T_2^*, \mathcal{A}^*) \\ &= \mathsf{FHE.Enc}(\mathcal{A}^*(M), \mathsf{pk}_{\mathsf{RG}}). \end{aligned}$$

Or, nous avons supposé que $\mathcal{R}^*(M) = 1$, donc, si le schéma FHE est correct, l'évaluation homomorphe de $\mathcal{A}^*$ sur un chiffré de M retourne 1. □

Indistinguabilité du trafic

Théorème 7.7. *Notre schéma BlindIDS-FHE est trafic-indistinguable pour tout schéma FHE* IND-CPA.

L'argument principal de la preuve est que les chiffrés BlindIDS-FHE sont *exactement* des chiffrés FHE, qui sont indistinguables.

Indistinguabilité des règles

Théorème 7.8. *Notre schéma BlindIDS-FHE est règle-indistinguable pour tout schéma FHE* IND-CPA *et tout ensemble de règle avec une haute min-entropie.*

De même que précédemment, l'adversaire $\mathcal{A}_g$ n'a moyen de distinguer entre deux règles que s'il crée un chiffré satisfaisant l'une de ces règles, ce qui arrive avec une probabilité négligeable du fait de la haute entropie-min des règles.

7.7 Conclusion

Si le développement du trafic chiffré permet de protéger les données échangées sur un réseau, il empêche également les fournisseurs de services d'effectuer leurs fonctionalités habituelles. Or, parmi ces fonctionnalités, certaines sont vitales pour la sécurité du réseau : c'est le cas des services de détection d'intrusion. Suite aux premiers protocoles proposés dans [SLPR15], nous avons développé deux schémas répondant aux limitations de ceux-ci. Notre premier schéma, BlindIDS-DSE, présente des temps d'exécution semblables à ceux de BlindBox avec des propriétés de sécurité et de confidentialité renforcées. Notre second schéma est avant tout théorique, mais répond de manière complète au problème.

Néanmoins, autant nos protocoles que ceux proposés par [SLPR15] reposent sur l'hypothèse que l'éditeur de sécurité est en mesure de fournir des règles pertinentes pour la détection d'intrusion. Si aujourd'hui la part de trafic clair reste suffisante pour être étudiée avec des méthodes d'apprentissage automatique et ainsi fournir des signatures d'attaques, la part de trafic chiffré augmente très rapidement. La question qui se pose alors est la suivante. Est-il possible d'être en mesure d'apprendre des attaques menées sur du trafic chiffré afin de fournir de nouvelles règles et ainsi s'adapter à l'évolution constante des logiciels malveillants ?

Si cela est probablement encore trop inefficace pour être déployé en pratique, des progrès ont été effectués récemment en terme d'apprentissage automatique et de réseau de neurones sur les données chiffrées. Cela nous semble une piste intéressante pour prolonger ces travaux sur la détection d'intrusions.

Conclusion

Le monde numérique, longtemps vu comme un monde abstrait, n'est en fait qu'une extension du réel. On ne peut pas s'y comporter sans précautions. Personne n'enverrait une lettre à sa banque, par exemple, sans la mettre dans une enveloppe pour en protéger le contenu. D'ailleurs, même pour envoyer de simples photos de vacances, on utiliserait également un moyen de protection. Or, ces précautions élémentaires n'ont pas toujours été prises dans le monde numérique. Les récents scandales sur la surveillance de masse ont brutalement fait (re)prendre conscience que non seulement l'utilisateur réel n'était absolument pas déconnecté de son avatar numérique, mais qu'en plus, la mémoire du monde numérique est sans commune mesure avec la nôtre.

Ce manuscrit s'intéresse aux méthodes de calculs sur les données chiffrées. En effet, la prise de conscience des utilisateurs de l'exploitation de leurs données personnelles s'accompagne d'une demande vers plus de chiffrement. Or celui-ci bloque des fonctionnalités importantes des services internet, pouvant aller jusqu'à mettre en danger les utilisateurs. La cryptographie fait donc face à un nouveau défi : pouvoir exécuter de nombreuses fonctionnalités sur des données chiffrées. Ce manuscrit s'intéresse ainsi aux méthodes de calculs sur les données chiffrées.

Dans cette thèse, nous avons proposé trois contributions à cette question : une amélioration des calculs génériques sur les données chiffrées, la déduplication vérifiable de données chiffrées et la détection d'intrusions sur du trafic chiffré.

En formalisant le problème du nombre de réamorçages nécessaire à la bonne évaluation d'un circuit, nous avons pu proposer à la fois une analyse théorique du problème (en prouvant qu'il est NP-complet) et une solution pratique permettant une nette diminution des appels à cette procédure coûteuse. Par exemple, le nombre de réamorçages pour l'évaluation d'un AES est divisé par trois.

La déduplication vérifiable de données dans le cloud permet à un serveur de s'assurer que deux messages clairs identiques seront effectivement identifiés comme tels une fois chiffrés. Même si notre schéma est plus efficace que ceux reposant sur des solutions génériques, ces travaux mettent en valeur l'énorme fossé qui existe entre les capacités théoriques de la cryptographie, qui viennent souvent au prix d'une efficacité très réduite, et les contraintes de déploiement.

Enfin, notre solution de détection d'intrusions sur du trafic chiffré se décline en deux variantes. La première, basée sur du chiffrement cherchable, améliore nettement les performances de l'état de l'art. Elle apporte aussi une fonctionnalité capitale pour l'acceptation de ce type de protocoles sur le marché : les règles de détection, élément capital pour la promotion de telle ou telle solution, restent privées. La deuxième variante, basée sur le chiffrement homomorphe, permet de détecter exactement les mêmes attaques que sur du trafic clair. Il est à noter toutefois que l'efficacité de ce protocole, étroitement liée à celle du chiffrement totalement homomorphe, reste bien inférieure à celle de notre premier protocole.

Perspectives

Le monde numérique est également un monde aux interactions complexes et avec un marché économique bien défini. Sans contrainte légale, le chiffrement ne sera donc spontanément adopté que s'il se plie à ses exigences. Nos travaux montrent que si la cryptographie a le pouvoir théorique d'offrir une telle flexibilité, l'efficacité n'est aujourd'hui pas toujours celle attendue de

services internet.

Une première piste d'amélioration est, de manière évidente, l'amélioration des schémas de chiffrement totalement homomorphes. Les résultats déjà obtenus par la communauté cryptographique sur ce point sont remarquables, notamment en passant d'une croissance exponentielle du bruit dans les chiffrés à une croissance linéaire, et en réduisant considérablement la complexité de l'opération de réamorçage. Néanmoins, la cryptographie basée sur les réseaux est en pleine expansion, et les récents travaux de cryptanalyse laissent penser qu'il va probablement falloir augmenter les paramètres des schémas homomorphes. Le choix des paramètres est d'ailleurs une question d'une très grande importance qui pour l'instant est traitée au cas par cas, protocole par protocole. Un modèle unifiant les différents schémas totalement homomorphes, sur le même principe que celui que nous avons conçu pour le problème de placement des réamorçages, permettrait une compréhension fine de la sécurité et donc d'augmenter la confiance que nous plaçons dans ces schémas.

De plus, les limitations du chiffrement homomorphe ne sont pas seulement dues à des schémas inefficaces qu'il serait possible d'améliorer et d'optimiser. Malgré le travail d'optimisation des protocoles déjà mené, le chiffrement homomorphe semble résister et continuer à rester terriblement inefficace pour nombre d'applications pratiques. Sur ce point, les problèmes soulevés au chapitre 4 semblent inhérents au calcul homomorphe. La nécessité d'exécuter des algorithmes sur les données de telle manière à ce que le temps d'exécution ne laisse rien transparaître sur celles-ci nous force à travailler avec les complexités pire cas des algorithmes actuels. Or, même Barack Obama est conscient que le monde numérique d'aujourd'hui ne peut pas se contenter du tri à bulles pour fonctionner [Oba08] !

Une piste pour améliorer l'efficacité du chiffrement homomorphe pour des cas pratiques est donc que les algorithmiciens et les cryptologues travaillent ensemble à concevoir des algorithmes dont le temps d'exécution est le plus possible indépendant des données sur lesquelles il sont exécutées. Pour ce type de chiffrement, il en effet est plus intéressant d'avoir des algorithmes de temps d'exécution constant plutôt que des algorithmes extrêmement optimisés pour les cas moyens et qui se révèlent inefficaces dans le pire cas. Pour des opérations basiques, comme la multiplication binaire, ces questions sont déjà très étudiées du côté des circuits électroniques et de nombreuses améliorations peuvent aussi naître d'un dialogue avec la communauté des électroniciens. Pour des opérations plus complexes, comme la fouille de données, il est nécessaire de se rapprocher des spécialistes du domaine pour obtenir des algorithmes adaptés à une évaluation sur des chiffrés.

Enfin, si la conception de schémas spécifiques peut permettre des performances relativement acceptables en temps et en mémoire, ou en tous cas comparativement à des solutions génériques, celles-ci posent un problème de compatibilité. En effet, chaque schéma ne permet qu'un seul traitement des données. Cela signifie donc que si l'on veut obtenir plusieurs fonctionnalités sur les chiffrés, il faut les chiffrer plusieurs fois, en utilisant à chaque fois le protocole approprié, ce qui diminue largement l'intérêt de ces solutions. Travailler sur des techniques rendant ces protocoles compatibles entre eux, sur la forme des protocoles *encryption-switching*, permettant de passer d'un schéma additivement homomorphe à un schéma multiplicativement homomorphe en calcul multi-parties [CPP16], pourrait être un moyen d'adresser ce problème.

Publications et brevets

Publications internationales

[CDK+17] *BlindIDS : Market-Compliant and Privacy-Friendly Intrusion Detection System over Encrypted Traffic,* (**AsiaCCS 2017**).
S. Canard, A. Diop, N. Kheir, M. Paindavoine et M. Sabt.

[CLP16] *Verifiable Message Locked Encryption,* (**CANS 2016**).
S. Canard, F. Laguillaumie et M. Paindavoine.

[PV15] *Minimizing the Number of Bootstrappings in Fully Homomorphic Encryption,* (**SAC 2015**).
M. Paindavoine et B. Vialla.

[LPM+14] *Practical Validation of Several Fault Attacks against the Miller Algorithm,* (**FDTC 2014**).
R. Lashermes, M. Paindavoine, N. El Mrabet, Jacques J.A. Fournier, et L. Goubin.

La publication [LPM+14] correspond à des résultats antérieurs à ceux développés dans ce mémoire.

Brevets

[CDKP16b] *Procédé et système de détection d'intrusions sur un réseau.*
S. Canard, A. Diop, N. Kheir et M. Paindavoine.

[CDKP16a] *Procédé et dispositif de détection d'intrusions sur un réseau utilisant un algorithme de chiffrement homomorphe.*
S. Canard, A. Diop, N. Kheir et M. Paindavoine.

Communications nationales

A Verifiable Message Locked Encryption. (**SEC2 2016**)
S. Canard, F. Laguillaumie et M. Paindavoine.
Minimizing the Number of Bootstrappings in Fully Homomorphic Encryption. (**JC2 2015**)
M. Paindavoine et B. Vialla.
Le chiffrement homomorphe, une boîte à outils pour la protection de la vie privée. (**APVP 2014**), S. Canard et M. Paindavoine.

Liste des tableaux

Liste des figures

Liste des algorithmes

Bibliographie

[AB09] Sanjeev Arora and Boaz Barak. *Computational Complexity - A Modern Approach*. Cambridge University Press, 2009.

[ABC+05] Michel Abdalla, Mihir Bellare, Dario Catalano, Eike Kiltz, Tadayoshi Kohno, Tanja Lange, John Malone-Lee, Gregory Neven, Pascal Paillier, and Haixia Shi. Searchable encryption revisited : Consistency properties, relation to anonymous ibe, and extensions. In Victor Shoup, editor, *Advances in Cryptology - CRYPTO 2005 : 25th Annual International Cryptology Conference*, volume 3621 of *Lecture Notes in Computer Science*, pages 205–222. Springer, 2005.

[ABD16] Martin R. Albrecht, Shi Bai, and Léo Ducas. A subfield lattice attack on overstretched NTRU assumptions - cryptanalysis of some FHE and graded encoding schemes. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Part I*, volume 9814 of *Lecture Notes in Computer Science*, pages 153–178. Springer, 2016.

[ABM+13] Martín Abadi, Dan Boneh, Ilya Mironov, Ananth Raghunathan, and Gil Segev. Message-locked encryption for lock-dependent messages. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Part I*, volume 8042 of *Lecture Notes in Computer Science*, pages 374–391. Springer, 2013.

[AFV11] Shweta Agrawal, David Mandell Freeman, and Vinod Vaikuntanathan. Functional encryption for inner product predicates from learning with errors. In Dong Hoon Lee and Xiaoyun Wang, editors, *Advances in Cryptology - ASIACRYPT 2011 - 17th International Conference on the Theory and Application of Cryptology and Information Security*, volume 7073 of *Lecture Notes in Computer Science*, pages 21–40. Springer, 2011.

[ALSZ13] Gilad Asharov, Yehuda Lindell, Thomas Schneider, and Michael Zohner. More efficient oblivious transfer and extensions for faster secure computation. In Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung, editors, *2013 ACM SIGSAC Conference on Computer and Communications Security, CCS'13*, pages 535–548, 2013.

[App16] Apple. Apple previews iOS 10, the biggest iOS release ever, Juin 2016. http://www.apple.com/newsroom/2016/06/apple-previews-ios-10-biggest-ios-release-ever.html.

[Bat68] Kenneth E. Batcher. Sorting networks and their applications. In *American Federation of Information Processing Societies : AFIPS Conference, 1968 Spring Joint Computer Conference*, volume 32 of *AFIPS Conference Proceedings*, pages 307–314. Thomson Book Company, Washington D.C., 1968.

[BBG05] Dan Boneh, Xavier Boyen, and Eu-Jin Goh. Hierarchical identity based encryption with constant size ciphertext. In Ronald Cramer, editor, *Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, volume 3494 of *Lecture Notes in Computer Science*, pages 440–456. Springer, 2005.

[BCOP04] Dan Boneh, Giovanni Di Crescenzo, Rafail Ostrovsky, and Giuseppe Persiano. Public key encryption with keyword search. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques*, volume 3027 of *Lecture Notes in Computer Science*, pages 506–522. Springer, 2004.

[Bel98] Mihir Bellare. Practice-oriented provable security. In *Lectures on Data Security, Modern Cryptology in Theory and Practice, Summer School, Aarhus, Denmark, July 1998*, pages 1–15, 1998.

[Ber85] Claude Berge. *Graphs*. North-Holland Mathematical Library, 1985.

[BF01] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the Weil pairing. In Joe Kilian, editor, *Advances in Cryptology - CRYPTO 2001, 21st Annual International Cryptology Conference*, volume 2139 of *Lecture Notes in Computer Science*, pages 213–229. Springer, 2001.

[BG92] Mihir Bellare and Oded Goldreich. On defining proofs of knowledge. In Ernest F. Brickell, editor, *Advances in Cryptology - CRYPTO '92, 12th Annual International Cryptology Conference*, volume 740 of *Lecture Notes in Computer Science*, pages 390–420. Springer, 1992.

[BGG+14] Dan Boneh, Craig Gentry, Sergey Gorbunov, Shai Halevi, Valeria Nikolaenko, Gil Segev, Vinod Vaikuntanathan, and Dhinakaran Vinayagamurthy. Fully key-homomorphic encryption, arithmetic circuit ABE and compact garbled circuits. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology - EUROCRYPT 2014 - 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques*, volume 8441 of *Lecture Notes in Computer Science*, pages 533–556. Springer, 2014.

[BGJT14] Razvan Barbulescu, Pierrick Gaudry, Antoine Joux, and Emmanuel Thomé. A heuristic quasi-polynomial algorithm for discrete logarithm in finite fields of small characteristic. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology - EUROCRYPT 2014 - 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques*, volume 8441 of *Lecture Notes in Computer Science*, pages 1–16. Springer, 2014.

[BGN05] Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. Evaluating 2-dnf formulas on ciphertexts. In Joe Kilian, editor, *Second Theory of Cryptography Conference, TCC 2005*, volume 3378 of *Lecture Notes in Computer Science*, pages 325–341. Springer, 2005.

[BGV12] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. In Shafi Goldwasser, editor, *Innovations in Theoretical Computer Science 2012*, pages 309–325. ACM, 2012.

[BHR12] Mihir Bellare, Viet Tung Hoang, and Phillip Rogaway. Foundations of garbled circuits. In Ting Yu, George Danezis, and Virgil D. Gligor, editors, *the ACM Conference on Computer and Communications Security, CCS'12*, pages 784–796, 2012.

[BK15] Mihir Bellare and Sriram Keelveedhi. Interactive message-locked encryption and secure deduplication. In Jonathan Katz, editor, *Public-Key Cryptography - PKC 2015 - 18th IACR International Conference on Practice and Theory in Public-Key Cryptography*, volume 9020 of *Lecture Notes in Computer Science*, pages 516–538. Springer, 2015.

[BKKS11] Bostjan Bresar, Frantisek Kardos, Ján Katrenic, and Gabriel Semanisin. Minimum k-path vertex cover. *Discrete Applied Mathematics*, 159(12) :1189–1195, 2011.

[BKR13] Mihir Bellare, Sriram Keelveedhi, and Thomas Ristenpart. Message-locked encryption and secure deduplication. In Thomas Johansson and Phong Q. Nguyen,

editors, *Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques*, volume 7881 of *Lecture Notes in Computer Science*, pages 296–312. Springer, 2013.

[BLLN13] Joppe W. Bos, Kristin E. Lauter, Jake Loftus, and Michael Naehrig. Improved security for a ring-based fully homomorphic encryption scheme. In Martijn Stam, editor, *Cryptography and Coding - 14th IMA International Conference, IMACC 2013*, volume 8308 of *Lecture Notes in Computer Science*, pages 45–64. Springer, 2013.

[BLMZ17] Fabrice Benhamouda, Tancrède Lepoint, Claire Mathieu, and Hang Zhou. Optimization of bootstrapping in circuits. In Philip N. Klein, editor, *Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017*, pages 2423–2433. SIAM, 2017.

[Blu16] BlueCoat. Integrating the proxysg and proxyav appliances, Août 2016. https://bto.bluecoat.com/sites/default/files/tech_pubs/SG_AV_Integration_1.pdf.

[BMR90] Donald Beaver, Silvio Micali, and Phillip Rogaway. The round complexity of secure protocols (extended abstract). In Harriet Ortiz, editor, *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing*, pages 503–513, 1990.

[BP16] Zvika Brakerski and Renen Perlman. Lattice-based fully dynamic multi-key FHE with short ciphertexts. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Part I*, volume 9814 of *Lecture Notes in Computer Science*, pages 190–213. Springer, 2016.

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical : A paradigm for designing efficient protocols. In Dorothy E. Denning, Raymond Pyle, Ravi Ganesan, Ravi S. Sandhu, and Victoria Ashby, editors, *CCS '93, Proceedings of the 1st ACM Conference on Computer and Communications Security*, pages 62–73, 1993.

[Bra12] Zvika Brakerski. Fully homomorphic encryption without modulus switching from classical gapsvp. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology - CRYPTO 2012 - 32nd Annual Cryptology Conference*, volume 7417 of *Lecture Notes in Computer Science*, pages 868–886. Springer, 2012.

[BSW06] Dan Boneh, Amit Sahai, and Brent Waters. Fully collusion resistant traitor tracing with short ciphertexts and private keys. In Serge Vaudenay, editor, *Advances in Cryptology - EUROCRYPT 2006, 25th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, volume 4004 of *Lecture Notes in Computer Science*, pages 573–592. Springer, 2006.

[BSW11] Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption : Definitions and challenges. In Yuval Ishai, editor, *Theory of Cryptography - 8th Theory of Cryptography Conference, TCC2011*, volume 6597 of *Lecture Notes in Computer Science*, pages 253–273. Springer, 2011.

[BV11a] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In Rafail Ostrovsky, editor, *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011*, pages 97–106. IEEE Computer Society, 2011.

[BV11b] Zvika Brakerski and Vinod Vaikuntanathan. Fully homomorphic encryption from ring-lwe and security for key dependent messages. In Phillip Rogaway, editor, *Advances in Cryptology - CRYPTO 2011 - 31st Annual Cryptology Conference*, volume 6841 of *Lecture Notes in Computer Science*, pages 505–524. Springer, 2011.

[BV16] Zvika Brakerski and Vinod Vaikuntanathan. Circuit-ABE from LWE : unbounded attributes and semi-adaptive security. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Part III*, volume 9816 of *Lecture Notes in Computer Science*, pages 363–384. Springer, 2016.

[BW07] Dan Boneh and Brent Waters. Conjunctive, subset, and range queries on encrypted data. In Salil P. Vadhan, editor, *Theory of Cryptography, 4th Theory of Cryptography Conference, TCC 2007*, volume 4392 of *Lecture Notes in Computer Science*, pages 535–554. Springer, 2007.

[Cam98] Jan Camenisch. *Group Signature Schemes and Payment Systems Based on the Discrete Logarithm Problem.* Phd thesis, ETH Zürich, 1998.

[CCK+13] Jung Hee Cheon, Jean-Sébastien Coron, Jinsu Kim, Moon Sung Lee, Tancrède Lepoint, Mehdi Tibouchi, and Aaram Yun. Batch fully homomorphic encryption over the integers. In Thomas Johansson and Phong Q. Nguyen, editors, *Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques*, volume 7881 of *Lecture Notes in Computer Science*, pages 315–335. Springer, 2013.

[CDK+17] Sébastien Canard, Aïda Diop, Nizar Kheir, Marie Paindavoine, and Mohamed Sabt. BlindIDS : Market-compliant and privacy-friendly intrusion detection system over encrypted traffic. In Ramesh Karri, Ozgur Sinanoglu, Ahmad-Reza Sadeghi, and Xun Yi, editors, *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security, AsiaCCS 2017*, pages 561–574, 2017.

[CDKP16a] Sébastien Canard, Aida Diop, Nizar Kheir, and Marie Paindavoine. Procédé et dispositif de détection d'intrusions sur un réseau utilisant un algorithme de chiffrement homomorphe, Octobre 2016. Brevet FR 1659537.

[CDKP16b] Sébastien Canard, Aida Diop, Nizar Kheir, and Marie Paindavoine. Procédé et système de détection d'intrusions sur un réseau, Septembre 2016. Brevet FR 1659428.

[CDS15] Sergiu Carpov, Paul Dubrulle, and Renaud Sirdey. Armadillo : A compilation chain for privacy preserving applications. In Feng Bao, Steven Miller, Sherman S. M. Chow, and Danfeng Yao, editors, *Proceedings of the 3rd International Workshop on Security in Cloud Computing, SCC@ASIACCS '15*, pages 13–19, 2015.

[CFGL12] Sébastien Canard, Georg Fuchsbauer, Aline Gouget, and Fabien Laguillaumie. Plaintext-checkable encryption. In Orr Dunkelman, editor, *Topics in Cryptology - CT-RSA 2012 - The Cryptographers' Track at the RSA Conference 2012*, volume 7178 of *Lecture Notes in Computer Science*, pages 332–348. Springer, 2012.

[CGGI16] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. Faster fully homomorphic encryption : Bootstrapping in less than 0.1 seconds. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *Advances in Cryptology - ASIACRYPT 2016 - 22nd International Conference on the Theory and Application of Cryptology and Information Security, Part I*, volume 10031 of *Lecture Notes in Computer Science*, pages 3–33, 2016.

[CGH98] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited (preliminary version). In Jeffrey Scott Vitter, editor, *Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing*, pages 209–218, 1998.

[CGH04] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited. *J. ACM*, 51(4) :557–594, 2004.

[CGH+15] Jean-Sébastien Coron, Craig Gentry, Shai Halevi, Tancrède Lepoint, Hemanta K. Maji, Eric Miles, Mariana Raykova, Amit Sahai, and Mehdi Tibouchi. Zeroizing without low-level zeroes : New MMAP attacks and their limitations. In Rosario Gennaro and Matthew Robshaw, editors, *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Part I*, volume 9215 of *Lecture Notes in Computer Science*, pages 247–266. Springer, 2015.

[CGS97] Ronald Cramer, Rosario Gennaro, and Berry Schoenmakers. A secure and optimally efficient multi-authority election scheme. In Walter Fumy, editor, *Advances in Cryptology - EUROCRYPT '97, International Conference on the Theory and Application of Cryptographic Techniques*, volume 1233 of *Lecture Notes in Computer Science*, pages 103–118. Springer, 1997.

[CL12] Guilhem Castagnos and Fabien Laguillaumie. Homomorphic encryption for multiplications and pairing evaluation. In Ivan Visconti and Roberto De Prisco, editors, *Security and Cryptography for Networks - 8th International Conference, SCN 2012*, volume 7485 of *Lecture Notes in Computer Science*, pages 374–392. Springer, 2012.

[CLP16] Sébastien Canard, Fabien Laguillaumie, and Marie Paindavoine. Verifiable message-locked encryption. In Sara Foresti and Giuseppe Persiano, editors, *Cryptology and Network Security - 15th International Conference, CANS 2016*, volume 10052 of *Lecture Notes in Computer Science*, pages 299–315, 2016.

[CLT14] Jean-Sébastien Coron, Tancrède Lepoint, and Mehdi Tibouchi. Scale-invariant fully homomorphic encryption over the integers. In Hugo Krawczyk, editor, *Public-Key Cryptography - PKC 2014 - 17th International Conference on Practice and Theory in Public-Key Cryptography*, volume 8383 of *Lecture Notes in Computer Science*, pages 311–328. Springer, 2014.

[CMNT11] Jean-Sébastien Coron, Avradip Mandal, David Naccache, and Mehdi Tibouchi. Fully homomorphic encryption over the integers with shorter public keys. In Phillip Rogaway, editor, *Advances in Cryptology - CRYPTO 2011 - 31st Annual Cryptology Conference*, volume 6841 of *Lecture Notes in Computer Science*, pages 487–504. Springer, 2011.

[CNT12] Jean-Sébastien Coron, David Naccache, and Mehdi Tibouchi. Public key compression and modulus switching for fully homomorphic encryption over the integers. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology - EUROCRYPT 2012 - 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques*, volume 7237 of *Lecture Notes in Computer Science*, pages 446–464. Springer, 2012.

[Coc01] Clifford Cocks. An identity based encryption scheme based on quadratic residues. In Bahram Honary, editor, *Cryptography and Coding, 8th IMA International Conference*, volume 2260 of *Lecture Notes in Computer Science*, pages 360–363. Springer, 2001.

[CPP16] Geoffroy Couteau, Thomas Peters, and David Pointcheval. Encryption switching protocols. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Part I*, volume 9814 of *Lecture Notes in Computer Science*, pages 308–338. Springer, 2016.

[CS90] Pierluigi Crescenzi and Riccardo Silvestri. Relative complexity of evaluating the optimum cost and constructing the optimum for maximization problems. *Inf. Process. Lett.*, 33(5) :221–226, 1990.

[CS97] Jan Camenisch and Markus Stadler. Efficient group signature schemes for large groups (extended abstract). In Burton S. Kaliski Jr., editor, *Advances in Cryptology - CRYPTO '97, 17th Annual International Cryptology Conference*, volume 1294 of *Lecture Notes in Computer Science*, pages 410–424. Springer, 1997.

[CS15] Jung Hee Cheon and Damien Stehlé. Fully homomophic encryption over the integers revisited. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Part I*, volume 9056 of *Lecture Notes in Computer Science*, pages 513–536. Springer, 2015.

[CV08] Kai-Min Chung and Salil P. Vadhan. Tight bounds for hashing block sources. In Ashish Goel, Klaus Jansen, José D. P. Rolim, and Ronitt Rubinfeld, editors, *Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques, 11th International Workshop, APPROX 2008, and 12th International Workshop, RANDOM 2008*, volume 5171 of *Lecture Notes in Computer Science*, pages 357–370. Springer, 2008.

[DAB+02] John R. Douceur, Atul Adya, William J. Bolosky, Dan Simon, and Marvin Theimer. Reclaiming space from duplicate files in a serverless distributed file system. In *ICDCS*, pages 617–624, 2002.

[Dam98] Ivan Damgård. Commitment schemes and zero-knowledge protocols. In *Lectures on Data Security, Modern Cryptology in Theory and Practice, Summer School, Aarhus, Denmark, July 1998*, pages 63–86, 1998.

[Del16] Dell. Dell security annual threat report, 2016. http://www.netthreat.co.uk/assets/assets/dell-security-annual-threat-report-2016-white-paper_197571.pdf.

[DM15] Léo Ducas and Daniele Micciancio. FHEW : bootstrapping homomorphic encryption in less than a second. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I*, volume 9056, pages 617–640. Springer, 2015.

[FP07] Thomas Fuhr and Pascal Paillier. Decryptable searchable encryption. In Willy Susilo, Joseph K. Liu, and Yi Mu, editors, *Provable Security, First International Conference, ProvSec 2007*, volume 4784 of *Lecture Notes in Computer Science*, pages 228–236. Springer, 2007.

[Fro16] Reporter Sans Frontières. Les ennemis d'internet, rapport spécial surveillance, Consulté le 4 octobre 2016. http://surveillance.rsf.org/en/blue-coat-2/.

[FS86] Amos Fiat and Adi Shamir. How to prove yourself : Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *Advances in Cryptology - CRYPTO '86, 7th Annual International Cryptology Conference*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194. Springer, 1986.

[FSF+13] Simon Fau, Renaud Sirdey, Caroline Fontaine, Carlos Aguilar Melchor, and Guy Gogniat. Towards practical program execution over fully homomorphic encryption schemes. In Fatos Xhafa, Leonard Barolli, Dritan Nace, Salvatore Venticinque, and Alain Bui, editors, *Eighth International Conference on P2P, Parallel, Grid, Cloud and Internet Computing, 3PGCIC 2013, Compiegne, France, October 28-30, 2013*, pages 284–290. IEEE, 2013.

[FV12] Junfeng Fan and Frederik Vercauteren. Somewhat practical fully homomorphic encryption. *IACR Cryptology ePrint Archive*, 2012 :144, 2012. http://eprint.iacr.org/2012/144.

[Gam84] Taher El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In G. R. Blakley and David Chaum, editors, *Advances in Cryptology - CRYPTO '84, 5th Annual International Cryptology Conference*, volume 196 of *Lecture Notes in Computer Science*, pages 10–18. Springer, 1984.

[Gen09a] Craig Gentry. *A fully homomorphic encryption scheme*. PhD thesis, Stanford University, 2009. crypto.stanford.edu/craig.

[Gen09b] Craig Gentry. Fully homomorphic encryption using ideal lattices. In Michael Mitzenmacher, editor, *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009*, pages 169–178, 2009.

[Gen14] Craig Gentry. Computing on the edge of chaos : Structure and randomness in encrypted computation. Cryptology ePrint Archive, Report 2014/610, 2014. http://eprint.iacr.org/2014/610.

[GGH+13] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013*, pages 40–49. IEEE Computer Society, 2013.

[GH11] Craig Gentry and Shai Halevi. Implementing Gentry's fully-homomorphic encryption scheme. In Kenneth G. Paterson, editor, *Advances in Cryptology - EUROCRYPT 2011 - 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, volume 6632 of *Lecture Notes in Computer Science*, pages 129–148. Springer, 2011.

[GHS12] Craig Gentry, Shai Halevi, and Nigel P. Smart. Homomorphic evaluation of the AES circuit. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology - CRYPTO 2012 - 32nd Annual Cryptology Conference*, volume 7417 of *Lecture Notes in Computer Science*, pages 850–867. Springer, 2012.

[GINX16] Nicolas Gama, Malika Izabachène, Phong Q. Nguyen, and Xiang Xie. Structural lattice reduction : Generalized worst-case to average-case reductions and homomorphic cryptosystems. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Part II*, volume 9666 of *Lecture Notes in Computer Science*, pages 528–558. Springer, 2016.

[GM84] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *J. Comput. Syst. Sci.*, 28(2) :270–299, 1984.

[GPS08] Steven D. Galbraith, Kenneth G. Paterson, and Nigel P. Smart. Pairings for cryptographers. *Discrete Applied Mathematics*, 156(16) :3113–3121, 2008.

[Gro10] Jens Groth. Short non-interactive zero-knowledge proofs. In Masayuki Abe, editor, *Advances in Cryptology - ASIACRYPT 2010 - 16th International Conference on the Theory and Application of Cryptology and Information Security*, volume 6477 of *Lecture Notes in Computer Science*, pages 341–358. Springer, 2010.

[GSW13] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors : Conceptually-simpler, asymptotically-faster, attribute-based. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Part I*, volume 8042 of *Lecture Notes in Computer Science*, pages 75–92. Springer, 2013.

[GT86] Andrew V. Goldberg and Robert Endre Tarjan. A new approach to the maximum flow problem. In *Proceedings of the 18th Annual ACM Symposium on Theory of Computing, May 28-30, 1986, Berkeley, California, USA*, pages 136–146, 1986.

[GVW13] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Attribute-based encryption for circuits. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *Symposium on Theory of Computing Conference, STOC'13*, pages 545–554. ACM, 2013.

[GVW15] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Predicate encryption for circuits from LWE. In Rosario Gennaro and Matthew Robshaw, editors, *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Part II*, volume 9216 of *Lecture Notes in Computer Science*, pages 503–523. Springer, 2015.

[HBB15] Rick Holland, Stephanie Balaouras, and Josh Blackborow. The state of the cyber-threat intelligence market. In *Forrester report*, 2015.

[HEK12] Yan Huang, David Evans, and Jonathan Katz. Private set intersection : Are garbled circuits better than custom protocols ? In *19th Annual Network and Distributed System Security Symposium, NDSS, 2012*. The Internet Society, 2012.

[HMEK11] Yan Huang, Lior Malka, David Evans, and Jonathan Katz. Efficient privacy-preserving biometric identification. In *Proceedings of the Network and Distributed System Security Symposium, NDSS 2011*. The Internet Society, 2011.

[HREJ14] Lin-Shung Huang, Alex Rice, Erling Ellingsen, and Collin Jackson. Analyzing forged SSL certificates in the wild. In *2014 IEEE Symposium on Security and Privacy, SP 2014*, pages 83–97. IEEE Computer Society, 2014.

[HS14] Shai Halevi and Victor Shoup. Algorithms in helib. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Part I*, volume 8616 of *Lecture Notes in Computer Science*, pages 554–571. Springer, 2014.

[HS15] Shai Halevi and Victor Shoup. Bootstrapping for helib. In *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Part I*, pages 641–670, 2015.

[IKNP03] Yuval Ishai, Joe Kilian, Kobbi Nissim, and Erez Petrank. Extending oblivious transfers efficiently. In Dan Boneh, editor, *Advances in Cryptology - CRYPTO 2003, 23rd Annual International Cryptology Conference*, volume 2729 of *Lecture Notes in Computer Science*, pages 145–161. Springer, 2003.

[ILL89] Russell Impagliazzo, Leonid A. Levin, and Michael Luby. Pseudo-random generation from one-way functions (extended abstracts). In David S. Johnson, editor, *21st Annual ACM Symposium on Theory of Computing*, pages 12–24, 1989.

[Jar12] Jeff Jarmoc. Ssl interception proxies and transitive trust. In *Black Hat Europe*, 2012.

[Jou00] Antoine Joux. A one round protocol for tripartite diffie-hellman. In Wieb Bosma, editor, *Algorithmic Number Theory, 4th International Symposium, ANTS-IV*, volume 1838 of *Lecture Notes in Computer Science*, pages 385–394. Springer, 2000.

[JP16] Eric Horvitz John Paparrizos, Ryen White. Screening for pancreatic adenocarcinoma using signals from web search logs : Feasibility study and results. *Journal of Oncology Practice*, 12 No. 8 :737–744, August 2016.

[Kah96] David Kahn. *The codebreakers : the story of secret writing.* Scribner, New York, 1996.

[Kar72] Richard M. Karp. Reducibility among combinatorial problems. In *Proceedings of a symposium on the Complexity of Computer Computations, held March 20-22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York.*, pages 85–103, 1972.

[KK13] Vladimir Kolesnikov and Ranjit Kumaresan. Improved OT extension for transferring short secrets. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Part II*, volume 8043 of *Lecture Notes in Computer Science*, pages 54–70. Springer, 2013.

[Kle56] Stephen Cole Kleene. Representation of Events in Nerve Nets and Finite Automata. In C.E. Shannon and J. McCarthy, editors, *Automata Studies*, pages 3–41. Princeton University Press, 1956.

[KSW08] Jonathan Katz, Amit Sahai, and Brent Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In Nigel P. Smart, editor,

Advances in Cryptology - EUROCRYPT 2008, 27th Annual International Conference on the Theory and Applications of Cryptographic Techniques, volume 4965 of *Lecture Notes in Computer Science*, pages 146–162. Springer, 2008.

[Lee11] Moon Sung Lee. On the sparse subset sum problem from Gentry-Halevi's implementation of fully homomorphic encryption. *IACR Cryptology ePrint Archive*, 2011 :567, 2011. http://eprint.iacr.org/2011/567.

[LN14] Tancrède Lepoint and Michael Naehrig. A comparison of the homomorphic encryption schemes FV and YASHE. In David Pointcheval and Damien Vergnaud, editors, *Progress in Cryptology - AFRICACRYPT 2014 - 7th International Conference on Cryptology in Africa*, volume 8469 of *Lecture Notes in Computer Science*, pages 318–335. Springer, 2014.

[LP13] Tancrède Lepoint and Pascal Paillier. On the minimal number of bootstrappings in homomorphic circuits. In Andrew A. Adams, Michael Brenner, and Matthew Smith, editors, *Financial Cryptography and Data Security - FC 2013 Workshops, USEC and WAHC 2013*, volume 7862 of *Lecture Notes in Computer Science*, pages 189–200. Springer, 2013.

[LP16] Kim Laine and Rachel Player. Simple encrypted arithmetic library - seal (v2.0). Technical report, Microsoft Research, September 2016.

[LPM+14] Ronan Lashermes, Marie Paindavoine, Nadia El Mrabet, Jacques J. A. Fournier, and Louis Goubin. Practical validation of several fault attacks against the miller algorithm. In Assia Tria and Dooho Choi, editors, *2014 Workshop on Fault Diagnosis and Tolerance in Cryptography, FDTC*, pages 115–122. IEEE Computer Society, 2014.

[LPR10] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In Henri Gilbert, editor, *Advances in Cryptology - EUROCRYPT 2010, 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, volume 6110 of *Lecture Notes in Computer Science*, pages 1–23. Springer, 2010.

[LTV12] Adriana López-Alt, Eran Tromer, and Vinod Vaikuntanathan. On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In Howard J. Karloff and Toniann Pitassi, editors, *Proceedings of the 44th Symposium on Theory of Computing Conference, STOC 2012*, pages 1219–1234. ACM, 2012.

[Mal] Malware domain list. https://www.malwaredomainlist.com/mdl.php.

[MFF+13] Carlos Aguilar Melchor, Simon Fau, Caroline Fontaine, Guy Gogniat, and Renaud Sirdey. Recent advances in homomorphic encryption : A possible future for signal processing in the encrypted domain. *IEEE Signal Process. Mag.*, 30(2) :108–117, 2013.

[MGBF14] Benjamin Mood, Debayan Gupta, Kevin R. B. Butler, and Joan Feigenbaum. Reuse it or lose it : More efficient secure computation through reuse of encrypted values. In Gail-Joon Ahn, Moti Yung, and Ninghui Li, editors, *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 582–596, 2014.

[MGC+16] Benjamin Mood, Debayan Gupta, Henry Carter, Kevin R. B. Butler, and Patrick Traynor. Frigate : A validated, extensible, and efficient compiler and interpreter for secure computation. In *IEEE European Symposium on Security and Privacy, EuroS&P 2016*, pages 112–127, 2016.

[MIT13] Shigeo MITSUNARI. A fast implementation of the optimal Ate pairing over BN curve on Intel Haswell processor. Cryptology ePrint Archive, Report 2013/362, 2013. http://eprint.iacr.org/2013/362.

[MM15] Markets and Markets. Threat intelligence security market by solution - global forecast to 2020. In *MarketsandMarkets report TC 3591*, 2015.

[MSZ16] Eric Miles, Amit Sahai, and Mark Zhandry. Annihilation attacks for multilinear maps : Cryptanalysis of indistinguishability obfuscation over GGH13. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Part II*, volume 9815 of *Lecture Notes in Computer Science*, pages 629–658. Springer, 2016.

[MVO91] Alfred Menezes, Scott A. Vanstone, and Tatsuaki Okamoto. Reducing elliptic curve logarithms to logarithms in a finite field. In Cris Koutsougeras and Jeffrey Scott Vitter, editors, *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing*, pages 80–89, 1991.

[NP01] Moni Naor and Benny Pinkas. Efficient oblivious transfer protocols. In S. Rao Kosaraju, editor, *Proceedings of the Twelfth Annual Symposium on Discrete Algorithms*, pages 448–457. ACM/SIAM, 2001.

[NS00] Toru Nakanishi and Yuji Sugiyama. Unlinkable divisible electronic cash. In Josef Pieprzyk, Eiji Okamoto, and Jennifer Seberry, editors, *Information Security, Third International Workshop, ISW 2000*, volume 1975 of *Lecture Notes in Computer Science*, pages 121–134. Springer, 2000.

[NS09] Arvind Narayanan and Vitaly Shmatikov. De-anonymizing social networks. In *30th IEEE Symposium on Security and Privacy (S&P 2009)*, pages 173–187. IEEE Computer Society, 2009.

[NTI16] NTIA. Lack of trust in internet privacy and security may deter economic and other online activities, Mai 2016. https://www.ntia.doc.gov/.

[Oba08] Barack Obama. Interview with Google CEO Eric Schmidt, Janvier 2008. https://www.youtube.com/watch?v=k4RRi_ntQc8.

[Ora] Orange. Data for development. http://www.d4d.orange.com/fr/Accueil.

[Pai99] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In Jacques Stern, editor, *Advances in Cryptology - EUROCRYPT '99, International Conference on the Theory and Application of Cryptographic Techniques*, volume 1592 of *Lecture Notes in Computer Science*, pages 223–238. Springer, 1999.

[PBS11] Henning Perl, Michael Brenner, and Matthew Smith. Poster : an implementation of the fully homomorphic Smart-Vercauteren crypto-system. In Yan Chen, George Danezis, and Vitaly Shmatikov, editors, *18th ACM Conference on Computer and Communications Security, CCS 2011*, pages 837–840, 2011.

[Ped91] Torben P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In Joan Feigenbaum, editor, *Advances in Cryptology - CRYPTO '91, 11th Annual International Cryptology Conference*, volume 576 of *Lecture Notes in Computer Science*, pages 129–140. Springer, 1991.

[Pre] Cafe Presse. You're the product. http://www.cafepress.com/youretheproduct.

[PS00] David Pointcheval and Jacques Stern. Security arguments for digital signatures and blind signatures. *J. Cryptology*, 13(3) :361–396, 2000.

[PV15] Marie Paindavoine and Bastien Vialla. Minimizing the number of bootstrappings in fully homomorphic encryption. In Orr Dunkelman and Liam Keliher, editors, *Selected Areas in Cryptography - SAC 2015 - 22nd International Conference, Revised Selected Papers*, volume 9566 of *Lecture Notes in Computer Science*, pages 25–43. Springer, 2015.

[Rab81] Michael O. Rabin. How to exchange secrets with oblivious transfer, 1981. Harvard University Technical Report 81 http://eprint.iacr.org/2005/187.

[RAD78] Ronald L Rivest, Leonard Adleman, and Miachael L Dertouzos. On data banks and privacy homomorphisms. *Foundations of Secure Computation, Academia Press*, 4 :169–179, 1978.

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *Proceedings of the 37th Annual ACM Symposium on Theory of Computing, 2005*, pages 84–93, 2005.

[RSA78] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2) :120–126, 1978.

[RSV13] Ananth Raghunathan, Gil Segev, and Salil P. Vadhan. Deterministic public-key encryption for adaptively chosen plaintext distributions. In Thomas Johansson and Phong Q. Nguyen, editors, *Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques*, volume 7881 of *Lecture Notes in Computer Science*, pages 93–110. Springer, 2013.

[SBC+07] Elaine Shi, John Bethencourt, Hubert T.-H. Chan, Dawn Xiaodong Song, and Adrian Perrig. Multi-dimensional range query over encrypted data. In *2007 IEEE Symposium on Security and Privacy (S&P 2007)*, pages 350–364. IEEE Computer Society, 2007.

[Sch89] Claus-Peter Schnorr. Efficient identification and signatures for smart cards. In Gilles Brassard, editor, *Advances in Cryptology - CRYPTO '89, 9th Annual International Cryptology Conference*, volume 435 of *Lecture Notes in Computer Science*, pages 239–252. Springer, 1989.

[Sha84] Adi Shamir. Identity-based cryptosystems and signature schemes. In G. R. Blakley and David Chaum, editors, *Advances in Cryptology - CRYPTO 1984 - 4th Annual Cryptology Conference*, volume 196 of *Lecture Notes in Computer Science*, pages 47–53. Springer, 1984.

[Sho97] Victor Shoup. Lower bounds for discrete logarithms and related problems. In Walter Fumy, editor, *Advances in Cryptology - EUROCRYPT '97, International Conference on the Theory and Application of Cryptographic Techniques*, volume 1233 of *Lecture Notes in Computer Science*, pages 256–266. Springer, 1997.

[Sho04] Victor Shoup. Sequences of games : a tool for taming complexity in security proofs. *IACR Cryptology ePrint Archive*, 2004 :332, 2004. http://eprint.iacr.org/2004/332.

[SLPR15] Justine Sherry, Chang Lan, Raluca Ada Popa, and Sylvia Ratnasamy. Blindbox : Deep packet inspection over encrypted traffic. In Steve Uhlig, Olaf Maennel, Brad Karp, and Jitendra Padhye, editors, *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication, SIGCOMM 2015*, pages 213–226, 2015.

[Sno] Snort. Snort intrusion detection system. http://manual-snort-org.s3-website-us-east-1.amazonaws.com/node27.html.

[SPV+16] Thomas Skybakmoen, Jayendra Pathak, Bhaarath Venkateswaran, Mike Spanbauer, and Bob Walder. Breach detection systems comparative report. In *NSS Labs Security Value Map*, 2016.

[ST] Nigel Smart and Stefan Tillich. Circuits of basic functions suitable for MPC and FHE. https://www.cs.bris.ac.uk/Research/CryptographySecurity/MPC/.

[Sta96] Markus Stadler. Publicly verifiable secret sharing. In Ueli M. Maurer, editor, *Advances in Cryptology - EUROCRYPT '96, International Conference on the Theory and Application of Cryptographic Techniques*, volume 1070 of *Lecture Notes in Computer Science*, pages 190–199. Springer, 1996.

[SW05] Amit Sahai and Brent Waters. Fuzzy identity-based encryption. In Ronald Cramer, editor, *Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, volume 3494 of *Lecture Notes in Computer Science*, pages 457–473. Springer, 2005.

[Sym15] Symantec. Internet security threat report, 2015. https://www.itu.int/en/ITU-D/Cybersecurity/Documents/Symantec_annual_internet_threat_report_ITU2015.pdf.

[Thr] Emerging Threats. Emerging threats rules. https://rules.emergingthreats.net/.

[URL] URL. Url blacklist. http://www.urlblacklist.com/?sec=home.

[vDGHV10] Marten van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. Fully homomorphic encryption over the integers. In Henri Gilbert, editor, *Advances in Cryptology - EUROCRYPT 2010, 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, volume 6110 of *Lecture Notes in Computer Science*, pages 24–43. Springer, 2010.

[Vig] Giovanni Vigna. The UC Santa Barbara iCTF competition. https://ictf.cs.ucsb.edu/#/.

[Vol] Voltage. https ://www.voltage.com/technology/data-encryption/identity-based-encryption/.

[Yao82] Andrew Chi-Chih Yao. Protocols for secure computations (extended abstract). In *23rd Annual Symposium on Foundations of Computer Science*, pages 160–164. IEEE Computer Society, 1982.

[Yao86] Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *27th Annual Symposium on Foundations of Computer Science*, pages 162–167. IEEE Computer Society, 1986.

[Yar] Yara. Rules repository. https://github.com/Yara-Rules/rules.

[YTHW10] Guomin Yang, Chik How Tan, Qiong Huang, and Duncan S. Wong. Probabilistic public key encryption with equality test. In Josef Pieprzyk, editor, *Topics in Cryptology - CT-RSA 2010, The Cryptographers' Track at the RSA Conference 2010*, volume 5985 of *Lecture Notes in Computer Science*, pages 119–131. Springer, 2010.

[ZRE15] Samee Zahur, Mike Rosulek, and David Evans. Two halves make a whole - reducing data transfer in garbled circuits using half gates. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Part II*, volume 9057 of *Lecture Notes in Computer Science*, pages 220–250. Springer, 2015.

Résumé

L'annonce de l'essor du chiffrement des données se heurte à celle de l'avènement du « big data ». Il n'est maintenant plus suffisant d'envoyer et de recevoir des données, il faut pouvoir les analyser, les exploiter ou encore les partager à grande échelle. Or, les données à protéger sont de plus en plus nombreuses, notamment avec la prise de conscience de l'impact qu'ont les nouvelles technologies (smartphones, internet of things, cloud, ...) sur la vie privée des utilisateurs. En rendant ces données inaccessibles, le chiffrement bloque a priori les fonctionnalités auxquelles les utilisateurs et les fournisseurs de service sont habitués. Pour rétablir ces fonctionnalités, il est nécessaire de savoir calculer des fonctions de données chiffrées, et cette thèse explore plusieurs pistes dans ce sens.

Dans une première partie, nous nous intéressons au chiffrement totalement homomorphe qui permet de réaliser des calculs arbitraires sur les données chiffrées. Ce type de chiffrement est cependant particulièrement coûteux, notamment à cause de l'appel souvent nécessaire à une procédure très coûteuse : le réamorçage. Nous prouvons ici que minimiser le nombre de réamorçages est un problème NP-complet et donnons une méthode pratique pour approximer ce minimum.

Dans une seconde partie, nous étudions des schémas dédiés à une fonctionnalité donnée. Le premier cas d'usage considéré est celui de la déduplication vérifiable de données chiffrées. Il s'agit pour un serveur de stockage externe d'être assuré qu'il ne conserve qu'un seul exemplaire de chaque fichier, même si ceux-ci sont chiffrés, ce qui lui permet d'optimiser l'usage de ses ressources mémoires. Ensuite, nous proposons un schéma de chiffrement cherchable permettant de détecter des intrusions dans un réseau de télécommunications chiffrés. En effet, le travail d'inspection du réseau par des moteurs d'analyse est actuellement entravé par la croissance du trafic chiffré. Les résultats obtenus permettent ainsi d'assurer la confidentialité des échanges tout en garantissant l'absence d'intrusions malveillantes dans le trafic.

Abstract

Nowadays, encryption and services issued of "big data" are at odds. Indeed, encryption is about protecting users privacy, while big data is about analyzing users data. Being increasingly concerned about security, users tend to encrypt their sensitive data that are subject to be accessed by other parties, including service providers. This hinders the execution of services requiring some kind of computation on users data, which makes users under obligation to choose between these services or their private life. We address this challenge in this thesis by following two directions.

In the first part of this thesis, we study fully homomorphic encryption that makes possible to perform arbitrary computation on encrypted data. However, this kind of encryption is still inefficient, and this is due in part to the frequent execution of a costly procedure throughout evaluation, namely the bootstrapping. Thus, efficiency is inversely proportional to the number of bootstrappings needed to evaluate functions on encrypted data. In this thesis, we prove that finding such a minimum is NP-complete. In addition, we design a new method that efficiently finds a good approximation of it.

In the second part, we design schemes that allow a precise functionality. The first one is verifiable deduplication on encrypted data, which allows a server to be sure that it keeps only one copy of each file uploaded, even if the files are encrypted, resulting in an optimization of the storage resources. The second one is intrusion detection over encrypted traffic. Current encryption techniques blinds intrusion detection services, putting the final user at risks. Our results permit to reconcile users' right to privacy and their need of keeping their network clear of all intrusion.